

IBM DB2 10.1  
for Linux, UNIX, and Windows

*Command Reference*





IBM DB2 10.1  
for Linux, UNIX, and Windows

*Command Reference*



**Note**

Before using this information and the product it supports, read the general information under Appendix D, "Notices," on page 1291.

**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 <http://www.ibm.com/shop/publications/order>
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at <http://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 IBM Corporation 1993, 2012.

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

---

# Contents

<b>About this book.</b> . . . . .	<b>vii</b>
Who should use this book . . . . .	vii
How this book is structured. . . . .	vii
Highlighting conventions . . . . .	viii

<b>How to read the syntax diagrams</b> . . . .	<b>ix</b>
--	-----------

## **Chapter 1. Command line processor (CLP)** . . . . . **1**

Command line processor features . . . . .	2
db2 - Command line processor invocation . . . . .	8
Command line processor options . . . . .	10
Command line processor return codes . . . . .	18
Invoking command help from the command line processor . . . . .	18
Invoking message help from the command line processor . . . . .	19

## **Chapter 2. Using command line SQL statements and XQuery statements** . . . **21**

## **Chapter 3. Command line processor plus (CLPPlus)** . . . . . **29**

Installing CLPPlus on existing IBM data server clients . . . . .	29
Starting CLPPlus . . . . .	30
CLPPLUS command . . . . .	31
CLPPlus console types. . . . .	33
Window mode and UTF-8 character support . . . . .	35
DSN aliases in CLPPlus . . . . .	35
Kerberos authentication in CLPPlus . . . . .	37
SERVER_ENCRYPT authentication in CLPPlus . . . . .	38
LDAP support in CLPPlus . . . . .	39
CREATE DATABASE in CLPPlus . . . . .	43
Running a script file in CLPPlus . . . . .	44
Skipping and interrupting CLPPlus commands . . . . .	45
Comments in CLPPlus. . . . .	46
Escape characters in CLPPlus . . . . .	47
Bind variables in CLPPlus . . . . .	49
Shell and environment variables in CLPPlus . . . . .	53
CLPPlus traces and record logging . . . . .	54
DB2 commands supported by CLPPlus . . . . .	54
CLPPlus restrictions . . . . .	55
CLPPlus troubleshooting hints and tips . . . . .	56

## **Chapter 4. How to read command syntax help.** . . . . . **59**

## **Chapter 5. CLP commands** . . . . . **63**

ACTIVATE DATABASE . . . . .	64
ADD CONTACT . . . . .	66
ADD CONTACTGROUP . . . . .	67
ADD DBPARTITIONNUM . . . . .	68

ADD XMLSCHEMA DOCUMENT . . . . .	71
ARCHIVE LOG . . . . .	73
ATTACH . . . . .	76
AUTOCONFIGURE . . . . .	78
BACKUP DATABASE . . . . .	81
BIND . . . . .	90
CATALOG DATABASE . . . . .	110
CATALOG DCS DATABASE . . . . .	113
CATALOG LDAP DATABASE. . . . .	115
CATALOG LDAP NODE . . . . .	118
CATALOG LOCAL NODE . . . . .	119
CATALOG NAMED PIPE NODE. . . . .	121
CATALOG ODBC DATA SOURCE . . . . .	123
CATALOG TCPIP/TCPIP4/TCPIP6 NODE . . . . .	125
CHANGE DATABASE COMMENT . . . . .	129
CHANGE ISOLATION LEVEL . . . . .	131
COMPLETE XMLSCHEMA . . . . .	133
CREATE DATABASE. . . . .	134
CREATE TOOLS CATALOG . . . . .	152
DEACTIVATE DATABASE . . . . .	154
DECOMPOSE XML DOCUMENT . . . . .	156
DECOMPOSE XML DOCUMENTS . . . . .	158
DEREGISTER . . . . .	161
DESCRIBE . . . . .	162
DETACH. . . . .	172
DROP CONTACT . . . . .	173
DROP CONTACTGROUP . . . . .	174
DROP DATABASE . . . . .	175
DROP DBPARTITIONNUM VERIFY. . . . .	177
DROP TOOLS CATALOG . . . . .	178
ECHO. . . . .	180
EDIT . . . . .	181
EXPORT . . . . .	182
FORCE APPLICATION . . . . .	192
GET ADMIN CONFIGURATION. . . . .	194
GET ALERT CONFIGURATION . . . . .	196
GET CLI CONFIGURATION . . . . .	202
GET CONNECTION STATE . . . . .	204
GET CONTACTGROUP. . . . .	205
GET CONTACTGROUPS . . . . .	206
GET CONTACTS . . . . .	207
GET DATABASE CONFIGURATION . . . . .	208
GET DATABASE MANAGER CONFIGURATION . . . . .	221
GET DATABASE MANAGER MONITOR SWITCHES . . . . .	230
GET DESCRIPTION FOR HEALTH INDICATOR . . . . .	233
GET HEALTH NOTIFICATION CONTACT LIST . . . . .	234
GET HEALTH SNAPSHOT. . . . .	235
GET INSTANCE . . . . .	238
GET MONITOR SWITCHES . . . . .	239
GET RECOMMENDATIONS FOR HEALTH INDICATOR. . . . .	242
GET ROUTINE. . . . .	245
GET SNAPSHOT . . . . .	247
HELP . . . . .	263
HISTORY. . . . .	264

IMPORT . . . . .	265	START DATABASE MANAGER . . . . .	614
INGEST . . . . .	294	START HADR . . . . .	623
INITIALIZE TAPE. . . . .	337	STOP DATABASE MANAGER . . . . .	625
INSPECT. . . . .	338	STOP HADR . . . . .	629
LIST ACTIVE DATABASES. . . . .	346	TAKEOVER HADR . . . . .	631
LIST APPLICATIONS . . . . .	348	TERMINATE . . . . .	635
LIST COMMAND OPTIONS . . . . .	351	UNCATALOG DATABASE . . . . .	636
LIST DATABASE DIRECTORY . . . . .	352	UNCATALOG DCS DATABASE . . . . .	637
LIST DATABASE PARTITION GROUPS . . . . .	355	UNCATALOG LDAP DATABASE . . . . .	638
LIST DBPARTITIONNUMS. . . . .	357	UNCATALOG LDAP NODE . . . . .	639
LIST DCS APPLICATIONS . . . . .	358	UNCATALOG NODE . . . . .	640
LIST DCS DIRECTORY . . . . .	360	UNCATALOG ODBC DATA SOURCE . . . . .	641
LIST DRDA INDOUBT TRANSACTIONS . . . . .	362	UNQUIESCE . . . . .	642
LIST HISTORY . . . . .	364	UPDATE ADMIN CONFIGURATION . . . . .	644
LIST INDOUBT TRANSACTIONS . . . . .	369	UPDATE ALERT CONFIGURATION . . . . .	646
LIST NODE DIRECTORY . . . . .	371	UPDATE ALTERNATE SERVER FOR DATABASE . . . . .	651
LIST ODBC DATA SOURCES . . . . .	374	UPDATE ALTERNATE SERVER FOR LDAP . . . . .	653
LIST PACKAGES/TABLES . . . . .	375	UPDATE CLI CONFIGURATION . . . . .	654
LIST TABLESPACE CONTAINERS . . . . .	378	UPDATE COMMAND OPTIONS. . . . .	656
LIST TABLESPACES . . . . .	380	UPDATE CONTACT . . . . .	658
LIST UTILITIES . . . . .	385	UPDATE CONTACTGROUP . . . . .	659
LOAD. . . . .	387	UPDATE DATABASE CONFIGURATION . . . . .	660
LOAD QUERY . . . . .	431	UPDATE DATABASE MANAGER . . . . .	664
MIGRATE DATABASE . . . . .	437	CONFIGURATION . . . . .	664
PING . . . . .	438	UPDATE HEALTH NOTIFICATION CONTACT . . . . .	666
PRECOMPILE . . . . .	440	LIST . . . . .	666
PRUNE HISTORY/LOGFILE . . . . .	466	UPDATE HISTORY . . . . .	667
PUT ROUTINE. . . . .	468	UPDATE LDAP NODE . . . . .	669
QUERY CLIENT . . . . .	469	UPDATE MONITOR SWITCHES . . . . .	671
QUIESCE. . . . .	470	UPDATE XMLSCHEMA. . . . .	673
QUIESCE TABLESPACES FOR TABLE . . . . .	474	UPGRADE DATABASE . . . . .	674
QUIT . . . . .	477		
REBIND . . . . .	478		
RECOVER DATABASE . . . . .	482	<b>Chapter 6. CLPPlus commands. . . . .</b>	<b>677</b>
REDISTRIBUTE DATABASE PARTITION GROUP . . . . .	489	.. . . . .	678
REFRESH LDAP . . . . .	497	!. . . . .	679
REGISTER . . . . .	499	/ . . . . .	680
REGISTER XMLSCHEMA . . . . .	502	@ . . . . .	681
REGISTER XSROBJECT . . . . .	504	@@ . . . . .	682
REORG INDEXES/TABLE . . . . .	506	ACCEPT . . . . .	683
REORGCHK . . . . .	523	APPEND . . . . .	685
RESET ADMIN CONFIGURATION . . . . .	534	BREAK . . . . .	686
RESET ALERT CONFIGURATION . . . . .	536	BTITLE . . . . .	687
RESET DATABASE CONFIGURATION. . . . .	538	CHANGE . . . . .	689
RESET DATABASE MANAGER . . . . .	540	CLEAR . . . . .	691
CONFIGURATION . . . . .	540	COLUMN . . . . .	693
RESET MONITOR. . . . .	542	COMPUTE . . . . .	698
RESTART DATABASE . . . . .	544	CONNECT . . . . .	700
RESTORE DATABASE . . . . .	547	COPY . . . . .	702
REWIND TAPE . . . . .	569	DEFINE . . . . .	704
ROLLFORWARD DATABASE . . . . .	570	DEL . . . . .	705
RUNCMD . . . . .	581	DESCRIBE . . . . .	707
RUNSTATS . . . . .	582	DISCONNECT . . . . .	712
SET CLIENT . . . . .	598	EDIT . . . . .	713
SET RUNTIME DEGREE . . . . .	601	EXECUTE . . . . .	714
SET SERVEROUTPUT . . . . .	603	EXIT . . . . .	717
SET TABLESPACE CONTAINERS . . . . .	605	EXPLAIN PLAN . . . . .	719
SET TAPE POSITION . . . . .	608	GET . . . . .	720
SET UTIL_IMPACT_PRIORITY . . . . .	609	HELP . . . . .	721
SET WORKLOAD . . . . .	611	HOST . . . . .	722
SET WRITE . . . . .	612	IMPORT CLPPlus command . . . . .	723

INPUT	725
LIST	726
PAUSE	728
PRINT	729
PROMPT	730
QUIT	731
REMARK	732
REORGCHK	733
REPFOOTER	735
REPHEADER	736
RUN	738
SAVE	739
SET	740
SPOOL	748
SHOW	749
START	750
TTITLE	751
UNDEFINE	753
WHENEVER OSERROR	754
WHENEVER SQLERROR	756

## Chapter 7. System commands . . . . 759

dasauto - Autostart DB2 administration server	760
dasCRT - Create a DB2 administration server	761
dasdrop - Remove a DB2 administration server	762
dasmigr - Migrate the DB2 administration server	763
dasupdt - Update DAS	765
db2_deinstall - Uninstall DB2 database products, features, or languages	767
db2_install - Install DB2 database product	770
db2_local_ps - DB2 process status for Linux/UNIX	773
db2acsutil - Manage DB2 snapshot backup objects	775
db2addicons - Create main menu entries for DB2 tools	779
db2admin - DB2 administration server	780
db2adutl - Managing DB2 objects within TSM	782
db2advis - DB2 Design Advisor	790
db2audit - Audit facility administrator tool	797
db2batch - Benchmark tool	806
db2bfd - Bind file description tool	816
db2caem - Capture activity event monitor data tool	817
db2cap - CLI/ODBC static package binding tool	821
db2cat - System catalog analysis	823
db2cfexp - Connectivity configuration export tool	826
db2cfimp - Connectivity configuration import tool	828
db2chglbpath - Modify the embedded runtime library search path	829
db2chgpath - Change embedded runtime path	831
db2ckbkp - Check backup	833
db2cklog - Check archive log file validity	837
db2ckrst - Check incremental restore image sequence	839
db2ckupgrade - Check database for upgrade	841
db2cli - DB2 interactive CLI	844
db2cmd - Open DB2 command window	857
db2cpts - Install or update DB2 HA scripts	858
db2dart - Database analysis and reporting tool	859
db2daslevel - Show DAS level	867
db2dclgn - Declaration generator	868
db2diag - db2diag logs analysis tool	871
db2drdat - DRDA trace	887

db2drvmp - Map DB2 database drive	889
db2empfa - Enable multi-page file allocation	891
db2envar.bat - Set environment of the current command window	892
db2evmon - Event monitor productivity tool	893
db2evtbl - Generate event monitor target table definitions	895
db2exfmt - Explain table format	897
db2exmig - Migrate explain tables	900
db2expln - SQL and XQuery Explain	901
db2extsec - Set permissions for DB2 objects	910
db2flsn - Find log sequence number	912
db2fm - DB2 fault monitor	917
db2fmcu - DB2 fault monitor controller	919
db2fodc - DB2 first occurrence data collection	921
db2fopt - Specify query optimizer parameters	932
db2fs - First Steps	934
db2gcf - Control DB2 instance	935
db2gov - DB2 governor	938
db2govlg - DB2 governor log query	940
db2gpm - Get distribution map	941
db2iauto - Autostart instance	942
db2iclus - Microsoft Cluster Server	943
db2icrt - Create instance	946
db2idrop - Remove instance	955
db2IdentifyType1 - Type-1 index identification tool	958
db2ilist - List instances	961
db2inidb - Initialize a mirrored database	962
db2inspf - Format inspect results	965
db2iprune - Reduce installation image size	966
db2isetup - Start instance creation interface	968
db2iupdt - Update instances	970
db2iupgrade - Upgrade instance	978
db2jdbcbind - DB2 JDBC package binder	981
db2ldcfg - Configure LDAP environment	983
db2level - Show DB2 service level	984
db2licm - License management tool	985
db2listvolumes - Display GUIDs for all disk volumes	988
db2logsforrwd - List logs required for rollforward recovery	989
db2look - DB2 statistics and DDL extraction tool	990
db2ls - List installed DB2 products and features	1002
db2move - Database movement tool	1005
db2mqdsn - MQListener	1014
db2mscs - Set up Windows failover utility	1018
db2mtrk - Memory tracker	1022
db2nchg - Change database partition server configuration	1027
db2ncrt - Add database partition server to an instance	1029
db2ndrop - Drop database partition server from an instance	1031
db2nrcfg - Non-root install configuration tool	1032
db2nrupdt - Non-root-installed instance update	1033
db2nrupgrade - Upgrade non-root instance	1034
db2osconf - Utility for kernel parameter values	1035
db2pd - Monitor and troubleshoot DB2 database	1038
db2pdcfg - Configure DB2 database for problem determination behavior	1130
db2perfc - Reset database performance values	1136

db2perfi - Performance counters registration utility . . . . .	1138
db2perfr - Performance monitor registration tool . . . . .	1139
db2rbind - Rebind all packages . . . . .	1140
db2relocatedb - Relocate database . . . . .	1142
db2rfe - Enable root features for non-root installations . . . . .	1148
db2rfpen - Reset rollforward pending state . . . . .	1150
db2rmicons - Remove DB2 tools from main menu . . . . .	1151
db2rspgn - Response file generator . . . . .	1152
db2sampl - Create sample database. . . . .	1153
db2schex - Active Directory schema extension . . . . .	1156
db2set - DB2 profile registry . . . . .	1158
db2setup - Install DB2 database products. . . . .	1163
db2snapcore - DB2 snapcore for Linux and Solaris . . . . .	1165
db2start - Start DB2 . . . . .	1167
db2stat - DB2 process status for Windows . . . . .	1168
db2stop - Stop DB2 . . . . .	1169
db2support - Problem analysis and environment collection tool . . . . .	1170
db2swtch - Switch default DB2 copy and database client interface copy . . . . .	1188
db2sync - Start DB2 synchronizer . . . . .	1190
db2systray - Start DB2 system tray . . . . .	1191
db2tapemgr - Manage log files on tape . . . . .	1193
db2tbst - Get table space state . . . . .	1196
db2tdbmgr - Migrate tools catalog database . . . . .	1197
db2top - DB2 monitoring tool . . . . .	1198
db2trc - Trace . . . . .	1203
db2trcoff - Trace OFF options for db2trc . . . . .	1218
db2trcon - Trace ON options for db2trc . . . . .	1219
db2unins - Uninstall DB2 database products, features, or languages . . . . .	1221
db2untag - Release container tag . . . . .	1223
db2updserv - Show product updates . . . . .	1224
db2val - DB2 copy validation tool . . . . .	1225
db2xdbmig - Migrate XSR objects . . . . .	1227
db2xpvt - Format trap file . . . . .	1228
disable_MQFunctions - Disable WebSphere MQ functions . . . . .	1229
doce_deinstall - Uninstall DB2 Information Center . . . . .	1230
doce_install - Install DB2 Information Center . . . . .	1231
enable_MQFunctions - Enable WebSphere MQ functions . . . . .	1233
installDSDriver - Extract Data Server Driver components . . . . .	1235
installFixPack - Update installed DB2 database products . . . . .	1236

setup - Install DB2 database products . . . . .	1241
---	------

## **Chapter 8. DB2 Text Search commands . . . . . 1243**

db2ts ALTER INDEX . . . . .	1244
db2ts CLEANUP FOR TEXT . . . . .	1249
db2ts CLEAR COMMAND LOCKS. . . . .	1250
db2ts CLEAR EVENTS FOR TEXT . . . . .	1252
db2ts CREATE INDEX . . . . .	1254
db2ts DISABLE DATABASE FOR TEXT . . . . .	1261
db2ts DROP INDEX. . . . .	1263
db2ts ENABLE DATABASE FOR TEXT . . . . .	1265
db2ts HELP . . . . .	1267
db2ts START FOR TEXT . . . . .	1269
db2ts STOP FOR TEXT. . . . .	1271
db2ts UPDATE INDEX. . . . .	1272

## **Appendix A. Naming conventions 1275**

Naming conventions . . . . .	1276
------------------------------	------

## **Appendix B. File type modifiers and delimiters . . . . . 1277**

File type modifiers for the export, import and load utilities . . . . .	1278
Delimiter considerations for moving data . . . . .	1279

## **Appendix C. Overview of the DB2 technical information . . . . . 1281**

DB2 technical library in hardcopy or PDF format . . . . .	1282
Displaying SQL state help from the command line processor . . . . .	1284
Accessing different versions of the DB2 Information Center . . . . .	1284
Updating the DB2 Information Center installed on your computer or intranet server . . . . .	1285
Manually updating the DB2 Information Center installed on your computer or intranet server . . . . .	1286
DB2 tutorials . . . . .	1288
DB2 troubleshooting information . . . . .	1288
Terms and conditions . . . . .	1288

## **Appendix D. Notices . . . . . 1291**

## **Index . . . . . 1295**



---

## 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, system, and DB2 Text Search commands.

The major subject areas discussed in the chapters of this book are as follows:

### Using the command line processor (CLP)

- Chapter 1, “Command line processor (CLP),” explains how to invoke and use the command line processor, and describes the CLP options.
- Chapter 2, “Using command line SQL statements and XQuery statements,” provides information about how to use SQL statements from the command line.

### Using the command line processor plus (CLPPlus)

- Chapter 3, “Command line processor plus (CLPPlus),” describes how to install, invoke, and use the command line processor plus interface.

### Command usage help

- Chapter 4, “How to read command syntax help,” describes how to invoke command help screens and explains the command help screen syntax conventions that are employed.

### CLP commands

- Chapter 5, “CLP commands,” describes all of the database manager commands, listed alphabetically.

### CLPPlus commands

- Chapter 6, “CLPPlus commands,” alphabetically lists and describes all of the commands that can be used in the CLPPlus interface.

### System commands

- Chapter 7, “System commands,” describes all of the commands, listed alphabetically, that can be entered at an operating system command prompt or in a shell script to access the database manager.

### DB2 Text Search commands

- Chapter 8, “DB2 Text Search commands,” describes all of the Text Search commands, listed alphabetically, that can be entered at an operating system command prompt prefixed with **db2ts**.

### Appendixes

- Appendix A, “Naming conventions,” describes the conventions used to name objects such as databases and tables.

## How this book is structured

- Appendix B, “File type modifiers and delimiters,” describes the file type modifiers for the load, import and export utilities. In addition, delimiter considerations for moving data is also presented.

---

## Highlighting conventions

The following highlighting conventions are used in this book.

---

<b>Bold</b>	Indicates commands, keywords, and other items whose names are predefined by the system. Commands written in uppercase are CLP commands, whereas commands written in lowercase are system commands.
<i>Italics</i>	Indicates one of the following items: <ul style="list-style-type: none"><li>• Names or values (variables) that must be supplied by the user</li><li>• General emphasis</li><li>• The introduction of a new term</li><li>• A reference to another source of information</li></ul>
Monospace	Indicates one of the following items: <ul style="list-style-type: none"><li>• Files and directories</li><li>• Information that you are instructed to type at a command prompt or in a window</li><li>• Examples of specific data values</li><li>• Examples of text similar to what might be displayed by the system</li><li>• Examples of system messages</li><li>• Samples of programming code</li></ul>

---

---

## How to read the syntax diagrams

This topic describes the structure of SQL syntax diagrams.

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.

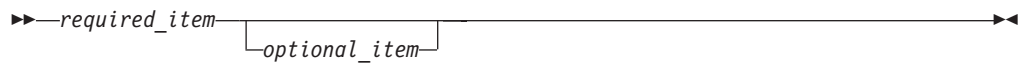
The — $\blacktriangleleft$  symbol indicates the end of a syntax diagram.

Syntax fragments start with the |— symbol and end with the —| symbol.

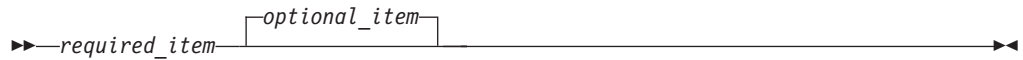
Required items appear on the horizontal line (the main path).



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.

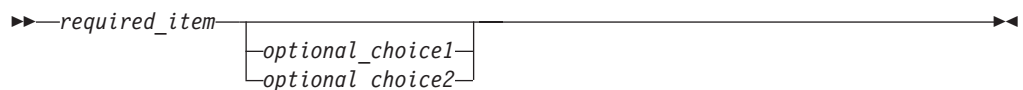


If you can choose from two or more items, they appear in a stack.

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

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

---

## Chapter 1. Command line processor (CLP)

### Command line processor features

This section provides information about the 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.
- When the CLP is called with a file input option, it will automatically set the CLIENT APPLNAME special register to CLP *filename*.

You can start the command line processor by either:

- typing the **db2** command, or,
- on Linux operating systems, click **Main Menu** and, select **IBM DB2 > Command Line Processor**.

Certain CLP commands and SQL statements require that the server instance is running and a database connection exists. Connect to a database by performing one of the following actions:

- 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 1. DB2\_CLPPROMPT tokens and runtime values

DB2_CLPPROMPT token	Value at run time
%ia	Authorization ID of the current instance attachment



Table 1. *DB2\_CLPPROMPT* tokens and runtime values (continued)

DB2_CLPPROMPT token	Value at run time
%i	Local alias of the currently attached instance. If no instance attachment exists, the value of the <b>DB2INSTANCE</b> registry variable. On Windows platforms only, if the <b>DB2INSTANCE</b> registry variable is not set, the value of the <b>DB2INSTDEF</b> registry variable.
%da	Authorization ID of the current database connection
%d	Local alias of the currently connected database. If no database connection exists, the value of the <b>DB2DBDFT</b> registry variable.
%n	New line

- If any token has no associated value at runtime, the empty string is substituted for that token.
- The interactive input prompt will always present the authorization IDs, database names, and instance names in uppercase, 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.

You can specify the number of commands to be stored in the command history by using the **DB2\_CLPHISTSZ** registry variable. The **HISTORY** command lets you access the contents of the command history that you run within a CLP interactive mode session.

You can also specify the editor that is opened when you issue the **EDIT** command by using the **DB2\_CLP\_EDITOR** registry variable. From a CLP interactive session, the **EDIT** command opens an editor preloaded with a user-specified command which can then be edited and run.

### 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 "keon14". **DB2INSTANCE** set to DB2. **DB2DBDFT** set to "SAMPLE".  
(KEON14@DB2, @SAMPLE)
- Database connection to database "sample" with authorization ID "horton7". **DB2INSTANCE** set to DB2. **DB2DBDFT** set to SAMPLE.  
(@DB2, HORTON7@SAMPLE)

## Command line processor features

- Instance attachment to instance "DB2" with authorization ID "keon14". Database connection to database "sample" with authorization ID "horton7". **DB2INSTANCE** set to DB2. **DB2DBDFT** not set.  
(KEON14@DB2, HORTON7@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',
- . test01 (where test01 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:

- test01
- . test01 &
- test01 &
- sh test01

### 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 2. Environment Variables

Variable	Minimum	Maximum	Default
<b>DB2BQTIME</b>	1 second	5294967295	1 second
<b>DB2BQTRY</b>	0 tries	5294967295	60 tries
<b>DB2RQTIME</b>	1 second	5294967295	5 seconds
<b>DB2IQTIME</b>	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.

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**.

## Command line processor features

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

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

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

The back-end process started by user A is still active when user B starts using the CLP, because the parent of user B's front-end process (the operating system window from which the commands are issued) is still active. The back-end process attempts to service the new commands issued by user B; however, user B's front-end process does not have enough authority to use the message queues of the back-end process, because it needs the authority of user A, who created that back-end process. A CLP session must end with a **TERMINATE** command before a user starts a new CLP session using the same operating system window. This creates a fresh back-end process for each new user, preventing authority problems, and setting the correct values of environment variables (such as **DB2INSTANCE**) in the new user's back-end process.

### CLP usage notes

Commands can be entered either in uppercase or in lowercase from the command prompt. However, parameters that are case sensitive to DB2 must be entered in the exact case required. 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 previous 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 staflist.txt in the mydata directory:

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

```
db2 -z mydata\staflist.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.

The command line processor does not support national language support (NLS) characters in file path names. This particularly affects commands such as **IMPORT**, **EXPORT**, and **REGISTER XMLSCHEMA**, where problematic file path names would most frequently be encountered.

Piping the output of a command line processor command into another command line processor command is supported. For example: **db2 -x <SQL\_statement> | db2 +p -tv**. This support is limited only by the pipe buffer size. Pipe buffer sizes are not configurable. If the pipe buffer size is exceeded by the first command, the command line processor might hang or fail when attempting to write the output to the pipe buffer. If the second command is not a command line processor command, for example a UNIX shell command, a hang or failure will not occur due to the pipe buffer size limitation.

### 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:

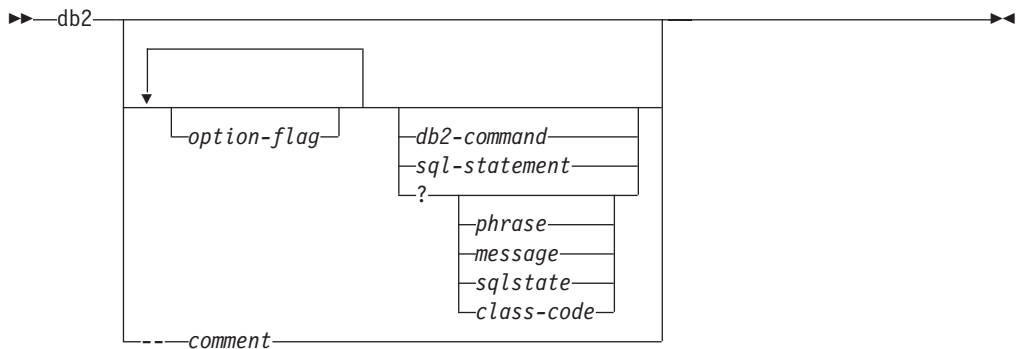
- Interactive input mode, characterized by the db2 => input prompt
- Command mode, where each command must be prefixed by **db2**
- Batch mode, which uses the **-f** file input option.

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 before 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 Linux and UNIX operating 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.

#### **? 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.

## db2 - Command line processor invocation

**? 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.

### 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 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 autocommit 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
-f <i>filename</i>	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
-l <i>filename</i>	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, and MERGE statements.	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



Table 3. CLP Command Options (continued)

Option Flag	Description	Default Setting
-o	This option tells the command line processor to display output data and messages to standard output.	ON
-p	This option tells the command line processor to display a command line processor prompt when in interactive input mode.	ON
-q	This option tells the command line processor to preserve whitespaces and linefeeds in strings delimited with single or double quotation marks. When option q is ON, option n is ignored.	OFF
-r <i>filename</i>	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 and SELECT warning messages.	ON
-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
-z <i>filename</i>	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

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-).

## Command line processor options

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

### Autocommit 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 statements 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 autocommit 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 autocommit 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 autocommit 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 the CLP is called with a file input option, it will automatically set the CLIENT APPLNAME special register to CLP *filename*.

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

```
db2 -tvf filename
```

When you run a CLP script file using the `db2 -tvf filename` command, it sets the CLIENT APPLNAME special register to CLP *filename*. The next command you run, resets the CLIENT APPLNAME and CLIENT ACCTNG special registers to the old value before the `db2 -tvf filename` command was issued. If the next command you run is `db2 terminate` or the last command in *filename* is **TERMINATE**, then the special registers are not reset. This is useful for monitoring which batch job is currently running and differentiating the CLP workload.

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.

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

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

Note that the default termination character is one of the new line characters unless otherwise specified with the `-t` option or the end-of-file.

### 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.

## Command line processor options

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

The `-l filename` option tells the command line processor to log commands to a specified file. This history file contains records of the commands executed and their completion status. *Filename* is an absolute or relative file name which 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 statements.

If set ON (`-m`), the number of rows affected will be displayed for the INSERT, DELETE, UPDATE, or MERGE statements. 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:

- `-r filename`: Interactive mode start-up information is not saved.
- `-e`: SQLCODE or SQLSTATE is displayed on standard output even if `+o` is specified.
- `-a`: No effect if `+o` is specified. If `-a`, `+o` and `-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 `-f filename` 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 quotation marks.

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 `-r filename` option causes any output data generated by a command to be written to a specified file, and is useful for capturing a report that would otherwise scroll off the screen. Messages or error codes are not written to the file. *Filename* is an absolute or relative file name which 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 (+r or -r-)`.

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

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

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

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

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

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

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

The following table summarizes this behavior:

## Command line processor options

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 Options (-t and -tdx or -tdxx):

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-).

**Note:** If you use the CLP to issue XQuery statements, it is best to choose a termination character other than the semicolon. This ensures that statements or queries that use namespace declarations are not misinterpreted, since namespace declarations are also terminated by a semicolon.

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 in an input file to set the statement termination characters. For example:

```
db2 -td%% -f file1.txt
```

or

```
db2 -f file2.txt
```

where file2.txt contains the following as the first statement in the file:

```
--#SET TERMINATOR %%
```

The default setting for this command option is OFF.

The termination character or characters cannot be used to concatenate multiple statements from the command line, since checks for a termination symbol are performed on only the last one or two non-blank characters of each input line.

The statement termination character options do 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 before 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 +0 (or -0-) 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 or SELECT statement). 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 -z *filename* option causes all output generated by a command to be written to a specified file, and is useful for capturing a report that would otherwise scroll off the screen. It is similar to the -r option; in this case, however, messages, error codes, and other informational output are also written to the file. *Filename* is an absolute or relative file name which 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 -z *filename* is set in **DB2OPTIONS**, the user can set the +z (or -z-) option from the command line to prevent output data for a particular command invocation from being written to the file.

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

### Command line processor return codes

This section provides information about the 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 values:

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.

---

### Invoking command help from the command line processor

Command help explains the syntax of commands in the command line processor.



## Invoking command help from 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.

---

## 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.

## Invoking message help from the command line processor

---

## Chapter 2. 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.

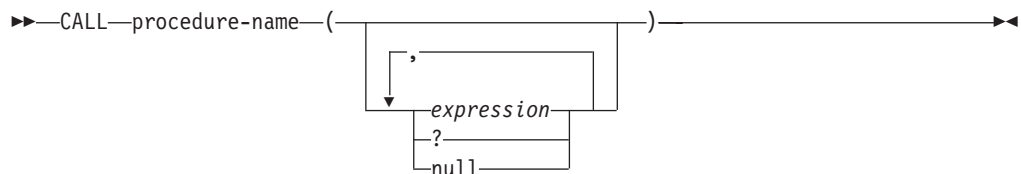
You can use SQL statements from the command line, and you can use a stored procedure (SYSPROC.ADMIN\_CMD()) to run some CLP commands through SQL. For more information about how to use this stored procedure, refer to the SQL Administrative Routines.

To issue XQuery statements in CLP, prefix the statements with the XQUERY keyword.

**Note:** If you use the CLP to issue XQuery statements, it is best to choose a termination character other than the semicolon (-t option). This ensures that statements or queries that use namespace declarations are not misinterpreted, since namespace declarations are also terminated by a semicolon.

All SQL statements that can be executed through the command line processor are listed in the CLP column of Table 5 on page 25. 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 in the following section:

### CALL

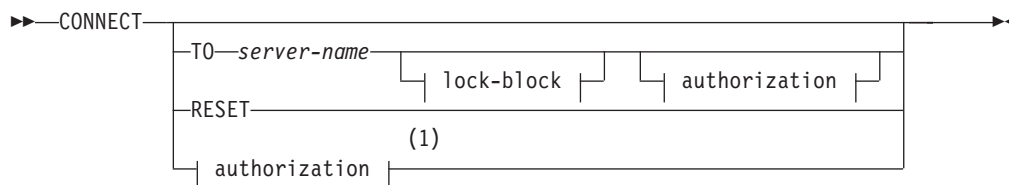


### CLOSE

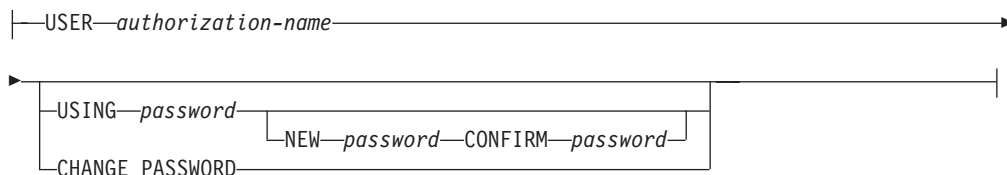


### CONNECT

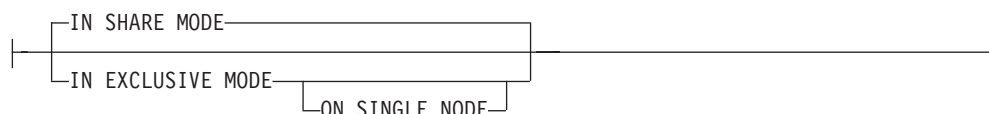
## Using command line SQL statements and XQuery statements



### authorization:



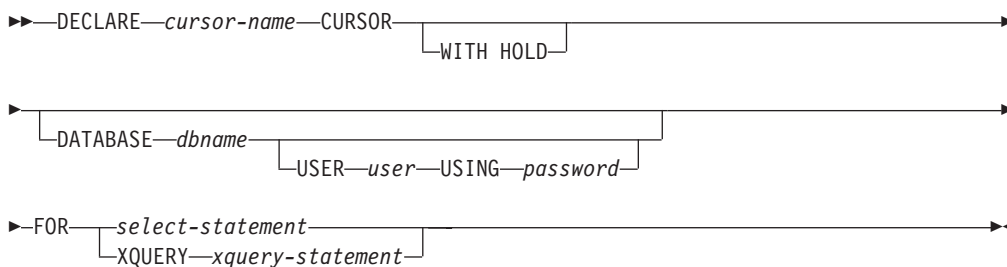
### lock-block:



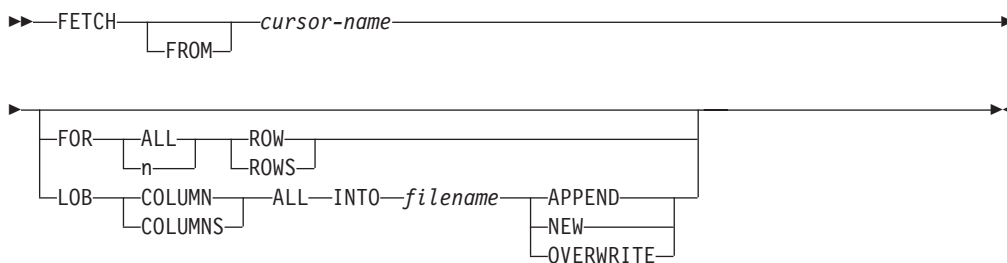
### Notes:

- 1 This form is only valid if implicit connect is enabled.

## DECLARE CURSOR



## FETCH



## OPEN

►►—OPEN—*cursor-name*—►►

### Note:

#### 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 *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

## Using command line SQL statements and XQuery statements

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 other countries/regions. 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/regions 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:

## Using command line SQL statements and XQuery statements

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
```

2. Connect to the database.
3. Issue the following command:  

```
db2 bind @clp.lst collection nullid datetime iso
```

Table 5. SQL Statements (DB2)

SQL Statement	Dynamic <sup>1</sup>	Command Line Processor (CLP)	Call Level Interface <sup>3</sup> (CLI)	SQL Procedure
ALLOCATE CURSOR				X
assignment statement				X
ASSOCIATE LOCATORS				X
ALTER { BUFFERPOOL, DATABASE PARTITION GROUP, NICKNAME, <sup>9</sup> SERVER, <sup>9</sup> TABLE, TABLESPACE, USER MAPPING, <sup>9</sup> TYPE, VIEW }	X	X	X	
BEGIN DECLARE SECTION <sup>2</sup>				
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 <sup>4</sup>	
compound statement				X
CONNECT (Type 1)		X	SQLBrowseConnect(), SQLConnect(), SQLDriverConnect()	
CONNECT (Type 2)		X	SQLBrowseConnect(), SQLConnect(), SQLDriverConnect()	
CREATE { ALIAS, BUFFERPOOL, DATABASE PARTITION GROUP, DISTINCT TYPE, EVENT MONITOR, FUNCTION, FUNCTION MAPPING <sup>9</sup> , GLOBAL TEMPORARY TABLE, INDEX, INDEX EXTENSION, METHOD, NICKNAME, <sup>9</sup> PROCEDURE, SCHEMA, SERVER, TABLE, TABLESPACE, TRANSFORM, TYPE MAPPING, <sup>9</sup> TRIGGER, USER MAPPING, <sup>9</sup> TYPE, VIEW, WRAPPER <sup>9</sup> }	X	X	X	X <sup>10</sup>
DECLARE CURSOR <sup>2</sup>		X	SQLAllocStmt()	X
DECLARE GLOBAL TEMPORARY TABLE	X	X	X	X

## Using command line SQL statements and XQuery statements

Table 5. SQL Statements (DB2) (continued)

SQL Statement	Dynamic <sup>1</sup>	Command Line Processor (CLP)	Call Level Interface <sup>3</sup> (CLI)	SQL Procedure
DELETE	X	X	X	X
DESCRIBE <sup>8</sup>		X	SQLColAttributes(), SQLDescribeCol(), SQLDescribeParam() <sup>6</sup>	
DISCONNECT		X	SQLDisconnect()	
DROP	X	X	X	X <sup>10</sup>
END DECLARE SECTION <sup>2</sup>				
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 <sup>4</sup>	X
GET DIAGNOSTICS				X
GOTO statement				X
GRANT	X	X	X	X
IF statement				X
INCLUDE <sup>2</sup>				
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



## Using command line SQL statements and XQuery statements

Table 5. SQL Statements (DB2) (continued)

SQL Statement	Dynamic <sup>1</sup>	Command Line Processor (CLP)	Call Level Interface <sup>3</sup> (CLI)	SQL Procedure
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
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 <sup>9</sup>	X	X	X	X
SET PATH	X	X	X	X
SET SCHEMA	X	X	X	X
SET SERVER OPTION <sup>9</sup>	X	X	X	X
SET transition-variable <sup>5</sup>	X	X	X	X
SIGNAL statement				X
SIGNAL SQLSTATE <sup>5</sup>	X	X	X	
UPDATE	X	X	X	X
VALUES INTO				X
WHENEVER <sup>2</sup>				
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 `SQLExecuteDirect()` or `SQLPrepare()` and `SQLExecute()`. If there is an equivalent CLI function, the function name is listed.
4. Although this statement is not dynamic, with CLI you can specify this statement when calling either `SQLExecuteDirect()`, 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 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 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.

## Using command line SQL statements and XQuery statements

---

## Chapter 3. Command line processor plus (CLPPlus)

Command line processor plus (CLPPlus) provides a command-line user interface that you can use to connect to databases and to define, edit, and run statements, scripts, and commands.

CLPPlus complements the functions that the command line processor (CLP) provides. CLPPlus includes the following features:

- Support for establishing connections to databases when you provide a database user ID and password.
- A buffer that you can use to store scripts, script fragments, SQL statements, SQL PL statements, or PL/SQL statements for editing and then execution. You can list, print, or edit the text in the buffer or run the text in the buffer as a batch script.
- A comprehensive set of processor commands that you can use to define variables and strings that you can store in the buffer.
- A set of commands that retrieve information about a database and database objects.
- The ability to store buffers or buffer output in a file.
- Multiple options for formatting the output of scripts and queries.
- Support for executing system-defined routines.
- Support for executing operating system commands.
- An option for recording the output of executed commands, statements, or scripts.

CLPPlus supports SERVER, SERVER\_ENCRYPT, and KERBEROS authentication only.

---

### Installing CLPPlus on existing IBM data server clients

CLPPlus, is not installed on all IBM data server clients by default. If your IBM data server client installation did not include CLPPlus, you can install and configure it later.

#### Before you begin

Ensure that an SDK or a JRE for Java, Version 1.5.0 or later, is installed. For all IBM data server clients except IBM Data Server Runtime Client and IBM Data Server Driver Package, the IBM data server client installation process automatically installs the SDK for Java, Version 5.

#### Procedure

To install and configure CLPPlus on an existing IBM data server client:

1. Ensure that the **JAVA\_HOME** environment variable is set to the Java installation directory on your system.
2. Ensure that the **PATH** environment variable setting includes the *Java installation directory/bin* directory on your system.

## Installing CLPPlus on existing IBM data server clients

3. Add the `clpplus.jar`, `jline-0.9.93.jar` and `db2jcc.jar` files to the **CLASSPATH** environment variable setting. If CLPPlus fails to find these files, it displays error messages for the files and does not start.

---

## Starting CLPPlus

CLPPlus must be started before any CLPPlus commands can be run.

### Before you begin

- CLPPlus might not be installed and set up on an existing IBM data server client. CLPPlus is set up for use with all other DB2 V9.7 products.
- Read the **CLPPLUS** command topic.

### Procedure

To start CLPPlus, use one of the following:

- On Windows, Linux, or UNIX operating systems, use a command:
  1. In the command line processor (CLP) or DB2 Command Window, run the **CLPPLUS** command without specifying a password. For example, to start CLPPlus and attempt to connect to a database named SAMPLE on the local host by using user ID db2admin and port number 50000, issue the following command:

```
clpplus db2admin@localhost:50000/sample
```
  2. Enter your password. Using the command in 1) as an example, if a valid password for db2admin is entered, the following is displayed:

```
C:\DB2\9.7\db2 > clpplus db2admin@localhost:50000/sample
Connected to SAMPLE v 9.7 AS db2admin
SQL>
```
- On Windows operating systems, use a menu option:
  1. Click **Start > IBM DB2 > CLPPlus**.
  2. Specify your user ID and connection information.
- On Linux operating systems, use a menu option:
  1. Click **Main Menu > IBM DB2 > Command Line Processor Plus**.
  2. Specify your user ID and connection information.

### Results

The CLPPlus prompt (SQL>) is available, and you are connected to the specified database.

### What to do next

You can now use CLPPlus commands and related features. Specify the commands at the CLPPlus prompt.

To end the CLPPlus session, issue the **EXIT** or **QUIT** CLPPlus command.

## CLPPLUS command

Starts command line processor plus (CLPPlus). After starting CLPPlus, you can issue CLPPlus commands, connect to databases, define and run SQL statements and database commands, and run scripts that contain SQL statements and commands.

### Invocation

You must run this command from the operating system command prompt.

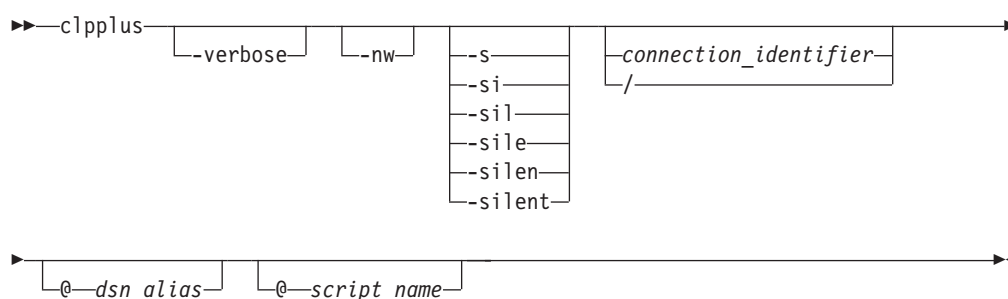
### Authorization

None

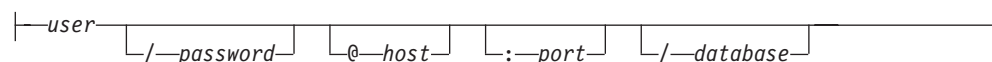
### Required connection

None

### Command Syntax



#### connection\_identifier:



### Command parameters

#### -verbose

Sets verbose on. When verbose is set on all CLPPlus messages are printed to the console.

#### -nw

Specifies that the CLPPlus session is started in the current command-line window.

#### -s | -si | -sil | -sile | -silen | -silent

Suppresses the version information, copyright information, prompt messages, command echo and connection information from being printed in the current CLPPlus session. During silent mode, by default, ECHO is set to OFF. Any attempt to set ECHO to ON is ignored.

/ Specifies the current operating system login user ID is used to connect to the database.

## CLPPLUS command

### *user*

Specifies the user ID to connect to the database.

### *password*

Specifies the password that corresponds to the user ID.

### *hostname*

Specifies the name of the computer on which the database is located. For example, for a computer named ascender, specify @ascender.

### *port*

Specifies the port number receiving connections on the computer where the database server is installed. The default is 50000.

### *database*

Specifies the database name to which the connection is made. The default is SAMPLE.

### *dsn\_alias*

Specifies that the database connection information is picked up from the db2dsdriver.cfg file from the dsn with alias name *dsn\_alias*. If the specified *dsn\_alias* is not found in the db2dsdriver.cfg file, the string *dsn\_alias* is used as a database name and all other connection parameters are obtained interactively.

### *script-name*

Specifies the name of a script file. If the file is not in the current working directory, you must also include the fully qualified path to the location of the file. The script file can contain SQL statements that are automatically run after you start CLPPlus and establish a database connection.

## Example

The following examples show how you can use the **CLPPLUS** command to start the CLPPlus processor and optionally connect to databases.

The following command starts CLPPlus in client mode:

```
clpplus
```

No database connection is attempted. After issuing this command, to connect to a database, use the **CONNECT** CLPPlus command.

The following command starts CLPPlus and attempts to connect to a database named SAMPLE on a local host with user ID db2admin and port number 50000:

```
clpplus db2admin@localhost:50000/sample
```

The following command starts CLPPlus and prompts for a password for the user ID db2admin. If the password is valid, CLPPlus attempts to connect to a database named SAMPLE, which is the default database name.

```
clpplus db2admin
```

The following command starts CLPPlus and attempts to connect to a database named SAMPLE with user ID db2admin and password mypassw0rd. If the user ID and password are valid, a database connection is established. The drawback to specifying a password is that the password is displayed on the screen.

```
clpplus db2admin/mypassw0rd
```

The following command starts CLPPlus and attempts to connect to a database named SAMPLE on the remote machine ascender using port 50000, user ID dbadmin, and password mypassw0rd. If these values are valid, a database connection is established.

```
clpplus db2admin/mypassw0rd@ascender:50000/sample
```

The following command starts CLPPlus and attempts to connect to a database named DB on the local computer with user ID db2admin, password mypassw0rd, and the default port number, which is 50000:

```
clpplus db2admin/mypassw0rd@localhost/db
```

The following command starts CLPPlus and attempts to connect to a database by first locating a db2dsdriver.cfg file. If one is found, the default\_dsn is read for the host, port, and database values. The current log on ID is used in the connection attempt. If no db2dsdriver.cfg file is found, all required parameters are requested interactively.

```
clpplus /
```

The following command starts CLPPlus and attempts to connect to a database by first locating a db2dsdriver.cfg file. If one is found, the default\_dsn is read for the host, port, and database values. The db2admin ID is used in the connection attempt. If no db2dsdriver.cfg file is found, all required parameters are requested interactively.

```
clpplus db2admin
```

The following command starts CLPPlus and attempts to connect to a database by fetching the parameters from the data\_dsn alias in the db2dsdriver.cfg file. Since no user ID is specified, the current log on user ID is used. Any parameters that cannot be read are requested interactively.

```
clpplus /@data_dsn
```

The following command starts CLPPlus and attempts to connect to a database by fetching the parameters from the data\_dsn alias in the db2dsdriver.cfg file. The db2admin user ID is used in the connection. Any parameters that cannot be read are requested interactively.

```
clpplus db2admin@data_dsn
```

---

## CLPPlus console types

As of V9.7 Fix Pack 2, CLPPlus adds a new window mode to the traditional command line console.

### Window mode

The new console introduced in V9.7 Fix Pack 2 is a window styled console. When CLPPlus is started with the **CLPPLUS** command, the window mode console is spawned by default. This new console window has better command editing capabilities. As such, when using CLPPlus in an interactive mode, this window mode might be the preferred console choice.

### Non-window mode

CLPPlus still includes the option of using the non-window mode or traditional command line console. When CLPPlus is started with the **CLPPLUS** command and the **-nw** option is specified, the current command line environment is used, that is,

## CLPPlus console types

no new window is spawned. This might be a preferred choice if you are calling a script file and do not require a spawned console window.



---

## Window mode and UTF-8 character support

As of V9.7 Fix Pack 3, the new window mode console supports only UTF-8 characters.

Before the introduction of the new window mode console, CLPPlus was not able to effectively read or write UTF-8 characters in the console from the file system or database. UTF-8 characters displayed on the screen were misaligned or incorrect. Users could not enter UTF-8 characters from the keyboard.

The new window mode console addresses all of these issues with UTF-8 characters while still providing a console look. CLPPlus can now read and write UTF-8 characters from the file system and database, or interactively from the keyboard or user interface without issue. Column alignment issues are also resolved with the new window mode console. Thus, table output is now properly displayed.

### Note:

This support is only available in window mode, which is the default console as of V9.7 Fix Pack 2. For non-window mode, support for UTF-8 characters is still limited. In non-window mode, you might not be able to enter certain UTF-8 characters from the keyboard. Misalignment of result set columns can occur if UTF-8 characters are included in that result set.

---

## DSN aliases in CLPPlus

CLPPlus supports connecting to DSN aliases defined in the `db2dsdriver.cfg` configuration file. Prior to this support, only interactive connects were allowed in the CLPPlus interface.

The `ds` driver configuration file, `db2dsdriver.cfg`, is an XML file that contains a list of DSN aliases and their properties. It is used to store connection details in one place. CLPPlus can use that information to automatically connect to a data source instead of interactively asking for all the connection details on every connect attempt.

You can set the `DB2DSDRIVER_CFG_PATH` environment variable to point to the `db2dsdriver.cfg` file.

If `DB2DSDRIVER_CFG_PATH` is not set, CLPPlus will search for the configuration file automatically in `install-directory/cfg/`.

If a configuration file is found and it is readable, CLPPlus will start using it on the subsequent connect attempts.

At this point, a user who attempts a connect is asked for a database name. That database name is treated as a DSN alias and searched in the configuration file. If that DSN alias is found, the connection attributes are read and a password is requested to complete the connection. If the DSN alias is not found, then the host name, port number, user name, and password are requested interactively to go with the original database name, and all information gathered is used to attempt a connection.

### Example

Consider the following `db2dsdriver.cfg` file contents:

## DSN aliases in CLPPlus

```
<configuration>
  <dsncollection>
    <dsn alias="S" name="SAMPLE" host="9.121.221.159" port="50000">
    </dsn>
  </dsncollection>
  <databases>
    <database name="SAMPLE" host="9.121.221.159" port="50000">
      <parameter name="UserID" value="john"/>
    </database>
  </databases>
</configuration>
```

This example shows a connection being established using the contents of the db2dsdriver.cfg file.

First the user sets *DB2DSDRIVER\_CFG\_PATH*.

```
C:\>set DB2DSDRIVER_CFG_PATH="C:\john\clpplus\db2dsdriver.cfg"
```

Then the user starts a CLPPlus session and attempts a connection to the DSN alias "S".

```
C:\>clpplus
CLPPlus: Version 1.1
Copyright (c) 2009, IBM CORPORATION. All rights reserved.
```

```
SQL> connect
```

```
Enter DATABASE NAME [SAMPLE]: S
Enter ID [john] :
Enter Password: *****
```

```
Database Connection Information
```

```
-----
Hostname = 9.121.221.159
Database server = DB2/NT SQL09071
SQL authorization ID = john
Local database alias = S
Port = 50000
```

```
SQL>
```

This next example shows a connection being established when the database name entered is not found as an alias in the db2dsdriver.cfg file.

```
SQL> connect
```

```
Enter DATABASE NAME [SAMPLE]:
Enter HOSTNAME [localhost]:
Enter PORT [50000]:
Enter ID : john
Enter Password: *****
```

```
Database Connection Information
```

```
-----
Hostname = 9.121.221.159
Database server = DB2/NT SQL09071
SQL authorization ID = john
Local database alias = SAMPLE
Port = 50000
```

```
SQL>
```

Since "sample" is not found as a DSN alias in the configuration file, the remaining values are requested interactively by CLPPlus and then a connection attempt is made.

## Kerberos authentication in CLPPlus

As of V9.7 Fix Pack 2, CLPPlus supports connecting to DSN aliases using Kerberos authentication as defined in the `db2dsdriver.cfg` configuration file. Prior to this support, only interactive connects were allowed in the CLPPlus interface.

The `db2dsdriver.cfg` configuration file is an XML file that contains a list of DSN aliases and their properties. If the **Authentication** property is read for the DSN alias in question, it will be read, and if the value is *kerberos* the Kerberos authentication mechanism will be attempted. For more details on connecting to DSN aliases in CLPPlus, see: "DSN aliases in CLPPlus" on page 35.

CLPPlus will not request a Kerberos TGT ticket on its own. It will use the ticket already obtained by the user using other applications or tools.

### Example

Consider the following `db2dsdriver.cfg` file contents:

```
<configuration>
<dsncollection>
  <dsn alias="S" name="SAMPLE" host="9.121.221.159" port="50000">
  </dsn>
</dsncollection>
<databases>
  <database name="SAMPLE" host="9.121.221.159" port="50000">
    <parameter name="UserID" value="john"/>
  </database>
</databases>
<parameters>
  <parameter name="Authentication" value="KERBEROS"/>
</parameters>
</configuration>
```

This example shows a connection being established using the contents of the `db2dsdriver.cfg` file which includes reading the **Authentication** parameter value.

The user starts a CLPPlus session and attempts a connection to the DSN alias "S".

```
C:\>clpplus
CLPPlus: Version 1.1
Copyright (c) 2009, IBM CORPORATION. All rights reserved.

SQL> connect

Enter DATABASE NAME [SAMPLE]: S
Enter ID [john] :
Enter Password: *****

Database Connection Information
-----
Hostname = 9.121.221.159
Database server = DB2/NT SQL09071
SQL authorization ID = john
Local database alias = S
Port = 50000

SQL>
```

---

## SERVER\_ENCRYPT authentication in CLPPlus

As of V9.7 Fix Pack 5, CLPPlus adds support to connect DSN aliases with SERVER\_ENCRYPT authentication as defined in the db2dsdriver.cfg configuration file.

The db2dsdriver.cfg configuration file is an XML file that contains a list of DSN aliases and their properties. If the **Authentication** property is read for the DSN alias in question, it is read, and if the value is *SERVER\_ENCRYPT* the SERVER\_ENCRYPT authentication mechanism is used. For more details on connecting to DSN aliases in CLPPlus, see the Related reference.

### Example

This example shows a connection with the contents of the db2dsdriver.cfg file which includes the **Authentication** parameter value *SERVER\_ENCRYPT*.

Consider this example db2dsdriver.cfg file :

```
<configuration>
<dsncollection>
  <dsn alias="S" name="SAMPLE" host="9.121.221.159" port="50000">
    <parameter name="Authentication" value="SERVER_ENCRYPT"/>
  </dsn>
</dsncollection>
<databases>
  <database name="SAMPLE" host="9.121.221.159" port="50000">
    <parameter name="UserID" value="john"/>
  </database>
</databases>
</configuration>
```

The user starts a CLPPlus session and connects to the DSN alias "S".

```
C:\>clpplus -nw
CLPPlus: Version 1.5
Copyright (c) 2009, 2011, IBM CORPORATION. All rights reserved.
```

```
SQL> connect
Enter DATABASE NAME [SAMPLE]: S
Enter ID [john]:
Enter password: *****
```

```
Database Connection Information :
-----
Hostname = 9.121.221.159
Database server = DB2/NT SQL09075
SQL authorization ID = john
Local database alias = S
Port = 50000
```

## LDAP support in CLPPlus

As of V9.7 Fix Pack 5, CLPPlus connections support DSN alias searches in a configured LDAP directory server.

### Description

Before V9.7 Fix Pack 5, CLPPlus searched the `db2dsdriver.cfg` file for the DSN alias that was provided as part of the connection attempt. If a match was found in the `db2dsdriver.cfg` file, information such as the host name and port number was retrieved from the file and used in the connection attempt. If the DSN alias was not found in the `db2dsdriver.cfg` file, the DSN alias name was treated as the database name during an interactive connection attempt where you had to provide information such as the host name and port number.

As of V9.7 Fix Pack 5, CLPPlus extends the DSN alias name lookup to a configured LDAP directory server. If you specify a DSN alias name and it is not found in the `db2dsdriver.cfg` file, CLPPlus attempts to connect to the LDAP directory server by using a URL if you provide one in the configuration file. If CLPPlus successfully connects to the configured LDAP directory server, a lookup is done for the DSN alias name on the server. If a match on the LDAP directory server is found, the required connection details, such as host name, port number, user ID, and password, are used to make a CLPPlus connection. If no match is found on the LDAP directory server or the connection to the LDAP directory server fails, the DSN alias name is treated as a database name during an interactive connection, as done in the past.

To enable the new support, use the `<ldapserver>` tag in the `db2dsdriver.cfg` file to specify the LDAP directory server URL. A single LDAP server entry is allowed in the `db2dsdriver.cfg` file. The `UserID` and `Password` fields in the `<ldapserver>` tag are optional; you can enter user ID and password information at run time. User ID and password information is cached for the duration of the CLPPlus session.

If you set the `UserID` parameter in the `db2dsdriver.cfg` file to `"*anonymous"`, an anonymous connection to the LDAP server is attempted; user ID and password information is not passed. You are not prompted for a password, and if you set the `Password` parameter in the `db2dsdriver.cfg` file, the parameter is ignored.

### Important:

If you set the `UserID` parameter to `"*anonymous"`, you must configure the LDAP server to support anonymous connections.

### Examples

Consider this sample `db2dsdriver.cfg` file:

```
<configuration>
<dsncollection>
  <dsn alias="alias1" name="name1" host="server1.net1.com" port="50001"/>
</dsncollection>

<databases>

  <database name="name1" host="server1.net1.com" port="50001">
    <parameter name="CurrentSchema" value="OWNER1"/>
    <wlb>
      <parameter name="enableWLB" value="true"/>
      <parameter name="maxTransports" value="50"/>
    </wlb>
  </database>
</databases>
```

## LDAP support in CLPPlus

```
<acr>
  <parameter name="enableACR" value="true"/>
</acr>
</database>

</databases>

<ldapsrv>
  <parameter name="EnableLDAP" value="YES"/>
  <parameter name="LDAPServerHost" value="ipv6lab7.torolab.ibm.com"/>
  <parameter name="LDAPServerPort" value="389"/>
  <parameter name="UserID" value="root"/>
  <parameter name="Password" value="itdsv23"/>
  <parameter name="BaseDN" value="O=IBM"/>
  <parameter name="Authentication" value="simple"/>
</ldapsrv>
</configuration>
```

As shown in the following example, using the sample `db2dsdriver.cfg` file, a connection to alias name `DBLDAP1` is attempted. The alias name is not found in the file. However, an LDAP directory server, `ipv6lab7.torolab.ibm`, is specified in the file. CLPPlus successfully connects to that server. The LDAP directory server is successfully searched for `DBLDAP1`, and the host, port, and database information is retrieved. A CLPPlus connection attempt occurs.

```
SQL> connect
Enter DATABASE NAME [SAMPLE]: DBLDAP1
Enter ID : db2admin
Enter password: *****
```

```
Database Connection Information :
-----
Hostname = winguest.torolab.ibm.com
Database server = DB2/NT SQL09075
SQL authorization ID = db2admin
Local database alias = DBLDAP1
Port = 50000
```

The situation in the following example is the same as that in the previous example, but the following output shows the results of running CLPPlus in **VERBOSE** mode:

```
SQL> connect
DB250001I: CLPPlus has successfully read the configuration file named
'C:\Documents and Settings\All Users\Application data\IBM\DB2\DB2COPY1\cfg\db2dsdriver.cfg'.

Enter DATABASE NAME [SAMPLE]: DBLDAP1

DB250004I: DSN alias 'DBLDAP1' is not found in the configuration file named
'C:\Documents and Settings\All Users\Application Data\IBM\DB2\DB2COPY1\cfg\db2dsdriver.cfg'.

DB250005I: CLPPlus successfully established a connection with LDAP directory
server 'ipv6lab7.torolab.ibm.com:389'

Enter ID : db2admin
Enter password: *****

Database Connection Information :
-----
Hostname = winguest.torolab.ibm.com
Database server = DB2/NT SQL09075
SQL authorization ID = db2admin
Local database alias = DBLDAP1
Port = 50000
```

In the following example, using the sample `db2dsdriver.cfg` again, a connection to alias name `DBLDAP2` is attempted. CLPPlus is running in **VERBOSE** mode. `DBLDAP2` is not found in the `db2dsdriver.cfg` file or on the specified LDAP directory server. An interactive CLPPlus connection attempt then occurs.

```

SQL> connect
DB250001I: CLPPlus has successfully read the configuration file named
'C:\Documents and Settings\All Users\Application data\IBM\DB2\DB2COPY1\cfg\db2dsdriver.cfg'.

Enter DATABASE NAME [SAMPLE]: DBLDAP2

DB250014I: DSN alias 'DBLDAP2' is not found in the configuration file named
'C:\Documents and Settings\All Users\Application Data\IBM\DB2\DB2COPY1\cfg\db2dsdriver.cfg'.

DB250015I: CLPPlus successfully established a connection with LDAP directory server 'ipv6lab7.torolab.ibm.com:389'

DB250016E: DSN alias 'DBLDAP2' was not found in LDAP directory server 'ipv6lab7.torolab.ibm.com:389'.
'DBLDAP2' is used as the database name in the subsequent interactive CLPPlus connect attempt.

Enter HOSTNAME [localhost]: 9.128.34.89
Enter PORT [50000]: 50003
Enter ID: db2admin
Enter password:*****

Database Connection Information :
-----
Hostname = 9.128.34.89
Database server = DB2/NT SQL09075
SQL authorization ID = db2admin
Local database alias = DBLDAP2
Port = 50003

```

The situation in the previous example is repeated in the following example, except that a connection to the LDAP directory server cannot be established. CLPPlus is running in **VERBOSE** mode.

```

SQL> connect
DB250001I: CLPPlus has successfully read the configuration file named
'C:\Documents and Settings\All Users\Application data\IBM\DB2\DB2COPY1\cfg\db2dsdriver.cfg'.

Enter DATABASE NAME [SAMPLE]: DBLDAP2

DB250014I: DSN alias 'DBLDAP2' is not found in the configuration file named
'C:\Documents and Settings\All Users\Application Data\IBM\DB2\DB2COPY1\cfg\db2dsdriver.cfg'.

DB250017E: CLPPlus failed to establish a connection with LDAP
directory server 'ipv6lab7.torolab.ibm.com:389'. 'DBLDAP2' is
used as the database name in an interactive connect attempt.

Enter HOSTNAME [localhost]: 9.128.34.89
Enter PORT [50000]: 50003
Enter ID: db2admin
Enter password:*****

Database Connection Information :
-----
Hostname = 9.128.34.89
Database server = DB2/NT SQL09075
SQL authorization ID = db2admin
Local database alias = DBLDAP2
Port = 50003

```

Consider the following modified version of the db2dsdriver.cfg file. The file does not include the **UserID** and **Password** parameters for the LDAP directory server configuration. The LDAP directory server is specified as ipv6lab7.torolab.ibm.

```

<configuration>
  <dsncollection>
    <dsn alias="alias1" name="name1" host="server1.net1.com" port="50001"/>
  </dsncollection>

  <databases>

    <database name="name1" host="server1.net1.com" port="50001">
      <parameter name="CurrentSchema" value="OWNER1"/>
    </wlb>
    <parameter name="enableWLB" value="true"/>
    <parameter name="maxTransports" value="50"/>
  </wlb>

```

## LDAP support in CLPPlus

```
<acr>
  <parameter name="enableACR" value="true"/>
</acr>
</database>

</databases>

<ldapsrv>
  <parameter name="EnableLDAP" value="YES"/>
  <parameter name="LDAPServerHost" value="ipv6lab7.torolab.ibm.com"/>
  <parameter name="LDAPServerPort" value="389"/>
  <parameter name="BaseDN" value="O=IBM"/>
  <parameter name="Authentication" value="simple"/>
</ldapsrv>
</configuration>
```

Using the updated db2dsdriver.cfg file, a connection to alias name SAMPLE32 is attempted. The alias name is not found in the file. When the user ID and password to the LDAP directory server are entered interactively, as shown in the following example, CLPPlus connects to the ipv6lab7.torolab.ibm LDAP directory server. The LDAP directory server is successfully searched for SAMPLE32, and the host, port, and database information is retrieved. A CLPPlus connection attempt occurs. CLPPlus is not running in **VERBOSE** mode in this example.

```
C:\Documents and Settings>clpplus /@SAMPLE32
```

```
CLPPlus: Version 1.4
Copyright (c) 2009, 2011, IBM CORPORATION. All rights reserved.
```

```
Connecting to LDAP server '9.234.67.89:389'.
Enter LDAP server user ID: root
Enter LDAP server password: *****
```

```
Enter password: *****
```

```
Database Connection Information :
-----
Hostname = 9.128.32.149
Database server = DB2/NT SQL09075
SQL authorization ID = db2admin
Local database alias = SAMPLE32
Port = 50002
```



## CREATE DATABASE in CLPPlus

As of V9.7 Fix Pack 5, limited support for the **CREATE DATABASE** command is added to CLPPlus for DB2 Database for Linux, UNIX, and Windows and IBM Informix®.

### Restrictions

For DB2 Database for Linux, UNIX, and Windows, the only valid parameters for the **CREATE DATABASE** command in the CLPPlus interface are as follows:

- The required *database-name* variable
- The optional **CODESET** parameter
- The optional **TERRITORY** parameter

For IBM Informix, **CREATE DATABASE** from the CLPPlus interface requires connection credentials, specifically to the **sysmaster** database. You are prompted for connection credentials if a connection does not exist. For more information about the **CREATE DATABASE** command in IBM Informix, see [http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp?topic=/com.ibm.sqls.doc/ids\\_sq\\_0368.htm](http://publib.boulder.ibm.com/infocenter/idshelp/v115/index.jsp?topic=/com.ibm.sqls.doc/ids_sq_0368.htm)

### Examples

The following command creates a database named testdb on DB2 Database for Linux, UNIX, and Windows:

```
create database testdb;
```

The following command creates a database named testdb on DB2 Database for Linux, UNIX, and Windows. The code page set is defined as UTF-8, and the territory is defined as US.

```
create db testdb using codeset UTF-8 territory US;
```

The following command creates a database named udttest on IBM Informix. No previous connection exists, so the command prompts for information.

```
SQL> create database udttest;
```

```
Enter DATABASE NAME [sysmaster]:
Enter HOSTNAME [localhost]: 9.130.34.100
Enter PORT [50000]: 9089
Enter ID: informix
Enter password: *****
```

```
DB250000I: The command completed successfully.
```

In the following example, the first command connects to an IBM Informix database. The second command creates a database named udttest.

```
SQL> connect Informix/informix123@9.130.34.100:9089/stores
```

```
Database Connection Information :
```

```
-----
Hostname = 9.130.34.100
Database server = IDS/NT32 IFX11700
SQL authorization ID = informix
Local database alias = stores
Port = 9089
```

```
SQL> create database udttest with log mode ansi ;
```

```
DB250000I: The command completed successfully.
```

---

### Running a script file in CLPPlus

In CLPPlus, a script file can be run in many ways. You can provide the name of a script file that contains database commands and SQL commands as a token for the **CLPPLUS** command. You can run a script file using the **START CLPPlus** command. You can also run a script by copying its contents into the CLPPlus SQL buffer using the **GET CLPPlus** command and then issuing the **RUN CLPPlus** command.

#### About this task

This task shows how to run a script with the **CLPPLUS** command. For other methods, see the related links.

#### Procedure

To run a script with the **CLPPLUS** command:

Run the **CLPPLUS** command, specifying a script name. For example, consider a script file named `dept_query.sql`:

```
SET PAGESIZE 9999
SET ECHO ON
SELECT * FROM DEPT;
EXIT
```

To run the `dept_query.sql` script on the default **SAMPLE** database on port 50000 with a user name of **db2user** and password **passw0rd**, issue the following command:

```
c1pp1us db2user/passw0rd @dept_query
```

The `dept_query.sql` script file is run after the user connects to the database. When the script is run, the commands **SET PAGESIZE** and **SET ECHO ON** and the statement **SELECT \* FROM** are issued.

The output of the script is as follows. **ECHO ON** displays the statement that was issued in the script file, and the values of **DEPT** are displayed up to a page limit of 9999.

```
C:\DB2\9.7\db2 > c1pp1us db2user/passw0rd @dept_query.sql
Connected to DB2 v 9.7 (localhost:5444/db2sampl) AS db2user
```

```
SQL>
SELECT * FROM dept;
```

DEPT	NODNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
4	OPERATIONS	BOSTON

```
SQL >
```

```
EXIT
```

## Skipping and interrupting CLPPlus commands

As of V9.7 Fix Pack 3, CLPPlus allows you to skip and interrupt command execution as well as script execution.

Previous releases of CLPPlus did not include any support for skipping or interrupting the execution of commands and scripts. V9.7 Fix Pack 3 addresses this requirement.

You can interrupt any command or script that CLPPlus is running using the Ctrl+C keystroke. This is helpful when you encounter a long running query or script and need to return control back to the CLPPlus interface.

You can also skip to the next SQL> prompt by pressing the ENTER key twice in succession. This is helpful when you enter an incorrect command and want to cancel it. The incorrect command is retained in the buffer and can be edited using any of the CLPPlus commands you use for editing and review.

### Example

The following example shows a command being skipped.

```
SQL> select *
      2 from emploee           <- first carriage return
      3                      <- second carriage return
SQL>                        <- next sql prompt
```

### Comments in CLPPlus

As of V9.7 Fix Pack 3, CLPPlus adds the ability for you to include comments in your scripts and commands.

In CLPPlus, comments can span one or more lines. Comments that are contained on a single line start with # or --. Comments that span multiple lines are enclosed in /\* and \*/.

#### Example

The following examples show both single and multiple line comments.

```
SQL> # This is a single line comment
SQL>
SQL> -- This is also a single line comment
SQL>
SQL> /* This comment
spans
multiple lines. */
SQL>
```

## Escape characters in CLPPlus

As of V9.7 Fix Pack 4, you can use escape characters in CLPPlus commands and queries.

### Description

In CLPPlus, you can use the ampersand (&) character to substitute variables in SQL statements. Escape characters can be used to escape the ampersand character in input values to avoid substitution, for example "AT&M".

Escape characters can also be used to escape the characters "\$" and "%", which are used to reference shell and environment variables in CLPPlus.

You can define the escape character with the **SET** command. The default escape character is "\". For more information about the **SET** command, see the related reference.

### Example

1. This example shows the use of the default escape character which avoids "&M" being treated as a substitution variable.

```
SQL> set escape ON
SQL> insert into testtab values('AT&M');
DB250000I: The command completed successfully.
```

```
SQL> select * from testtab;
TEXT
-----
AT&M
```

2. This example shows the use of a user-defined escape character which avoids "&G" being treated as a substitution variable.

```
SQL> set escape ^
SQL> insert into testtab values('L^&G');
DB250000I: The command completed successfully.
```

```
SQL> select * from testtab;
TEXT
-----
AT&M
L&G
```

3. This example shows the behavior when no escape character is used. "&V" is treated as a substitution variable and requires the user to provide the input value of "Planet".

```
SQL> set escape OFF
SQL> insert into testtab values('Smarter &V');
Enter a value for variable V: Planet
```

```
Original statement: insert into testtab values('Smarter &V')
New statement with substitutions: insert into testtab values('Smarter Planet')
DB250000I: The command completed successfully.
```

```
SQL> select * from testtab;
TEXT
-----
AT&M
L&G
Smarter Planet
```

## Escape characters in CLPPlus

- This example shows the behavior when no escape character is used. "&V" is treated as a substitution variable and requires the user to provide the input value of "Gene".

```
SQL> set escape OFF
SQL> insert into testtab values('Blue \&V');
Enter a value for variable V: Gene
```

```
Original statement: insert into testtab values('Blue \&V')
New statement with substitutions: insert into testtab values('Blue \Gene')
DB250000I: The command completed successfully.
```

```
SQL> select * from testtab;
TEXT
-----
AT&M
L&G
Smarter Planet
Blue \Gene
```

- This example shows the behavior when an escape character is used. "\$100" is treated as a value and not a shell or environment variable.

```
SQL> set escape ON
SQL> insert into testsub values('\$100');
DB250000I: The command completed successfully.
```

```
SQL> select * from testsub;
TEXT
-----
$100
```

- This example shows the behavior when an escape character is used. "86%" is treated as a value and not a shell or environment variable.

```
SQL> set escape ON
SQL> insert into testsub values('86\%');
DB250000I: The command completed successfully.
```

```
SQL> select * from testsub;
TEXT
-----
$100
86%
```

## Bind variables in CLPPlus

Bind variables are used in place of literal values. If you issue SQL statements multiple times, you can use bind variables to reduce the number of literal values.

### Authorization

No special authorization is required.

### Declaration

A bind variable can be declared using the following syntax:

```
▶▶—VARIABLE—name—datatype;—————▶▶
```

*name*

Specifies the name of the bind variable.

#### **datatype**

Specifies the datatype associated with the bind variable. This can be one of: BOOLEAN, CHARACTER, DATE, DECIMAL, DOUBLE, FLOAT, INTEGER, REAL, SMALLINT, or VARCHAR.

As of V9.7 Fix Pack 2, REFCURSOR is also supported. REFCURSOR is used to receive the **OUT** parameter values of type **CURSOR** in procedures, functions, and anonymous PL/SQL blocks.

As of V9.7 Fix Pack 3, NUMBER, NUMBER(p[,s]) and VARCHAR2 are also supported. NUMBER and NUMBER(p[,s]) are implicitly mapped to the DECIMAL datatype. VARCHAR2 is implicitly mapped to the VARCHAR datatype.

As of V9.7 Fix Pack 4, with DB2 for Linux, UNIX, and Windows servers, CLPPlus allows the use of BOOLEAN, ROW, and ARRAY datatypes as parameters for stored procedures. You can run a stored procedure with the **CALL** or **EXEC** CLPPlus statements.

### Scope

Bind variables persist over the duration of a user's CLPPlus session. Once a CLPPlus session has started, bind variables can be declared and used for the duration of that session. When a CLPPlus session is ended, any bind variables are cleared.

### Restrictions

When used in an SQL statement or an anonymous PL/SQL block, a bind variable can appear only once. If the bind variable is used more than once an error from the database server is returned.

DB2 for z/OS® and Informix Dynamic Server data servers have the following limitations with the usage of bind variables:

- Bind variables cannot be initialized using the **EXEC** command.

```
Exec :var_name:='john' /* this is not supported */
```

- Bind variables cannot be initialized using a begin-end block.

## Bind variables in CLPPlus

```
begin
:var_name:='john'; /* this is not supported in a begin-end block */
end;
```

- Since PL/SQL is not supported on DB2 for z/OS and Informix Dynamic Server data servers, bind variables are not supported in a PL/SQL body.
- Variables with type CURSOR are not supported.

```
SQL> CREATE PROCEDURE getEmployeeData( ID INT, OUT NAME char(10),
      OUT DOB Date, OUT SAL DECIMAL(7,2))
      LET NAME='dummy';
      LET DOB='10/10/2010';
      LET SAL=0;
      SELECT empname, empdob, salary INTO name, dob, sal FROM emp WHERE empid = ID;
END PROCEDURE;
```

```
/
DB250000I: The command completed successfully.
```

```
SQL> define var_id=1001 /* usage of substitution variable */
```

```
SQL> Variable name varchar(10)
```

```
DB250000I: The command completed successfully.
```

```
SQL> Variable dob date
```

```
DB250000I: The command completed successfully.
```

```
SQL> Variable salary double
```

```
DB250000I: The command completed successfully.
```

```
Call getEmployeeData(&var_id, :name, :dob, :salary)
```

```
DB250000I: The command completed successfully.
```

```
SQL> Print name
```

```
'JOHN'
```

```
SQL> Print dob
```

```
'26/04/1982'
```

```
SQL> Print salary
```

```
10000.50
```

- Precision and scale values can be specified while creating bind variables of with the NUMBER and DECIMAL datatypes. There is a limitation in precision support. Any decimal or number values assigned are not modified to the precision specified in the definition of the variable. See example 13 for more details.

These restrictions apply to the **EXECUTE** CLPPlus command as well.

## Examples

The following examples show how you can define, initialize, and use bind variables.

1. Bind variables named **ID** and **LNAME** of type **VARCHAR**:

```
VARIABLE ID VARCHAR
VARIABLE LNAME VARCHAR
```

2. A bind variable named **ID** initialized in a PL/SQL block:

```
BEGIN
  :ID := '000020';
END;
/
```

3. Bind variables **ID** and **LNAME** used in a PL/SQL block:

```
BEGIN
  SELECT lastname INTO :LNAME FROM employee
  WHERE empno = :ID;
END;
/
```

4. A single PL/SQL statement initializes a bind variable named **ID** :

```
EXECUTE :ID := '000022';
```



5. The variable **ID** is initialized from a substitution variable *a* (*a* is defined with the **DEFINE** CLPPlus command):

```
EXECUTE :ID := &a;
```

6. The **ID** bind variable is used in a **SELECT** statement:

```
SELECT lastname FROM employee WHERE empno = :ID;
```

7. The **ID** and **LNAME** bind variables are used in an **UPDATE** statement:

```
UPDATE employee SET lastname = :LNAME WHERE empno = :ID;
```

8. The **salary** bind variable is defined with the number datatype:

```
variable salary number
exec :salary := 1000.00
```

9. The **bonus** bind variable is defined with the number(p[,s]) datatype:

```
variable bonus number(6)
exec :bonus:= 999.999
```

10. The **comm** bind variable is defined with the number(p[,s]) datatype:

```
variable bonus comm(4,2)
exec :comm:= 10.455
```

```
SQL> print comm
10.45
```

11. The **name** bind variable is defined with the varchar2 datatype:

```
variable name varchar2
exec :name:='MICHAEL'
```

12. This example shows the substitution of bind variables as input and output arguments in procedure execution. Assume a file named `example_proc.db2` contains the following statement:

```
CREATE PROCEDURE DEPT_MEDIAN
(IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
```

Bind variables substituted as input and output arguments. Define the bind variables:

```
variable in_var integer
variable out_var double
```

Run the procedure and substitute the variables as parameters:

```
call dept_median(:in_var, :out_var)
```

Optionally print the contents of the output argument, which is the **out\_var** bind variable:

```
print out_var
```

13. This example shows a bind variable **var1** which does not reflect the precision in the definition:

```
variable var1 number(4,2)
DB250000I: The command completed successfully.
```

Assign a value with a precision of 6 digits and a scale of 3 digits:

```
exec :var1 := 333.333
/
```

DB250000I: The command completed successfully.

Print the contents of **var1**:

```
print var1
333.33
```

## Bind variables in CLPPlus

The scale is correct, 2. The precision is not 4 as defined. This scenario is a current limitation in functionality.

## Shell and environment variables in CLPPlus

As of V9.7 Fix Pack 4, you can use shell and environment variables in CLPPlus commands and queries.

### Description

With this support, any shell or environment variable value can be used in a CLPPlus command or query. This use is identical to how these variables are accessed in shell or command windows.

Any variable prefixed with "\$" is treated as a shell or environment variable. This character provides operating system independence to scripts which are used across differing operating systems. Environment variables on Microsoft platforms can also be accessed by wrapping "%" around the variable name, for example, %VAR%.

CLPPlus understands and resolves a shell or environment variable if it includes the characters a-z, A-Z, 0-9, and "\_". If any other character is encountered, then it is treated as the end of the variable name. The following examples highlight this naming rule:

```
$clpplus&version <==== "clpplus" followed by "&"
$clpplus[version] <==== "clpplus" followed by "["
$clpplus version <==== "clpplus" followed by " "(space)
$clpplus#version <==== "clpplus" followed by "#"
$clpplus-version <==== "clpplus" followed by "-"
$clpplus(version) <==== "clpplus" followed by "("
```

If CLPPlus attempts to use a variable and it is defined, then its value is retrieved and used as expected. If CLPPlus attempts to use a variable and it is not defined, an empty string value is substituted and an informational message is logged. The informational message is visible when CLPPlus is operating in verbose mode.

You can access shell and environment variables in CLPPlus with the following methods:

- While starting CLPPlus, you can pass shell or environment variables as arguments to a script. These passed variables are converted into position parameters which are accessed with the "&" character and their position (number)
- Shell or environment variables can be directly accessed in CLPPlus with the "\$" character followed by the variable name. This method is irrespective of the operating system on which the script is run.

### Example

1. The following example shows how you can use shell or environment variables while starting CLPPlus:

On UNIX and Linux platforms:

```
Export TABLE_NAME=employee
Export SCRIPT_PATH=/home/user
```

```
clpplus -nw @$SCRIPT_PATH/script.sql $TABLE_NAME
```

On Windows platforms:

```
Set TABLE_NAME=employee
Set SCRIPT_PATH=c:\testfiles
```

```
clpplus -nw @%SCRIPT_PATH%\script.sql %TABLE_NAME%
```

## Shell and environment variables in CLPPlus

where `script.sql` contains

```
Select * from &1
```

and `&1` resolves to the `TABLE_NAME` variable value.

2. The following example shows how you can use shell or environment variables directly in CLPPlus:

On UNIX and Linux platforms:

```
select * from $TABLE_NAME;  
insert into test values ($NAME, $ID, $DESG);
```

On Windows platforms:

```
select * from %TABLE_NAME%;  
insert into test values (%NAME%, %ID%, %DESG%);
```

---

## CLPPlus traces and record logging

CLPPlus provides mechanisms for file traces and record logging. CLPPlus supports logging or traces from the CLPPlus client layer and JDBC driver layer.

The IBM Data Server Driver for JDBC and SQLJ and IBM Data Server Driver for ODBC and CLI offer comprehensive tracing facilities. These facilities have been extended to CLPPlus. Trace facilities generate text log files whenever an application accesses a specified driver (CLPPlus Client layer or JDBC Driver layer) using the `SET` command. These log files provide detailed information about the CLPPlus Client and JDBC:

- functions called by an application
- function contents; including input and output parameters passed to and received from
- function return codes and any error or warning messages generated.

To configure the CLPPlus trace facilities, issue the **SET** command from a CLPPlus command prompt. To enable client layer or driver layer traces, set the **LOGMODE** parameter:

```
CLPPlus> SET LOGMODE logmode-value
```

where *logmode-value* indicates whether to perform tracing and for which layer. The default value is `NONE`, which indicates no tracing is done. Other valid values are `CLPPLUS`, which traces the client layer, `JCC`, which traces the JDBC layer, and `BOTH`, which traces both the client and JDBC layers.

To perform more detailed JDBC tracing, set *logmode-value* to `JCC` or `BOTH`, and specify the **JCCLOGMODE** parameter:

```
SET LOGMODE JCC  
SET JCCLOGMODE jcclogmode_value
```

where *jcclogmode\_value* indicates the features to be traced and logged. For more information about valid *jcclogmode\_value* settings, see “SET” on page 740.

---

## DB2 commands supported by CLPPlus

CLPPlus supports a subset of DB2 commands for database and database manager administration, tuning, and maintenance.

The following DB2 commands are supported in the CLPPlus interface:

- **GET DATABASE CONFIGURATION**

- **GET DATABASE MANAGER CONFIGURATION**
- **UPDATE DATABASE CONFIGURATION**
- **UPDATE DATABASE MANAGER CONFIGURATION**
- **RESET DATABASE CONFIGURATION**
- **RESET DATABASE MANAGER CONFIGURATION**

As of V9.7 Fix Pack 3, CLPPlus support for the following DB2 commands is added:

- **LIST PACKAGES**
- **IMPORT**
- **EXPORT**
- **LOAD**

As of V9.7 Fix Pack 4, CLPPlus support for the following DB2 commands is added:

- **REORG**, also supported when connected to DB2 for z/OS.
- **RUNSTATS**, also supported when connected to DB2 for z/OS.
- **REORGCHK**, for more information see the related reference.

As of V9.7 Fix Pack 5, CLPPlus support for these DB2 commands is added:

- Limited support for **CREATE DATABASE**, also supported when connected to IBM Informix. For more information about restrictions see the related reference.
- **DROP DATABASE**, also supported when connected to IBM Informix.

---

## CLPPlus restrictions

CLPPlus has certain connection, command, and statement restrictions.

CLPPlus can establish database connections with the following DB2 database products but with the following restrictions:

- DB2 for Linux, UNIX, and Windows Version 9.8 Fix Pack 1 or higher. You must apply Fix Pack 1 to V9.8 before connectivity is supported.

CLPPlus has the following restrictions on PL/SQL support:

- PL/SQL statement compilation and execution on DB2 Personal Edition is not supported.
- PL/SQL functions and triggers cannot be created in a partitioned database environment.
- The NCLOB data type is not supported for use in PL/SQL statements or in PL/SQL contexts when the database is not defined as a Unicode database. In Unicode databases, the NCLOB data type is mapped to a DB2 DBCLOB data type.
- The XMLTYPE data type is not supported.
- TYPE declaration is not supported in a function, procedure, trigger, or anonymous block.
- The FOR EACH STATEMENT option is not supported for PL/SQL triggers.

## CLPPlus restrictions

Table 6. CLPPlus limitations across different data servers.

CLPPlus Feature	DB2 Database for Linux, UNIX, and Windows	DB2 for z/OS	IBM Informix
Variable REFCURSOR	Yes	No	No
SERVEROUTPUT	Yes	No	No
EXECUTE	Yes	No	No
LIST PACKAGES	Yes	Yes	No
SHOW ERRORS	Yes	No	No
UPDATE/GET/RESET DB CFG	Yes	No	No
UPDATE/GET/RESET DBM CFG	Yes	No	No
EXPORT	Yes	No	No
IMPORT	Yes	No	No
LOAD	Yes	No	No

## CLPPlus troubleshooting hints and tips

Use this information to help troubleshoot CLPPlus problems and errors.

Table 7. Starting CLPPlus: Issues and solutions

Issue	Explanation and solution
<p>The following message is displayed when you try to start CLPPlus:</p> <p>CLPPlus requires Java 1.5 or higher to execute. Please ensure Java is in your PATH.</p> <p>CLPPlus fails to start.</p>	<p>Ensure you have Java 1.5 or later installed. When the CLPPlus installation process does not install a Java product, CLPPlus looks for a Java product in the <b>JAVA_HOME</b> and <b>PATH</b> environment variables.</p> <p>Download and install a Java JRE or SDK version 1.5 or later. Set <b>JAVA_HOME</b> to point to the Java installation directory. Add the Java bin directory to your <b>PATH</b> environment variable.</p>
<p>The following message is displayed when you try to start CLPPlus:</p> <p>Could not find db2jcc.jar. Please ensure that your installation completed successfully. If the problem persists, please locate and add db2jcc.jar to your CLASSPATH.</p> <p>CLPPlus fails to start.</p>	<p>CLPPlus requires the Java universal drivers in the db2jcc.jar file. When CLPPlus cannot find the db2jcc.jar file in the <b>CLASSPATH</b> or in the <i>installation directory/java</i> directory, startup fails.</p> <p>Ensure that the installation was completed successfully. Add the absolute path of the db2jcc.jar file to the <b>CLASSPATH</b> environment variable setting.</p>

Table 7. Starting CLPPlus: Issues and solutions (continued)

Issue	Explanation and solution
<p>The following message is displayed when you try to start CLPPlus:</p> <p>Could not find clpplus.jar. Please ensure that your installation completed successfully. If the problem persists, please locate and add clpplus.jar to your CLASSPATH.</p> <p>CLPPlus fails to start.</p>	<p>CLPPlus requires the clpplus.jar file shipped with the product. When CLPPlus cannot find clpplus.jar in the <b>CLASSPATH</b> or in the installation directory, startup fails.</p> <p>Ensure that the installation was completed successfully. Add the absolute path of the db2jcc.jar file to the <b>CLASSPATH</b> environment variable setting.</p>
<p>The following message is displayed when you try to start CLPPlus:</p> <p>Could not find jline-0.9.93.jar. Please ensure that your installation completed successfully. If the problem persists, please locate and add jline-0.9.93.jar to your CLASSPATH.</p> <p>CLPPlus fails to start.</p>	<p>CLPPlus requires the jline-0.9.93.jar file shipped with the product. When CLPPlus cannot find jline-0.9.93.jar in the <b>CLASSPATH</b> or in the installation directory, CLPPlus fails to start.</p> <p>Ensure that the installation was completed successfully. Add the absolute path of the jline-0.9.93.jar file to the <b>CLASSPATH</b> environment variable setting.</p>





---

## Chapter 4. How to read command syntax help

From time to time, you might forget the options that are valid for a command. You can invoke useful command help screen output which uses a syntax convention which is explained here.

All Command Line Processor (CLP) commands can invoke a help screen at the CLP prompt by preceding the command keyword(s) with a question mark (?). For many of the system commands, a summarizing help screen can be displayed by issuing the command keyword followed by a **help** parameter.

### Invoking help

#### CLP commands

To display a CLP command help screen, preface the command keyword(s) with a question mark at the db2 interactive mode prompt (db2 =>), as shown in the following example for the **BACKUP DATABASE** command:

```
db2 => ? backup database
```

or, outside the 'db2' interactive mode, preface each command help screen invocation with db2, as shown for the **BACKUP DATABASE** command:

```
=> db2 ? backup database
```

#### System commands

Most of the system commands can display a command help screen by entering the system command keyword followed by a *help* option. Many system commands use a common *help* option, while other system commands may use different and/or additional *help* options. For the first attempts, without having to search for a command's forgotten *help* option just yet, try the following most common options which are likely to result in successfully invoking the command help screen:

#### Help options

- -h
- -?
- -help
- nothing entered after the command keyword.

**Note:** When nothing is entered after the command keyword, in some cases this could actually execute the command if options are not required.

### Help screen syntax conventions

[ ] Encloses optional parameters

{ } Encloses mandatory parameters

| Separates two or more items, only one of which may be chosen

... Indicates a repeatable parameter

( ) Repeatable parameter delimiter (not always used)

Command KEYWORDS appear in uppercase

variables, that require you to determine and enter the appropriate input, appear in lowercase

## How to read command syntax help

### Example command help screen output

The following is the CLP command help screen for the **UPDATE MONITOR SWITCHES** command:

```
db2 => ? update monitor
UPDATE MONITOR SWITCHES USING {switch-name {ON | OFF} ...}
[AT DBPARTITIONNUM db-partition-number | GLOBAL]
```

switch-name:  
BUFFERPOOL, LOCK, SORT, STATEMENT, TABLE, TIMESTAMP, UOW

The following is the system command help screen for the **db2look** command which, in this case, was not invoked by its specified **-h** help option:

```
C:\Program Files\IBM\SQLLIB\BIN>db2look
```

```
Syntax: db2look -d DBname [-e] [-xs] [-xdir Path] [-u Creator] [-z Schema]
        [-t Tname1 Tname2...TnameN] [-tw Tname] [-h]
        [-o Fname] [-a] [-m] [-c] [-r] [-l] [-x] [-xd] [-f]
        [-fd] [-td x] [-noview] [-i userID] [-w password]
        [-v Vname1 Vname2 ... VnameN] [-dp] [-ct]
        [-wrapper WrapperName] [-server ServerName] [-nofed]
        [-wlm] [-ap]

        [-wrapper WrapperName] [-server ServerName] [-fedonly] [-nofed]
```

```
db2look [-h]
```

-d: Database Name: This must be specified

-e: Extract DDL file needed to duplicate database

-xs: Export XSR objects and generate a script containing DDL statements

-xdir: Path name: the directory in which XSR objects will be placed

-u: Creator ID: If -u and -a are both not specified then \$USER will be used

-z: Schema name: If -z and -a are both specified then -z will be ignored

-t: Generate statistics for the specified tables

-tw: Generate DDLs for tables whose names match the pattern criteria (wildcard characters) of the table name

-ap: Generate AUDIT USING Statements

-wlm: Generate WLM specific DDL Statements

-h: More detailed help message

-o: Redirects the output to the given file name

-a: Generate statistics for all creators

-m: Run the db2look utility in mimic mode

-c: Do not generate COMMIT statements for mimic

-r: Do not generate RUNSTATS statements for mimic

-l: Generate Database Layout: Database partition groups, Bufferpools and Tablespace

-x: Generate Authorization statements DDL excluding the original definer of the object

-xd: Generate Authorization statements DDL including the original definer of the object

-f: Extract configuration parameters and environment variables

-td: Specifies x to be statement delimiter (default is semicolon(;))

-i: User ID to log on to the server where the database resides

-w: Password to log on to the server where the database resides

-noview: Do not generate CREATE VIEW ddl statements

-wrapper: Generates DDLs for federated objects that apply to this wrapper

-server: Generates DDLs for federated objects that apply to this server

-FEDONLY: Only created Federated DDL Statements

-nofed: Do not generate Federated DDL

-fd: Generates db2fopt statements for opt\_buffpage and opt\_sortheap along with other cfg and env parameters.

-v: Generate DDL for view only, this option is ignored when -t is specified

-dp: Generate DROP statement before CREATE statement

-ct: Generate DDL Statements by object creation time

**Note:** In general, a system command help screen tends to provide more detailed information than a CLP command help screen.

### Example command inputs

Using the **UPDATE MONITOR SWITCHES** command help screen as an example,

```
db2 => ? update monitor
UPDATE MONITOR SWITCHES USING {switch-name {ON | OFF} ...}
[AT DBPARTITIONNUM db-partition-number | GLOBAL]
```

```
switch-name:
  BUFFERPOOL, LOCK, SORT, STATEMENT, TABLE, TIMESTAMP, UOW
```

the following command inputs are valid,

```
UPDATE MONITOR SWITCHES USING LOCK OFF
UPDATE MONITOR SWITCHES USING LOCK OFF TIMESTAMP ON
UPDATE MONITOR SWITCHES USING STATEMENT ON AT DBPARTITIONNUM 1
UPDATE MONITOR SWITCHES USING SORT ON GLOBAL
```

while the following command inputs are invalid:

```
UPDATE MONITOR SWITCHES LOCK OFF
UPDATE MONITOR SWITCHES USING LOCK GLOBAL
UPDATE MONITOR SWITCHES USING STATEMENT ON AT DBPARTITIONNUM 1 GLOBAL
```

### Reminder

To remind yourself about the command help screen syntax conventions without searching the online Information Center, issue the following command at the CLP prompt:

```
db2 => ? help
```

or, at the system command prompt, enter the following query:

```
=> db2 ? help
```

## How to read command syntax help

---

## Chapter 5. CLP commands

---

# 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 target database on all members in the instance. If one or more of these members encounters an error during activation, a warning is returned. The database remains activated on all members on which the command succeeds.

A database is activated either implicitly or explicitly. The activation of a database driven by a first user connection is known as implicit activation. Activation of a database that is based on issuing the activate database command is known as explicit activation.

### Authorization

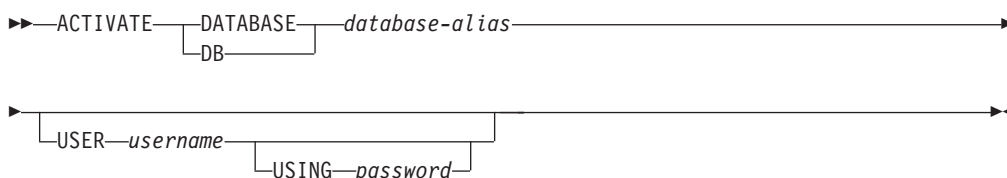
One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT

### Required connection

None

### Command syntax



### Command parameters

**DATABASE | DB** *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

Database administrators can use the **ACTIVATE DATABASE** command to start the selected databases. This eliminates any application time spent on database initialization. The **ACTIVATE DATABASE** command is the only way to explicitly activate a database.

Databases initialized by the **ACTIVATE DATABASE** command can be shut down by using the **DEACTIVATE DATABASE** or **db2stop** command.

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

If a database is started by issuing a **CONNECT** statement (or an implicit connect) and subsequently an **ACTIVATE DATABASE** command is issued for that same database, then the **DEACTIVATE DATABASE** command must be used to shut down that database. If the **ACTIVATE DATABASE** command was not used to start the database, the database will shut down when the last application disconnects. However, in a DB2 pureScale® environment, an active database on a member can only be deactivated on that member by issuing a **DEACTIVATE DATABASE** command.

In a DB2 pureScale environment, neither the **ACTIVATE DATABASE** command nor the **CONNECT** statement can be used to restart a database. In this case, the database is automatically restarted depending on the **autorestart** database configuration parameter. See the **autorestart** configuration parameter for database connection behavior when the target database is in an inconsistent state.

**Related information:**

## ADD CONTACT

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.

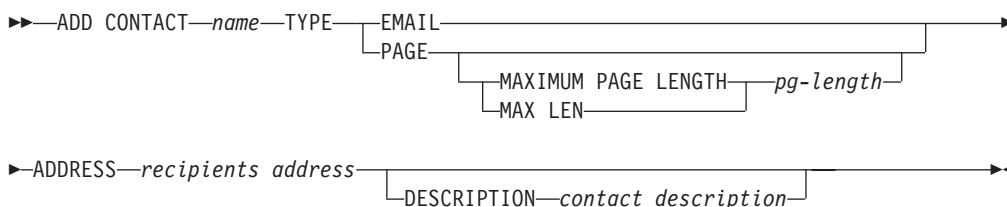
### Authorization

None

### Required connection

None. Local execution only: this command cannot be used with a remote connection.

### Command syntax



### Command parameters

#### ADD 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 wants to be notified by email at (**ADDRESS**).

**PAGE** This contact wants 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 email 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 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.

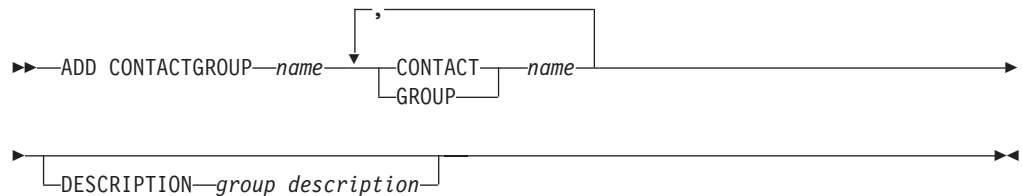
### Authorization

None

### Required connection

None. Local execution only: this command cannot be used with a remote connection.

### Command Syntax



### Command Parameters

#### **ADD 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.

## ADD DBPARTITIONNUM

Adds a database partition to a database partition server.

### Scope

This command only affects the database partition server on which it is executed.

### Authorization

One of the following authorities:

- 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 (this refers to 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 table space's storage group. 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 (this refers to 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

This command should only be used if a database partition server is added to an environment that has one database and that database is not cataloged at the time of the add partition operation. In this situation, because the database is not cataloged, the add partition operation does not recognize the database, and does not create a database partition for the database on the new database partition server. Any attempt to connect to the database partition on the new database partition server results in an error. The database must first be cataloged before the **ADD DBPARTITIONNUM** command can be used to create the database partition for the database on the new database partition server.

This command should not be used if the environment has more than one database and at least one of the databases is cataloged at the time of the add partition operation. In this situation, use the **AT DBPARTITIONNUM** parameter of the **CREATE DATABASE** command to create a database partition for each database that was not cataloged at the time of the add partition operation. Each uncataloged database must first be cataloged before the **CREATE DATABASE** command can be used to create the database partition for the database on the new database partition server.

Before adding a new database partition, ensure that there is sufficient storage for the containers that must be created.

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.

**Note:** Any uncataloged database is not recognized when adding a new database partition. The uncataloged database will not be present on the new database partition. An attempt to connect to the database on the new database partition returns the error message SQL1013N.

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.

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 group definitions are retrieved as part of that communication. Likewise, if system

## ADD DBPARTITIONNUM

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.

When the **ADD DBPARTITIONNUM** command is issued there cannot be any storage group entries that are not transaction consistent.

## ADD XMLSCHEMA DOCUMENT

Adds one or more XML schema documents to an existing but incomplete XML schema before completing registration.

### Authorization

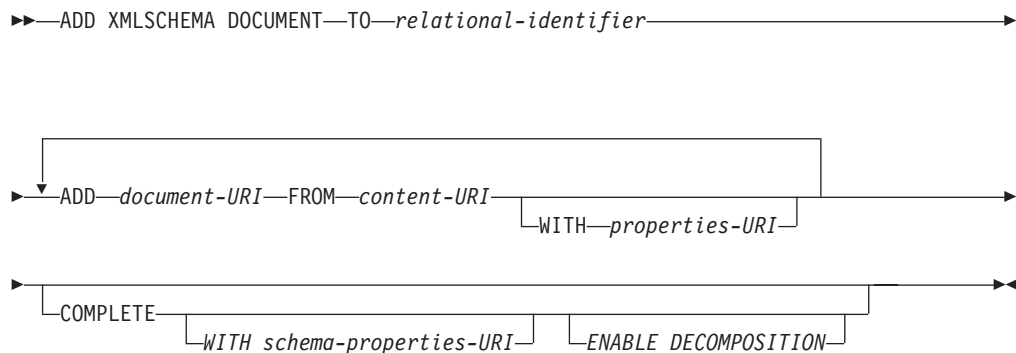
The following authority is required:

- 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'
```

## ARCHIVE LOG

Closes and truncates the active log file for a recoverable database.

### Authorization

One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- DBADM

### Required connection

None. This command establishes a database connection for the duration of the command.

### Command syntax

```

▶▶ ARCHIVE LOG FOR DATABASE database-alias
   DB
▶
  USER username
   USING password
▶
  On Database Partition Number Clause
▶▶

```

#### On Database Partition Number Clause:

```

| ON Database Partition Number List Clause
| ALL DBPARTITIONNUMS
| EXCEPT Database Partition Number List Clause
|

```

#### Database Partition Number List Clause:

```

| DBPARTITIONNUM
| DBPARTITIONNUMS
▶
  ( db-partition-number
    TO db-partition-number )
▶

```

### Command parameters

**DATABASE** *database-alias*

Specifies the alias of the database whose active log is to be archived.

**USER** *username*

Specifies 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 data up to a known point. The log data can then be used to update a standby database.

If log data up to the time the **ARCHIVE LOG** command is issued is in the middle of a log file, this log file will be truncated and logging will continue on the next file.

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. The command may fail with SQL0964 if not enough log space is available.

The **ARCHIVE LOG** command is asynchronous. When you issue the **ARCHIVE LOG** command, the log is closed, making it available for archiving. The log is not



archived immediately; there might be a delay between the time when you submit the command and the time when the log is archived. This delay is particularly apparent if you deactivate a database immediately after issuing the **ARCHIVE LOG** command. Archiving of the log is done by the **db2logmgr** process. The log may not archive until the next database activation.

The command is distributed to every member the database has access to when issuing the **ARCHIVE LOG** command or invoking the db2ArchiveLog API in a DB2 pureScale environment. If a member is consistent (either online or offline), then **ARCHIVE LOG** will skip that member, since its last log file would have already been truncated when the member had last shut down cleanly. If an offline member is not consistent, then **ARCHIVE LOG** waits for MCR to complete before truncation.

## 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.

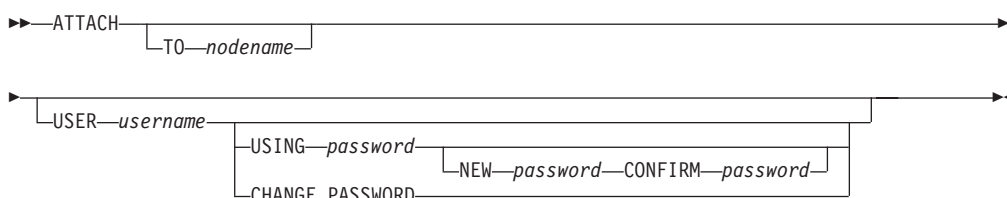
### Authorization

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 flat-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. The system on which the password will be changed depends on how user authentication has been set up. The DB2 database system provides support for changing passwords on AIX, Linux and Windows operating systems, and supports up to 255 characters for your own written plugins. See *Password rules* for additional information about passwords.

#### 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.

## AUTOCONFIGURE

Calculates and displays initial values for the buffer pool size, database configuration and database manager configuration parameters, with the option of applying these reported values.

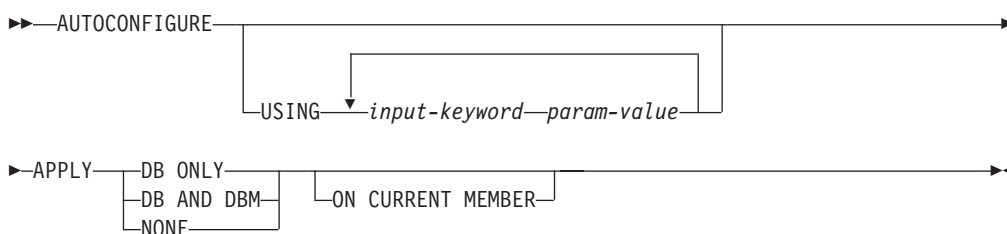
### Authorization

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
<b>mem_percent</b>	1-100	25	Percentage of instance memory that is assigned to the database. However, if the <b>CREATE DATABASE</b> command invokes the configuration advisor and you do not specify a value for <b>mem_percent</b> , the percentage is calculated based on memory usage in the instance and the system up to a maximum of 25% of the instance memory.
<b>workload_type</b>	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.
<b>num_stmts</b>	1-1 000 000	10	Number of statements per unit of work
<b>tpm</b>	1-200 000	60	Transactions per minute
<b>admin_priority</b>	performance, recovery, both	both	Optimize for better performance (more transactions per minute) or better recovery time
<b>is_populated</b>	yes, no	yes	Is the database populated with data?
<b>num_local_apps</b>	0-5 000	0	Number of connected local applications
<b>num_remote_apps</b>	0-5 000	10	Number of connected remote applications

Table 8. Valid input keywords and parameter values (continued)

Keyword	Valid values	Default value	Explanation
<b>isolation</b>	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.
<b>bp_resizeable</b>	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 MEMBER**

In a partitioned database environment or DB2 pureScale environment, the Configuration Advisor updates the database configuration on all members by default. Specifying the **ON CURRENT MEMBER** option causes the Configuration Advisor to set the member-level configuration parameters on the current member determined by your connection, while the global-level configuration parameters, that can be configured to be functional at only the global level, are set and affect all members.

The buffer pool changes are always applied to the system catalogs. Thus, all members are affected. The **ON CURRENT MEMBER** option is ignored for buffer pool recommendations.

**Usage notes**

- This command makes configuration recommendations for the currently connected database and assumes that the database is the only active database on the instance. If you have not enabled the self tuning memory manager and you have more than one active database on the instance, specify a **mem\_percent** value that reflects the database memory distribution. For example, if you have two active databases on the instance that should use 80% of the instance memory and should share the resources equally, specify 40% (80% divided by 2 databases) as the **mem\_percent** value.
- If you have multiple instances on the same computer and the self tuning memory manager is not enabled, you should set a fixed value for **instance\_memory** on each instance or specify a **mem\_percent** value that reflects the database memory distribution. For example, if all active databases should use 80% of the computer memory and there are 4 instances each with one database, specify 20% (80% divided by 4 databases) as the **mem\_percent** value.

## AUTOCONFIGURE

- 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.

### Compatibilities

For compatibility with previous versions:

- **NODE** and **DBPARTITIONNUM** can be specified in place of **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.

## BACKUP DATABASE

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

For information about 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”.

### Scope

In a partitioned database environment, if no database partitions are specified, this command affects only the database partition on which it is executed.

If the option to perform a partitioned backup is specified, the command can be called only on the catalog database partition. If the option specifies that all database partition servers are to be backed up, it affects all database partition servers that are listed in the `db2nodes.cfg` file. Otherwise, it affects the database partition servers that are specified on the command.

### Authorization

One of the following authorities:

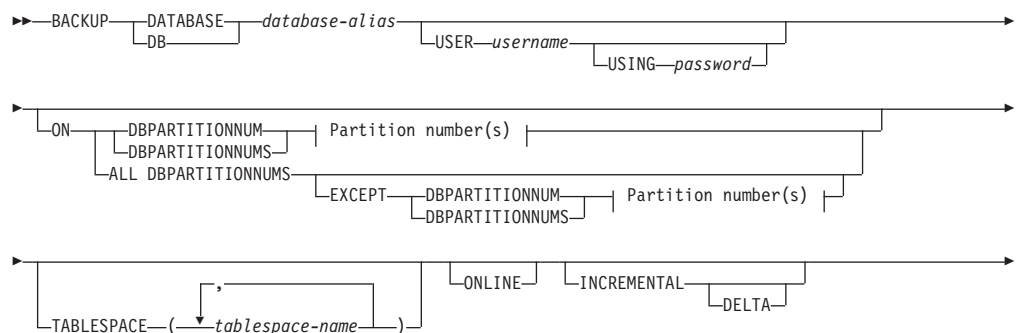
- SYSADM
- SYSCtrl
- SYSMAINT

### Required connection

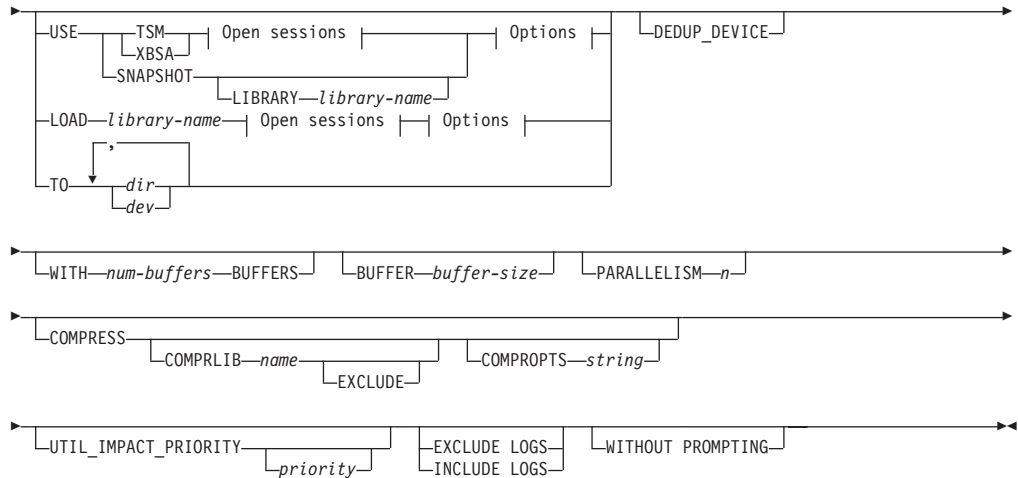
Database. This command automatically establishes a connection to the specified database.

**Note:** 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.

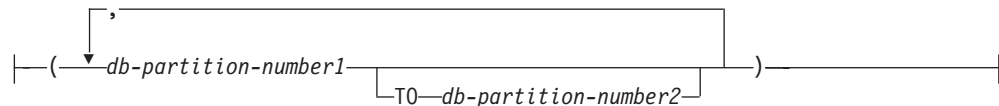
### Command syntax



## BACKUP DATABASE



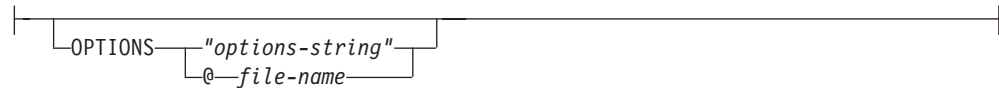
### Partition number(s):



### Open sessions:



### Options:



## Command parameters

### **DATABASE** | **DB** *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.

### **ON** Backup the database on a set of database partitions.

#### **DBPARTITIONNUM** *db-partition-number1*

Specifies a database partition number in the database partition list.

#### **DBPARTITIONNUMS** *db-partition-number1 TO db-partition-number2*

Specifies a range of database partition numbers, so that all partitions from *db-partition-number1* up to and including *db-partition-number2* are included in the database partition list.



### ALL DBPARTITIONNUMS

Specifies that the database is to be backed up on all partitions specified in the `db2nodes.cfg` file.

**EXCEPT** Specifies that the database is to be backed up on all partitions specified in the `db2nodes.cfg` file, except those specified in the database partition list.

**DBPARTITIONNUM** *db-partition-number1*

Specifies a database partition number in the database partition list.

**DBPARTITIONNUMS** *db-partition-number1 TO db-partition-number2*

Specifies a range of database partition numbers, so that all partitions from *db-partition-number1* up to and including *db-partition-number2* are included in the database partition list.

**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 **logarchmeth1** enabled. During an online backup, DB2 obtains IN (Intent None) locks on all tables existing in SMS table spaces as they are processed. S (share locks) are no longer held on LOB data in SMS table spaces during online backup.

### 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 noncumulative (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 (TSM) as the target device.

**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.

### SNAPSHOT

Specifies that a snapshot backup is to be taken.

You cannot use the **SNAPSHOT** parameter with any of the following parameters:

- **TABLESPACE**
- **INCREMENTAL**
- **WITH num-buffers BUFFERS**
- **BUFFER**
- **PARALLELISM**
- **COMPRESS**
- **UTIL\_IMPACT\_PRIORITY**

### • SESSIONS

The default behavior for a snapshot backup is a full database offline backup of all paths that make up the database including all containers, local volume directory, database path (**DBPATH**), and primary log and mirror log paths (**INCLUDE LOGS** is the default for all snapshot backups unless **EXCLUDE LOGS** is explicitly stated).

#### **LIBRARY** *library-name*

Integrated into IBM Data Server is a DB2 ACS API driver for the following storage hardware:

- IBM TotalStorage SAN Volume Controller
- IBM Enterprise Storage Server<sup>®</sup> Model 800
- IBM System Storage<sup>®</sup> DS6000<sup>™</sup>
- IBM System Storage DS8000<sup>®</sup>
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS

If you have other storage hardware, and a DB2 ACS API driver for that storage hardware, you can use the **LIBRARY** parameter to specify the DB2 ACS API driver.

The value of the **LIBRARY** parameter is a fully-qualified library file name.

### OPTIONS

#### *"options-string"*

Specifies options to be used for the backup operation. The string will be passed exactly as it was entered, without the double quotation marks.

#### *@ 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. The file must be a fully qualified file name.

You cannot use the **vendoropt** database configuration parameter to specify vendor-specific options for snapshot backup operations. You must use the **OPTIONS** parameter of the backup utilities instead.

### **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. For an online backup, if the **INCLUDE LOGS** option is specified than an extra session will be created for this parameter after the initial sessions are closed.

### **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.

In a partitioned database, the target directory or device must exist on all database partitions, and can optionally be a shared path. The directory or

device name may be specified using a database partition expression. For more information about database partition expressions, see “Automatic storage databases”.

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.

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.

### **DEDUP\_DEVICE**

Optimizes the format of the backup images for target storage devices that support data deduplication.

### **WITH** *num-buffers* **BUFFERS**

The number of buffers to be used. If the number of buffers that you specify is not enough to create a successful backup, then the minimum value necessary to complete the backup is automatically chosen for this parameter. If you are backing up to multiple locations, you can specify a larger number of buffers to improve performance. If you specify the **COMPRESS** parameter, to help improve performance, you can add an extra buffer for each table space that you specify for the **PARALLELISM** parameter.

### **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 the default buffer size included with DB2 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 (for example, `db2compr.dll` for Windows; `libdb2compr.so` for Linux and UNIX operating systems). 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, with the exception of snapshot backups. Logs are excluded by default in the following backup scenarios:

- Offline backup of a single-partitioned database.
- Online or offline backup of a multi-partitioned database, when not using a single system view backup.

### **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, with the exception of snapshot backups. **INCLUDE LOGS** is always the default option for any online backup operation, except a multi-partitioned online backup where each database partition is backed up independently (that is, a non-single system view backup).

If any of the log files that are required for the backup have previously been backed up and are no longer in the log path, then the DB2 database manager retrieves them for backup from the overflow log path, if the path has been set. Otherwise, the database manager retrieves them for backup from the current log path or mirror log path. These log files are removed from the log path after the backup has completed.

### WITHOUT PROMPTING

Specifies that the backup will run unattended, and that any actions which normally require user intervention will return an error message.

#### Note:

1. If the backup command indicates which partitions in a partitioned database are to be backed up, the backup operation is implicitly performed **WITHOUT PROMPTING**.
2. Options that are specified on the **BACKUP DATABASE** command in a partitioned database environment will be applied on each partition individually. For example, if a backup operation is specified to **USE TSM OPEN 3 SESSIONS**, DB2 will open three TSM sessions on each partition.

### 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. To perform an offline backup of all the WSDB database partitions to /dev3/backup, issue the following command from database partition 0:

```
db2 BACKUP DATABASE wsdb ON ALL DBPARTITIONNUMS TO /dev3/backup
```

The backup is performed simultaneously on all partitions. All four database partition backup images will be stored in the /dev3/backup directory, which can be a shared directory accessible from more than one partition, or a locally-mounted directory accessible from each partition individually, or a combination of both.

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

## BACKUP DATABASE

5. Following is a sample weekly incremental backup strategy for a recoverable database. It includes a weekly full database backup operation, a daily noncumulative (delta) backup operation, and a midweek 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.

7. In the following example, the database WSDB is defined on all 4 database partitions, numbered 0 through 3. Database partition 0 is the catalog partition. To perform an online backup of table space USERSPACE1 on database partitions 1 and 2, with the backup image to be stored on a TSM server, issue the following command from partition 0:

```
db2 BACKUP DATABASE wsdb ON DBPARTITIONNUMS (1, 2) TABLESPACE (USERSPACE1)
ONLINE USE TSM
```

8. Sample output generated to indicate the sqlcode returned by each partition.

### Example 1

All partitions are successful (sqlcode >= 0)

```
$ db2 backup db foo on all dbpartitionnums tablespace(T1)
```

```
Part Result
```

```
----
```

```
0000 DB20000I The BACKUP DATABASE command completed successfully.
```

```
0001 SQL2430W The database backup succeeded, but the following
table spaces do not exist on this database partition: "T1".
```

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

```
20040908010203
```

### Example 2

One or more partitions fail (sqlcode < 0)

```
$ db2 backup db foo on all dbpartitionnums to /backups
```

```
Part Result
```

```
----
```

```
0000 SQL2001N The utility was interrupted. The output data may be
incomplete.
```

```
0001 SQL2419N The target disk "/backups" has become full.
```

```
SQL2429N The database backup failed. The following database
partitions returned errors: "1".
```

9. The following backups will include the log directories in the image created:

```
db2 backup db sample use snapshot
```

```
db2 backup db sample online use snapshot
```

```
db2 backup db sample use snapshot INCLUDE LOGS
```

```
db2 backup db sample online use snapshot INCLUDE LOGS
```

10. The following backups will NOT include the log directories in the image created:

```
db2 backup db sample use snapshot EXCLUDE LOGS
```

```
db2 backup db sample online use snapshot EXCLUDE LOGS
```

### 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.
- 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.
- Snapshot backups should be complemented with regular disk backups in case of failure in the filer/storage system.
- As you regularly backup your database, you might accumulate very large database backup images, many database logs and load copy images, all of which might be taking up a large amount of disk space. Refer to “Managing recovery objects” for information about how to manage these recovery objects.
- You can use the **OPTIONS** parameter to enable backup operations in TSM environments supporting proxy nodes. For more information, see the “Configuring a Tivoli Storage Manager client” topic.

### Related information:

**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 authorizations:

- DBADM authority
- If EXPLAIN ONLY is specified, EXPLAIN authority or an authority that implicitly includes EXPLAIN is sufficient.
- If a package does not exist, BINDADD authority and:
  - If the schema name of the package does not exist, IMPLICIT\_SCHEMA authority on the database.
  - If the schema name of the package does exist, CREATEIN privilege on the schema.
- If the package exists, one of the following privileges:
  - ALTERIN privilege on the schema
  - BIND privilege on the package

In addition, if capturing explain information using the EXPLAIN or the EXPLSNAP clause, one of the following authorizations is required:

- INSERT privilege on the explain tables
- DATAACCESS authority

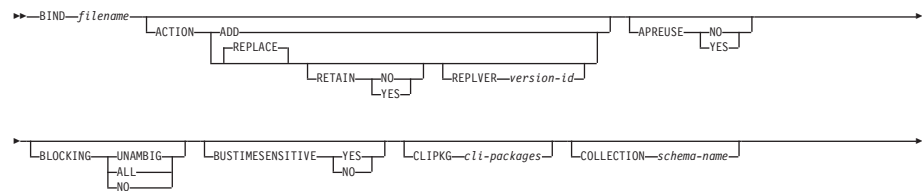
The user also needs all privileges required to compile any static SQL statements in the application. Privileges granted to groups are not used for authorization checking of static statements.

**Required connection**

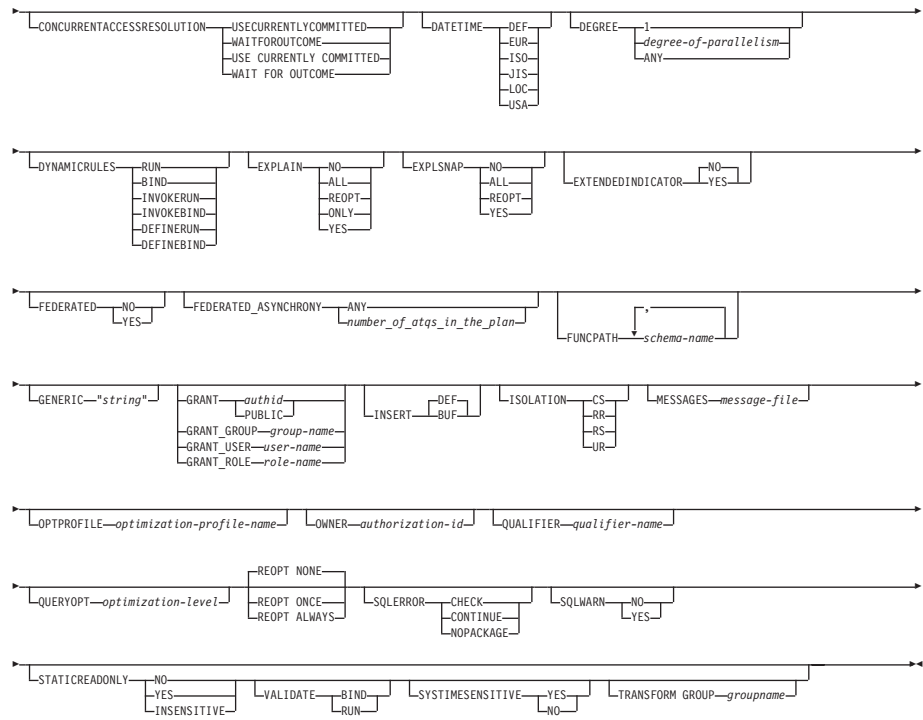
Database. If implicit connect is enabled, a connection to the default database is established.

**Command syntax**

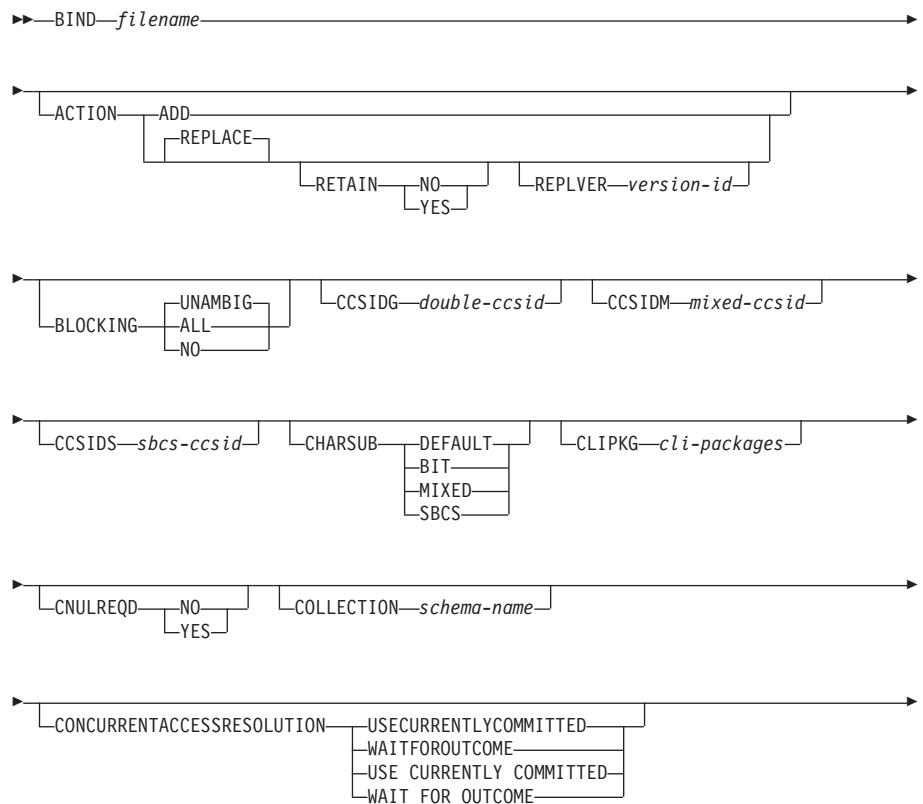
For DB2 for Linux, Windows and UNIX



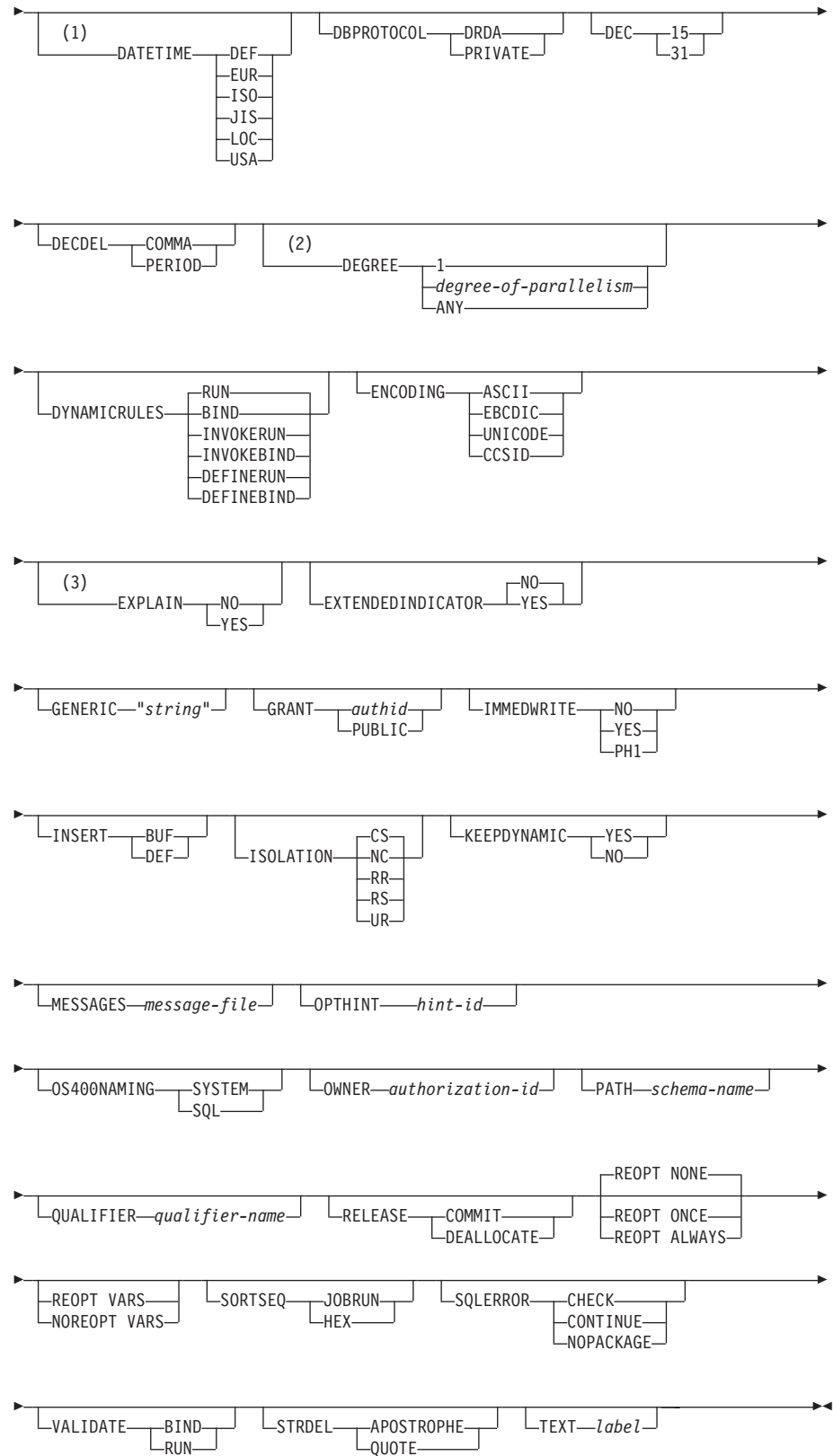




For DB2 on servers other than Linux, Windows and UNIX



# 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<sup>®</sup> 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

## BIND

name, creator, and version of the package being bound, that package will be replaced; if not, a new package will be added.

### APREUSE

Specifies whether static SQL access plans are to be reused. When this option is enabled, the query compiler will attempt to reuse the access plans for the statement in any existing packages during the bind and during future implicit and explicit rebinds.

**YES** The query compiler will attempt to reuse the access plans for the statements in the package. If there is an existing package, the query compiler will attempt to reuse the access plan for every statement that can be matched with a statement in the new bind file. For a statement to match, the statement text must be identical and the section number for the statement in the existing package must match what the section number will be for the statement in the new package.

**NO** The query compiler will not attempt to reuse access plans for the statements in the package. This is the default setting.

### BLOCKING

Specifies the type of row blocking for cursors. The blocking of row data that contains references to LOB column data types is also supported in partitioned database environments.

**ALL** For cursors that are specified with the FOR READ ONLY clause or cursors not specified as FOR UPDATE, blocking occurs.

Ambiguous cursors are treated as read-only.

**NO** Blocking does not occur for any cursor.

For the definition of a read-only cursor and an ambiguous cursor, refer to *DECLARE CURSOR statement*.

Ambiguous cursors are treated as updatable.

### UNAMBIG

For cursors that are specified with the FOR READ ONLY clause, blocking occurs.

Cursors that are not declared with the FOR READ ONLY or FOR UPDATE clause which are not ambiguous and are read-only will be blocked. Ambiguous cursors will not be blocked.

Ambiguous cursors are treated as updatable.

### BUSTIMESENSITIVE

Indicates whether references to application-period temporal tables in static and dynamic SQL statements are affected by the value of the CURRENT TEMPORAL BUSINESS\_TIME special register.

**YES** References to application-period temporal tables are affected by the value of the CURRENT TEMPORAL BUSINESS\_TIME special register. This value is the default value.

**NO** References to application-period temporal tables are not affected by the value of the CURRENT TEMPORAL BUSINESS\_TIME special register.

### 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 the server for DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a built-in 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 the server for DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a built-in default value if this option is not specified.

**CCSIDS** *sbc-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 built-in 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 precompile/bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows.

**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 built-in 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. It is valid only if the bind file is created from a C or a C++ application. This bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows.

**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.

## BIND

### **COLLECTION** *schema-name*

Specifies a 128-byte collection identifier for the package. If not specified, the authorization identifier for the user processing the package is used.

### **CONCURRENTACCESSRESOLUTION**

Specifies the concurrent access resolution to use for statements in the package.

### **USE CURRENTLY COMMITTED**

Specifies that the database manager can use the currently committed version of the data for applicable scans when it is in the process of being updated or deleted. Rows in the process of being inserted can be skipped. This clause applies when the isolation level in effect is Cursor Stability or Read Stability (for Read Stability it skips uncommitted inserts only) and is ignored otherwise. Applicable scans include read-only scans that can be part of a read-only statement as well as a non read-only statement. The settings for the registry variables **DB2\_EVALUNCOMMITTED**, **DB2\_SKIPDELETED**, and **DB2\_SKIPINSERTED** do not apply to scans using currently committed. However, the settings for these registry variables still apply to scans that do not use currently committed.

### **WAIT FOR OUTCOME**

Specifies Cursor Stability and higher scans wait for the commit or rollback when encountering data in the process of being updated or deleted. Rows in the process of being inserted are not skipped. The settings for the registry variables **DB2\_EVALUNCOMMITTED**, **DB2\_SKIPDELETED**, and **DB2\_SKIPINSERTED** no longer apply.

### **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 precompile/bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a built-in 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 precompile/bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a built-in 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 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 stand-alone 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.

## **BIND**

If the package is used as a stand-alone 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 stand-alone 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 stand-alone 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.

**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).

**ONLY** The **ONLY** option allows you to explain statements without having the privilege to execute them. The explain tables are populated but no persistent package is created. If an existing package with the same name and version is encountered during the bind process, the existing package is neither dropped nor replaced even if you specified **ACTION REPLACE**. If an error occurs during population of the explain tables, explain information is not added for the statement that returned the error and for any statements that follow it.

**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).

**EXPLSNAP**

Stores Explain Snapshot information in the Explain tables.

**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 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

## BIND

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).

### EXTENDEDINDICATOR

Enables the recognition of extended indicator variable values during the execution of the associated plan or package.

- NO** Extended indicator variable values are not recognized. Indicator variables are normal indicator variables; negative indicator variable values imply null, and positive or zero values imply non-null. This is the default condition.
- YES** Extended indicator variable values are recognized. Using any non-recognized indicator variable values, or using the default or unassigned indicator variable-based values in a non-supported location will cause DB2 database manager to generate an error message during execution of the bound statement.

### 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.

- 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.

### FEDERATED\_ASYNCHRONY

Specifies the maximum number of asynchrony table queues (ATQs) that the federated server supports in the access plan for programs that use embedded SQL.

- ANY** The optimizer determines the number of ATQs for the access plan. The optimizer assigns an ATQ to all eligible SHIP or remote pushdown operators in the plan. The value that is specified for **DB2\_MAX\_ASYNC\_REQUESTS\_PER\_QUERY** server option limits the number of asynchronous requests.

*number\_of\_atqs\_in\_the\_plan*

The number of ATQs in the plan. You specify a number in the range 0 to 32767.

**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 schema name SYSPUBLIC cannot be specified for the function path. The number of schemas that can be specified is limited by the length of the resulting function path, which cannot exceed 2048 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 supported by the target server, but not supported by DB2 for Linux, UNIX, and Windows as options for BIND PACKAGE or REBIND PACKAGE. Do not use this option to pass bind options that *are* defined in **BIND** or **PRECOMPILE**. 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, one could use the following to bind each of the OPT1, OPT2, and OPT3 options:

```
generic "opt1 value1 opt2 value2 opt3 value3"
```

The maximum length of the string is 4096 bytes. The maximum length of each option name in the string is 255 bytes.

For remote **BIND** commands, the BIND GENERIC clause can be used to flow new DB2 server bind options without requiring upgrade of the client. The client will pass the **BIND** command to the DB2 server without checking the syntax of this option. The DB2 server will check **BIND** command syntax and process.

**GRANT**

**Note:** If more than one of the **GRANT**, **GRANT\_GROUP**, **GRANT\_USER**, and **GRANT\_ROLE** options are specified, only the last option specified is executed.

*authid* Grants EXECUTE and BIND privileges to a specified user name, role name or group ID. The SQL GRANT statement and its rules are used to determine the type of *authid* when none of USER, GROUP, or ROLE is provided to specify the type of the grantee on a GRANT statement. For the rules, see *GRANT (Role) statement*.

**PUBLIC**

Grants EXECUTE and BIND privileges to PUBLIC.

**GRANT\_GROUP** *group-name*

Grants EXECUTE and BIND privileges to a specified group name.

**GRANT\_USER** *user-name*

Grants EXECUTE and BIND privileges to a specified user name.

## BIND

### GRANT\_ROLE *role-name*

Grants EXECUTE and BIND privileges to a specified role name.

### 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 Database for Linux, UNIX, and Windows.

**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 page sets 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

This parameter specifies whether dynamic SQL statements are to be kept after commit points.

For a list of supported option values supported for DB2 for z/OS, see documentation in the DB2 for z/OS Information Center.

Starting with DB2 for Linux, UNIX, and Windows Version 9.8 Fix Pack 2, you can modify the value of the **KEEPDYNAMIC** bind option for a package without requiring a fresh bind operation, thereby avoiding unnecessary recompilation until the next bind operation occurs. The **KEEPDYNAMIC** bind option controls how long the statement text and section associated with a prepared statement are kept in the SQL context, specifying whether dynamic SQL statements are kept after a COMMIT or ROLLBACK.

#### YES

Instructs the SQL context to keep the statement text and section associated with prepared statements indefinitely. Dynamic SQL statements are kept across transactions. All packages bound with **KEEPDYNAMIC YES** are by default compatible with the existing package cache behavior.

#### No

Instructs the SQL context to remove the statement text and section associated with prepared statements at the end of each unit of

work. Inactive dynamic SQL statements prepared in a package bound with **KEEPDYNAMIC NO** are removed from the SQL context during a COMMIT or ROLLBACK operation. The statements must be prepared again in a new transaction. The client, driver, or application needs to prepare any dynamic SQL statement it wants to reuse in a new unit of work again. If the package is executed by a remote application, executable versions for prepared statements are disassociated from the application SQL context when the transaction ends.

Active dynamic SQL statements must be kept until the next COMMIT or ROLLBACK operation where they are inactive. Here are some situations where dynamic SQL statements can be active at transaction boundaries:

- Cursors declared using the WITH HOLD option are open at a commit point.
- A dynamic SQL statement is executing a COMMIT or ROLLBACK operation.
- A dynamic SQL statement invokes a stored procedure or a user defined function that is executing COMMIT or ROLLBACK operation.

#### **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.

#### **OPTPROFILE** *optimization-profile-name*

Specifies the name of an existing optimization profile to be used for all static statements in the package. The default value of the option is an empty string. The value also applies as the default for dynamic preparation of DML statements for which the CURRENT OPTIMIZATION PROFILE special register is null. If the specified name is unqualified, it is an SQL identifier, which is implicitly qualified by the **QUALIFIER** bind option.

The **BIND** command does not process the optimization file, but only validates that the name is syntactically valid. Therefore if the optimization profile does not exist or is invalid, an SQL0437W warning with reason code 13 will not occur until a DML statement is optimized using that optimization profile.

#### **OS400NAMING**

Specifies which naming option is to be used when accessing DB2 for System i® data. Supported by DB2 for System i only. For a list of supported option values, refer to the documentation for DB2 for System i.

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 System i 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 System i system naming convention is used, whether or not it has been precompiled or bound using the **OS400NAMING** SYSTEM option.

**OWNER** *authorization-id*

Designates a 128-byte 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 DBADM authority can specify an authorization identifier other than the user ID. The default value is the authorization ID of the invoker of the precompile/bind process. SYSIBM, SYSCAT, and SYSSTAT are not valid values for this option. The *authorization-id* must be a user. A role or a group cannot be specified using the **OWNER** 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 128-byte 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 required 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.

**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 Database for Linux, UNIX, and Windows. For DB2 z/OS Version 10 servers or later, the default value is DEALLOCATE.

**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 System i. Supported by DB2 for System i only. For a list of supported option values, refer to the documentation for DB2 for System i.

**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, global variables, and special registers. Valid values are:

**NONE**

The access path for a given SQL statement containing host variables, parameter markers, global variables, 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, global variables, 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, global variables, 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 previous compatibility. Specifies whether to have DB2 determine an access path at run time using values for host variables, global 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** or **INSENSITIVE**.

**NO** All static cursors will take on the attributes as would normally be

## BIND

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.

### INSENSITIVE

Any static cursor that does not contain the FOR UPDATE clause will be considered READ ONLY and INSENSITIVE.

### STRDEL

Designates whether an apostrophe (') or double quotation marks (") will be used as the string delimiter within SQL statements. This precompile/bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a built-in 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.

### SYSTIMESENSITIVE

Indicates whether references to system-period temporal tables in static and dynamic SQL statements are affected by the value of the CURRENT TEMPORAL SYSTEM\_TIME special register.

**YES** References to system-period temporal tables are affected by the value of the CURRENT TEMPORAL SYSTEM\_TIME special register. This value is the default value.

**NO** References to system-period temporal tables are not affected by the value of the CURRENT TEMPORAL SYSTEM\_TIME special register.

### TEXT *label*

The description of a package. Maximum length is 255 characters. The default value is blanks. This precompile/bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows.

### 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.

#### *groupname*

An SQL identifier of up to 18 bytes 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 previously listed 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:

```
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.

## BIND

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.

The privileges from the roles granted to the authorization identifier used to bind the package (the value of the **OWNER** bind option) or to `PUBLIC`, are taken into account when binding a package. Roles acquired through groups, in which the authorization identifier used to bind the package is a member, will not be used.

For an embedded SQL program, if the bind option is not explicitly specified the static statements in the package are bound using the **federated\_async** configuration parameter. If the **FEDERATED\_ASYNCHRONY** bind option is specified explicitly, that value is used for binding the packages and is also the initial value of the special register. Otherwise, the value of the database manager configuration

parameter is used as the initial value of the special register. The **FEDERATED\_ASYNC** bind option influences dynamic SQL only when it is explicitly set.

The value of the **FEDERATED\_ASYNC** bind option is recorded in the **FEDERATED\_ASYNC** column in the SYSCAT.PACKAGES catalog table. When the bind option is not explicitly specified, the value of **federated\_async** configuration parameter is used and the catalog shows a value of -2 for the **FEDERATED\_ASYNC** column.

If the **FEDERATED\_ASYNC** bind option is not explicitly specified when a package is bound, and if this package is implicitly or explicitly rebound, the package is rebound using the current value of the **federated\_async** configuration parameter.

## 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 database partition server.

### 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

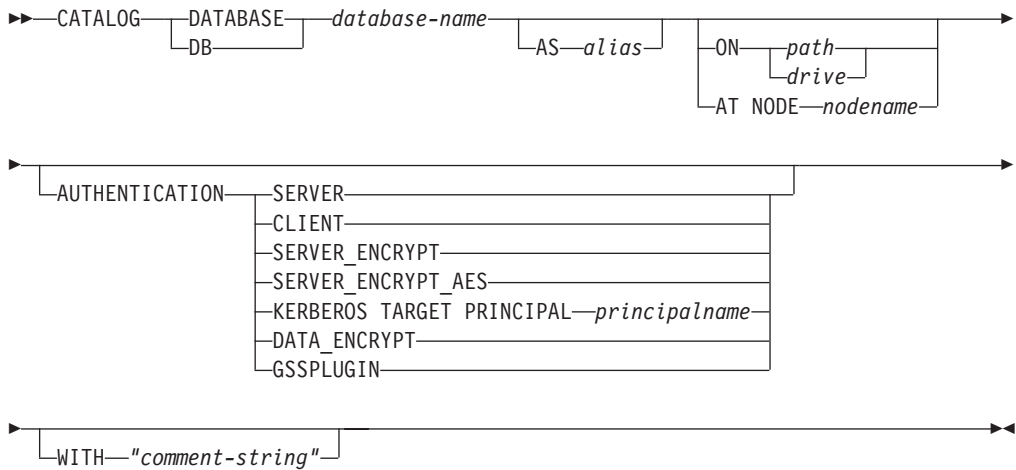
One of the following authorities:

- SYSADM
- SYCTRL

### 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 database partition server 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 required.

**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 database partition server containing the target database.

**CLIENT** Specifies that authentication takes place on the database partition server where the application is invoked.

**SERVER\_ENCRYPT**

Specifies that authentication takes place on the database partition server containing the target database, and that user IDs and passwords are encrypted at the source. User IDs and 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.

**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 database partition server 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.

**SERVER\_ENCRYPT\_AES**

Specifies that authentication takes place on the database partition server containing the target database, and that user IDs and passwords are encrypted with an Advanced Encryption Standard (AES) encryption algorithm at the source and decrypted at the target.

**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

## CATALOG DATABASE

carriage return or a line feed character is not permitted. The comment text must be enclosed by double quotation marks.

### Examples

```
db2 catalog database sample on /databases/sample
with "Sample Database"
```

### Usage notes

Use **CATALOG DATABASE** to catalog databases located on local or remote database partition servers, 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 database partition server 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 database partition server as the database manager instance are cataloged as *indirect* entries. Databases on other database partition servers 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, node and DCS directory files are cached in memory. To see if directory caching is enabled, check the value for the *dir\_cache* directory cache support configuration parameter in the output from 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 the database manager's shared cache, stop (**db2stop**) and then restart (**db2start**) the database manager. To refresh the directory cache for another application, stop and then restart that application.

## CATALOG DCS DATABASE

Stores information about remote host or System i 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

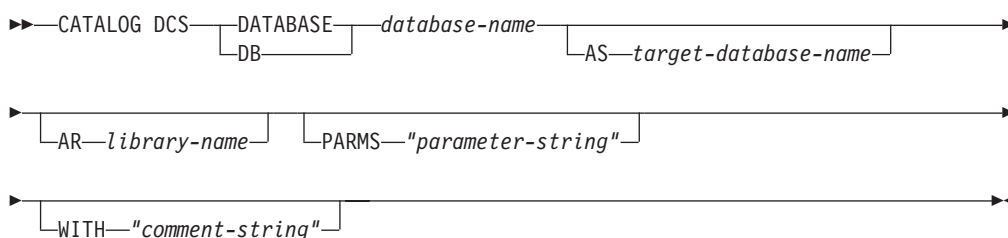
one of the following authorities:

- SYSADM
- SYSCTRL

### Required connection

None

### Command syntax



### 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 database partition server.

#### **AS** *target-database-name*

Specifies the name of the target host or System i 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 Linux and UNIX operating 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.

## 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<sup>®</sup> architecture host computers.
- DB2 for VM and VSE databases on System/370 and System/390 architecture host computers.
- System i databases on Application System/400<sup>®</sup> (System i) and System i 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 the shared cache in DB2, stop (**db2stop**) and then restart (**db2start**) the database manager. To refresh the directory cache for another application, stop and then restart that application.



## CATALOG LDAP DATABASE

Registers the database in Lightweight Directory Access Protocol (LDAP).

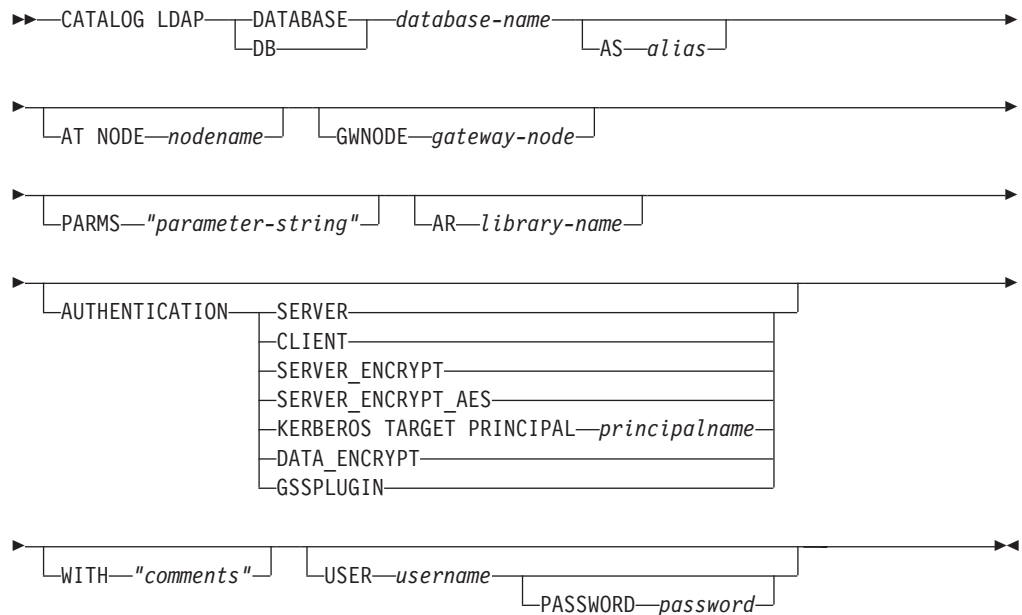
### Authorization

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* 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.

## CATALOG LDAP DATABASE

### **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\d11`. On UNIX operating systems, the path is `$HOME/sql1lib/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 database partition server containing the target database, and that user IDs and passwords are encrypted at the source. User IDs and passwords are decrypted at the target, as specified by the authentication type cataloged at the source.

### **SERVER\_ENCRYPT\_AES**

Specifies that authentication takes place on the database partition server containing the target database, and that user IDs and passwords are encrypted with an Advanced Encryption Standard (AES) encryption algorithm at the source and decrypted at the target.

### **KERBEROS**

Specifies that authentication takes place using Kerberos Security Mechanism.

### **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.

### **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 System i 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.

---

## 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.

## 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 authorities:

- SYSADM
- SYSCTRL

### Required connection

None

### Command syntax

```

▶▶ CATALOG [ADMIN] LOCAL NODE nodename [INSTANCE instancename]
▶ [SYSTEM system-name] [OSTYPE operating-system-type]
▶ [WITH "comment-string"]
  
```

### Command parameters

#### ADMIN

Specifies that a local administration server node is to be cataloged.

#### 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.

### 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
```

## CATALOG LOCAL NODE

### 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.

## 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 authorities:

- SYSADM
- SYSCTRL

### Required connection

None

### Command syntax

```

▶▶ CATALOG [ADMIN] NPIPE NODE nodename REMOTE computername
▶▶ INSTANCE instancename [SYSTEM system-name]
▶▶ [OSTYPE operating-system-type] [WITH "comment-string"]

```

### Command parameters

#### ADMIN

Specifies that an NPIPE administration server node is to be cataloged.

#### 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.

### 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,

## CATALOG NAMED PIPE NODE

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.



## 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.

#### DATA SOURCE *data-source-name*

Specifies the name of the data source to be cataloged. The name of the data source and the name of the database must be the same. Therefore, the name of the data source is limited to the maximum length for a database name.

#### ALL DATA SOURCES

Specifies to catalog all local database aliases as ODBC data sources (DSNs).

### Usage notes

On Microsoft Windows Vista or later versions, you must execute the **CATALOG SYSTEM ODBC DATA SOURCE** command from a DB2 command window running with full administrator privileges.

Specifying the **ALL DATA SOURCES** parameter will not update an existing ODBC DSN that has set its `dbalias` parameter to a value that matches the alias of a database in the local database directory

In Windows environments, the ODBC DSN must be restored after upgrading DB2. To import and export the ODBC DSN settings, use the **db2cfimp** command or the **db2cfexp** command, or add DSNs manually.

## CATALOG ODBC DATA SOURCE

### Example 1

Assume there is an existing ODBC DSN named "MyProdDatabase". The dbalias parameter is set to "PRODDB". Assume there is also a database in the local directory with the alias "PRODDB". Executing the **CATALOG ODBC DATA SOURCE myproddatabase** command or the **CATALOG ODBC ALL DATA SOURCES** command will not alter the "MyProdDatabase" DSN because the DSN does not match the database alias. Instead, an ODBC DSN entry is created for "PRODDB" with the dbalias set to "PRODDB". If there is an existing ODBC DSN with the same name as the database alias, the existing ODBC DSN's dbalias parameter will be updated with the database alias. All associated CLI parameters and values will remain unchanged.

### Example 2

Assume there is an existing DSN called "MYDB" that has the dbalias parameter set to "salesdb". If there is a database in the local directory named "MYDB" then executing the **CATALOG ODBC DATA SOURCE mydb** command or the **CATALOG ODBC ALL DATA SOURCES** command will change the DSN's dbalias parameter to "MYDB".

## CATALOG TCPIP/TCPIP4/TCPIP6 NODE

Adds a Transmission Control Protocol/Internet Protocol (TCP/IP) database partition server entry to the node directory. The TCP/IP communications protocol is used to access the remote database partition server. The **CATALOG TCPIP/TCPIP4/TCPIP6 NODE** command is run on a client.

**Note:** Starting with the DB2 for Linux, UNIX, and Windows Version 10.1 release, the AIX 5.3 operating system is not supported. DB2 for Linux, UNIX, and Windows Version 9.7 is the last release to support the AIX 5.3 operating system. The AIX 6.1 operating system is the minimum supported level.

### Authorization

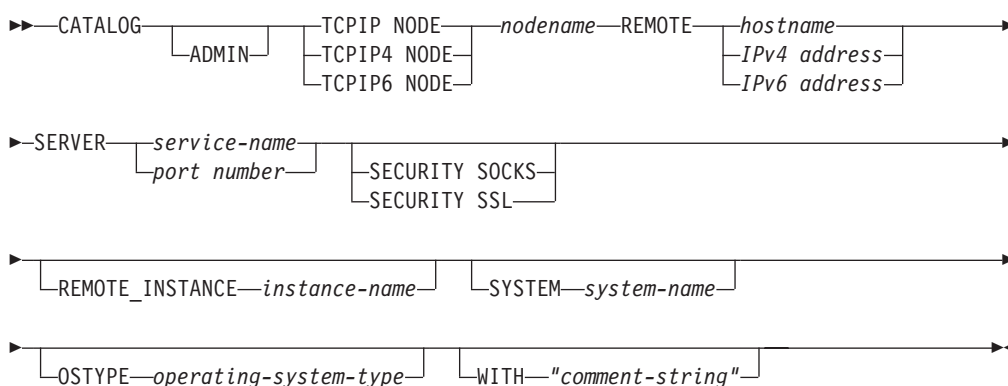
one of the following authorities:

- 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.

**TCPIP NODE** *nodename*

The nodename of the TCPIP, TCPIP4, or TCPIP6 database partition server represents a local nickname you can set for the machine that contains the database you want to catalog. Only specify **TCPIP4** when specifying an IPv4 IP address, and only specify **TCPIP6** when specifying an IPv6 IP address.

**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 database partition server that is known to the TCP/IP network. The maximum length of the hostname is 255 characters.

## CATALOG TCPIP/TCPIP4/TCPIP6 NODE

**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 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 **SECURITY 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 IBM Data Server 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.

### SECURITY SSL

Specifies that the node is SSL enabled. You cannot specify the **SECURITY SSL** clause if you also specify the **ADMIN** parameter.

**REMOTE\_INSTANCE** *instance-name*

Specifies the name of the server instance where the database resides, and 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 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.

### Examples

To specify a hostname using the **CATALOG TCP/IP NODE** command, issue:

```
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.

The following example catalogs a node for an SSL connection (the server hostname is *hostname*, and *ssl\_port* is the port number at which this database server waits for communication from remote client nodes using the SSL protocol):

```
db2 catalog tcpip node db2tcp4 remote hostname server ssl_port
```

### 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.

## CATALOG TCPIP/TCPIP4/TCPIP6 NODE

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.

To get the DB2 database manager to listen on IPv6, the operating system and server must first be configured for IPv6. Speak to your system administrator to ensure this configuration has been done before cataloging an IPv6 TCPIP node. Follow *Upgrading to IPv6 with IPv4 configured* to see how this can be done on AIX 6.1.

## 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 authorities:

- SYSADM
- SYSCTRL

### Required connection

None

### Command syntax

```

▶—CHANGE—┬—DATABASE—┬—database-alias—COMMENT—┬——┬——┬——▶
            │         │                             │   │   │
            └─DB—┘   └──────────────────────────┘   │   │
                                                       └─ON—┬—path—┘
                                                           │
                                                           └─drive—┘

▶—WITH—"comment-string"—▶
  
```

### Command parameters

#### DATABASE *database-alias*

Specifies the alias of the database whose comment is to be changed. To change the comment in the system database directory, specify the alias for the database. To change the comment in the local database directory, specify the path where the database resides (with the *path* parameter), and enter the name (not the alias) of the database.

#### ON *path* | *drive*

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.

## CHANGE DATABASE COMMENT

### Examples

The following example changes the text in the system database directory comment for the SAMPLE database from "Test 2 - Holding" to "Test 2 - Add employee inf rows":

```
db2 change database sample comment  
with "Test 2 - Add employee inf rows"
```

### Usage notes

New comment text replaces existing text. To append information, enter the old comment text, followed by the new text.

Only the comment for an entry associated with the database alias is modified. Other entries with the same database name, but with different aliases, are not affected.

If the path is specified, the database alias must be cataloged in the local database directory. If the path is not specified, the database alias must be cataloged in the system database directory.



## CHANGE ISOLATION LEVEL

Changes the way that DB2 isolates data from other processes while a database is being accessed.

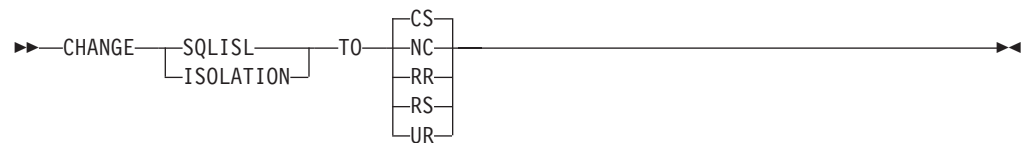
### Authorization

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:

## CHANGE ISOLATION LEVEL

```
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).

## COMPLETE XMLSCHEMA

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

```

▶▶ COMPLETE XMLSCHEMA relational-identifier
└──────────────────────────────────────────┘
└── WITH schema-properties-URI ───┘
└──────────────────────────────────────────┘
└── ENABLE DECOMPOSITION ───┘

```

### 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.POschema 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.

---

### CREATE DATABASE

Initializes a new database with an optional user-defined collating sequence, creates the three initial table spaces, creates the system tables, and allocates the recovery log file. When you initialize a new database, the **AUTOCONFIGURE** command is issued by default.

**Note:** When the instance and database directories are created by the DB2 database manager, the permissions are accurate and should not be changed.

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, refer to the **DB2\_ENABLE\_AUTOCONFIG\_DEFAULT** registry variable. To disable Automated Runstats, refer to the **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. For multi-partition databases, Adaptive Self Tuning Memory is disabled by default.

If no code set is specified on the **CREATE DATABASE** command, then the collations allowed are: SYSTEM, IDENTITY\_16BIT, *language-aware-collation*, and *locale-sensitive-collation* (SQLCODE -1083). The default code set for a database is UTF-8. If a particular code set and territory is needed for a database, the required 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

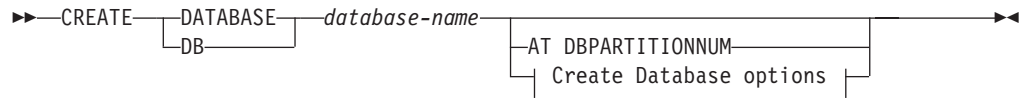
You must have one of the following authorities:

- SYSADM
- SYSCTRL

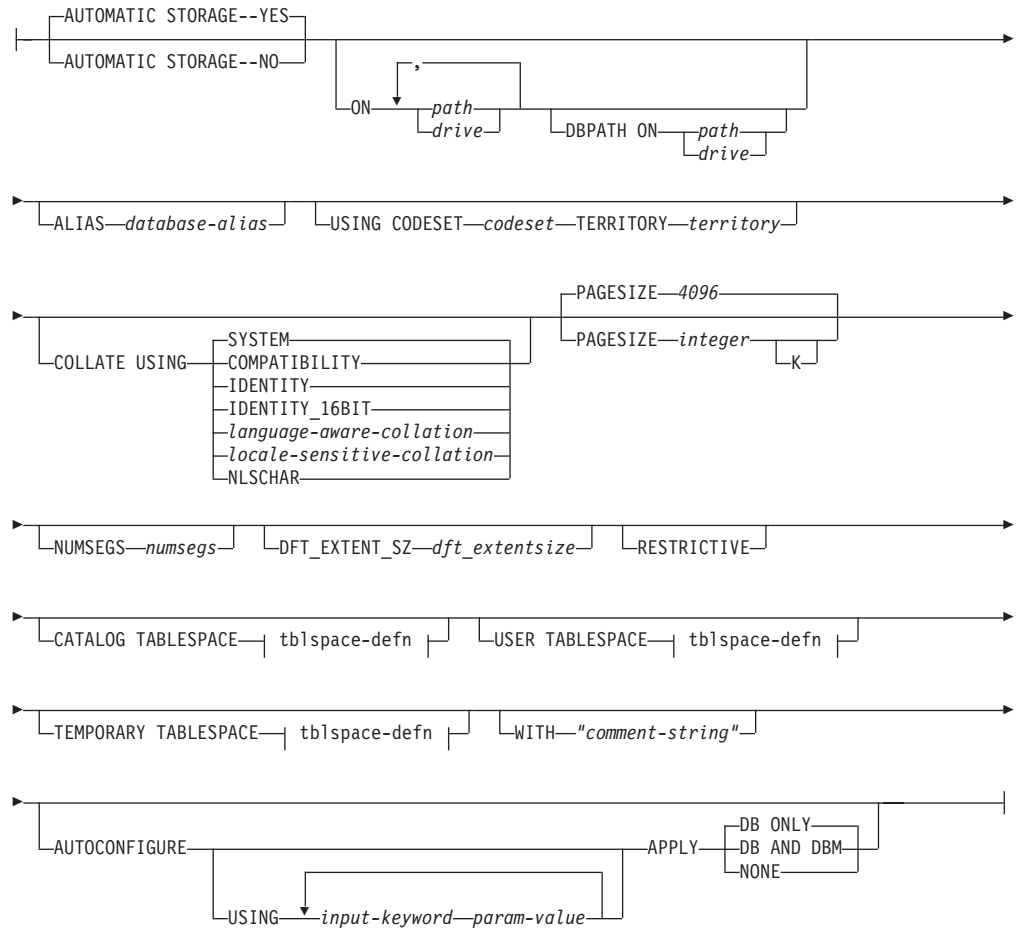
#### Required connection

Instance. To create a database at another (remote) database partition server, you must first attach to that server. A database connection is temporarily established by this command during processing.

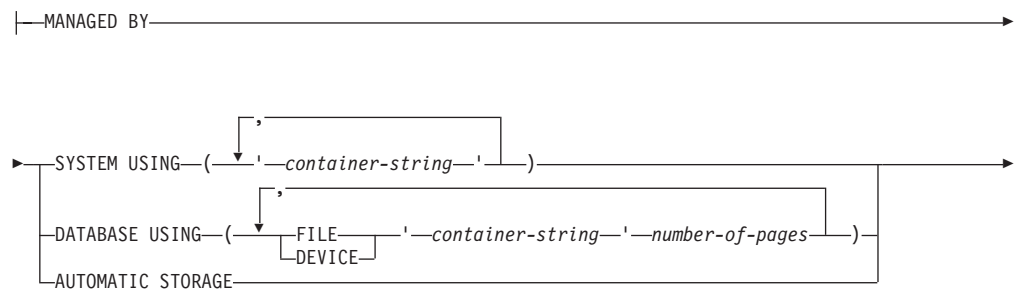
Command syntax



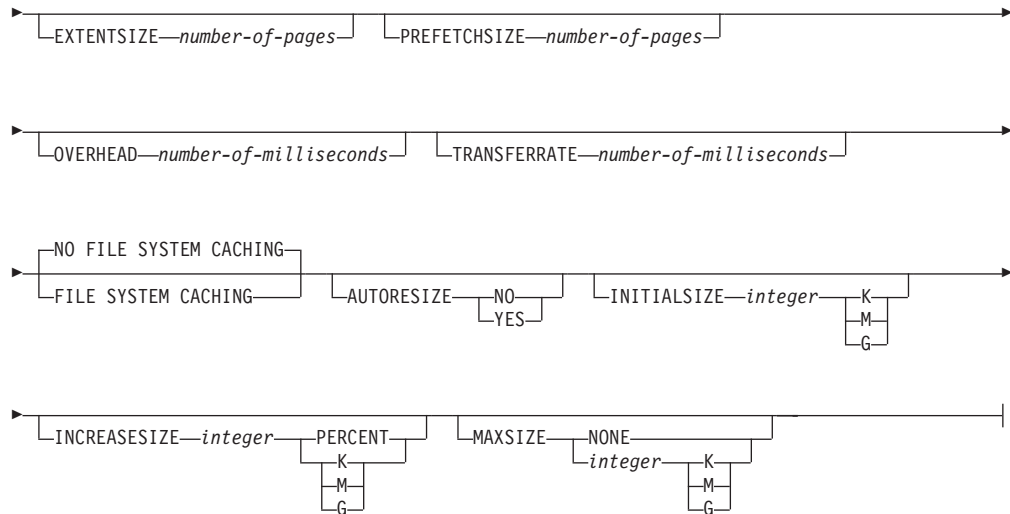
Create Database options:



tblspace-defn:



## CREATE DATABASE



### Note:

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** parameter 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 parameter when you create a new database. You can use it to re-create a database partition that you dropped because it was damaged. After you use the **CREATE DATABASE** command with the **AT DBPARTITIONNUM** parameter, the database at this database partition is in the restore-pending state. You must immediately restore the database on this database partition server. This parameter is not intended for general use. For example, it should be used with **RESTORE DATABASE** command if the database partition at a database partition server 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 re-create 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 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. This parameter cannot be specified in a DB2 pureScale environment.

**YES** Automatic storage is being enabled for the database. The default storage group, IBMSTOGROUP, is created in the SYSSTOGROUPS catalog table. To modify a storage group, use the ALTER STOGROUP statement.

**Important:** This parameter has been deprecated and might be removed in a future release. Once removed, AUTOMATIC STORAGE YES will be the only option.

#### **ON path or drive**

The meaning of this parameter depends on the value of the **AUTOMATIC STORAGE** parameter.

- 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** parameter, 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).
- 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 defined to the default storage group IBMSTOGROUP 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.

The **DBPATH ON** parameter specifies on which paths to create the database. If the **DBPATH ON** parameter is not specified, the database is created on the first path listed in the **ON** parameter. If no paths are specified with the **ON** parameter, 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 default storage group.

The database path is the location where a hierarchical directory structure is created. The structure holds the following files needed for the operation of the database:

- Buffer pool information
- Table space information
- Storage path information
- Database configuration information
- History file information regarding backups, restores, loading of tables, reorganization of tables, altering of table spaces, and other database changes
- Log control files with information about active logs

## CREATE DATABASE

The **DBPATH ON** parameter can be used to place these files and information in a directory that is separate from the storage paths where the database data is kept. It is suggested that the **DBPATH ON** parameter be used when automatic storage is enabled to keep the database information separate from the database data.

The maximum length of a path is 175 characters.

For a partitioned database environment, 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 operating systems, it should not specify the \$HOME directory of the instance owner). The path specified for this command in a partitioned database environment cannot be a relative path. Also, all paths specified as part of the **ON** parameter 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** parameter specifies the path on which to create the database. If automatic storage is enabled and the **DBPATH ON** parameter is not specified, the database is created on the first path listed with the **ON** parameter.

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.

### **TERRITORY** *territory*

Specifies the territory identifier or locale identifier to be used for data entered into this database. After you create the database, you cannot change the specified territory. The combination of the code set and territory or locale values must be valid.

### **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.

In a Unicode database, the catalog tables and views are always created with the **IDENTITY** collation, regardless of the collation specified in the **COLLATE USING** clause. In non-Unicode databases, the catalog tables and views are created with the database collation.

### **COMPATIBILITY**

The DB2 Version 2 collating sequence. Some collation tables have been enhanced. This parameter specifies that the previous version of these tables is to be used.

### **IDENTITY**

Identity collating sequence, in which strings are compared byte for byte.



**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 website ([www.unicode.org](http://www.unicode.org)). This parameter can only be specified when creating a Unicode database.

*language-aware-collation*

This parameter can only be used for Unicode databases. The database collating sequence is based on the SYSTEM collation for a non-Unicode database. This string must be of the format `SYSTEM_codepage_territory`. If the string supplied is invalid, the create database will fail (SQLCODE -204; object not found). See “Language-aware collations for Unicode data” for more information and for the naming of system based collations.

**Note:** When the **CREATE DATABASE** command is performed against a Version 9.0 server, this parameter cannot be used. By default, a Unicode database on such a server will be created with SYSTEM collation.

*locale-sensitive-collation*

This parameter can only be used for Unicode databases. See “Unicode Collation Algorithm based collations” for more information and for the naming of locale-sensitive UCA-based collations. If the collation name provided is invalid, the **CREATE DATABASE** command execution will fail (SQLCODE -204).

**NLSCHAR**

Built-in collating sequence using the unique collation rules for the specific code set/territory.

This parameter can only be used with the Thai code page (CP874). If this parameter is specified in non-Thai environments, the command will fail and return the error SQL1083N with Reason Code 4.

**SYSTEM** This is the default parameter when creating a database. For non-Unicode databases, the collating sequence is based on the database territory. For Unicode databases, this parameter maps to a language-aware collation, based on the client code set and territory. If an appropriate language-aware collation is unavailable, then the IDENTITY collation is used.

**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 (table space 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

## CREATE DATABASE

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** parameter is present it causes the **restrict\_access** database configuration parameter to be set to YES and no privileges or authorities are automatically granted to PUBLIC. If the **RESTRICTIVE** parameter is not present then the **restrict\_access** database configuration parameter is set to NO and privileges are automatically granted to PUBLIC. For information about privileges, see: "Default privileges granted on creating a database".

### **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 "CREATE TABLESPACE statement" for more information about 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 “CREATE TABLESPACE statement” for more information about 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**.

See “CREATE TABLESPACE statement” for more information about the table space definition fields.

#### *tblspace-defn*

Various table space definitions can be specified through the following command parameters. In a DB2 pureScale environment, only **MANAGED BY AUTOMATIC STORAGE** can be used.

#### MANAGED BY

##### SYSTEM USING *container-string*

Specifies that the table space is to be an SMS table space. When the type of table space is not specified, the default behavior is to create a regular table space.

For an SMS table space, identifies one or more containers that will belong to the table space and in which the table space data will be stored. The *container-string* cannot exceed 240 bytes in length.

Each *container-string* can be an absolute or relative directory name.

The directory name, if not absolute, is relative to the database directory, and can be a path name alias (a symbolic link on UNIX operating systems) to storage that is not physically associated with the database directory. For example, *dbdir/work/c1* could be a symbolic link to a separate file system.

If any component of the directory name does not exist, it is created by the database manager. When a table space is dropped, all components created by the database manager are deleted. If the directory identified by *container-string* exists, it must not contain any files or subdirectories (SQLSTATE 428B2).

The format of *container-string* is dependent on the operating system. On Windows operating systems, an absolute directory path name begins with a drive letter and

## CREATE DATABASE

a colon (:); on UNIX operating systems, an absolute path name begins with a forward slash (/). A relative path name on any platform does not begin with an operating system-dependent character.

Remote resources (such as LAN-redirectioned drives or NFS-mounted file systems) are currently only supported when using Network Appliance Filers, IBM iSCSI, IBM Network Attached Storage, Network Appliance iSCSI, NEC iStorage S2100, S2200, or S4100, or NEC Storage NS Series with a Windows DB2 server. Note that NEC Storage NS Series is only supported with the use of an uninterrupted power supply (UPS); continuous UPS (rather than standby) is recommended. An NFS-mounted file system on AIX must be mounted in uninterruptible mode using the `-o nointr` parameter.

**Important:** For **USER TABLESPACE, MANAGED BY SYSTEM** is deprecated and might be removed in a future release. Use **MANAGE BY AUTOMATIC STORAGE** or **MANAGE BY DATABASE** instead.

### DATABASE USING

Specifies that the table space is to be a DMS table space. When the type of table space is not specified, the default behavior is to create a large table space.

For a DMS table space, identifies one or more containers that will belong to the table space and in which the table space data will be stored. The type of the container (either **FILE** or **DEVICE**) and its size (in **PAGESIZE** pages) are specified. A mixture of **FILE** and **DEVICE** containers can be specified. The *container-string* cannot exceed 254 bytes in length.

Remote resources (such as LAN-redirectioned drives or NFS-mounted file systems) are currently only supported when using Network Appliance Filers, IBM iSCSI, IBM Network Attached Storage, Network Appliance iSCSI, NEC iStorage S2100, S2200, or S4100, or NEC Storage NS Series with a Windows DB2 server. Note that NEC Storage NS Series is only supported with the use of an uninterrupted power supply (UPS); continuous UPS (rather than standby) is recommended.

All containers must be unique across all databases. A container can belong to only one table space. The size of the containers can differ; however, optimal performance is achieved when all containers are the same size. The exact format of *container-string* is dependent on the operating system.

**FILE** *container-string number-of-pages*

For a **FILE** container, *container-string* must be an absolute or relative file name. The file name, if not absolute, is relative to the database directory. If any component of the directory name does not exist, it is created by the database manager. If the file does

not exist, it will be created and initialized to the specified size by the database manager. When a table space is dropped, all components created by the database manager are deleted.

**Note:** If the file exists, it is overwritten, and if it is smaller than specified, it is extended. The file will not be truncated if it is larger than specified.

**DEVICE** *container-string number-of-pages*

For a DEVICE container, *container-string* must be a device name. The device must already exist.

#### **AUTOMATIC STORAGE**

Specifies that the table space is to be an automatic storage table space. If there are no storage groups defined, an error is returned (SQLSTATE 55060).

An automatic storage table space is created as a system managed space (SMS) table space if it is a temporary table space and as a database managed space (DMS) table space if it is a permanent table space. If the type of DMS table space is not specified, the default behavior is to create a large table space. With an automatic storage table space, the database manager determines which containers are to be assigned to the table space, based upon the storage paths that are associated with the database.

**EXTENTSIZE** *number-of-pages*

Specifies the number of **PAGESIZE** pages that will be written to a container before skipping to the next container. The extent size value can also be specified as an integer value followed by K (for kilobytes) or M (for megabytes). If specified in this way, the floor of the number of bytes divided by the page size is used to determine the value for the extent size. The database manager cycles repeatedly through the containers as data is stored.

The default value is provided by the **dft\_extent\_sz** database configuration parameter, which has a valid range of 2-256 pages.

**PREFETCHSIZE** *number-of-pages*

Specifies the number of **PAGESIZE** pages that will be read from the table space when data prefetching is being performed. The prefetch size value can also be specified as an integer value followed by K (for kilobytes), M (for megabytes), or G (for gigabytes). If specified in this way, the floor of the number of bytes divided by the page size is used to determine the number of pages value for prefetch size.

**OVERHEAD** *number-of-milliseconds*

Number that specifies the I/O controller usage, disk seek, and latency time in milliseconds. This value is used to determine the cost of I/O during query optimization. The value of *number-of-milliseconds* is any numeric literal (integer, decimal, or floating point). If this value is not the

## CREATE DATABASE

same for all containers, the number should be the average for all containers that belong to the table space.

For a database that was created in Version 9 or later, the default I/O controller usage and disk seek and latency time is 7.5 milliseconds. For a database that was upgraded from a previous version of DB2 to Version 9 or later, the default is 12.67 milliseconds.

### **TRANSFERRATE** *number-of-milliseconds*

Specifies the time to read one page into memory. This value is used to determine the cost of I/O during query optimization. The value of *number-of-milliseconds* is any numeric literal (integer, decimal, or floating point). If this value is not the same for all containers, the number should be the average for all containers that belong to the table space.

For a database that was created in Version 9 or later, the default time to read one page into memory is 0.06 milliseconds. For a database that was upgraded from a previous version of DB2 to Version 9 or later, the default is 0.18 milliseconds.

### **NO FILE SYSTEM CACHING**

Specifies that all I/O operations are to bypass the file system-level cache. See "Table spaces without file system caching" for more details. This is the default parameter on most configurations. See "File system caching configurations" for details.

### **FILE SYSTEM CACHING**

Specifies that all I/O operations in the target table space are to be cached at the file system level. See "Table spaces without file system caching" for more details. This is the default parameter on some configurations. See "File system caching configurations" for details.

### **AUTORESIZE**

Specifies whether or not the auto-resize capability of a DMS table space or an automatic storage table space is to be enabled. Auto-resizable table spaces automatically increase in size when they become full. The default is NO for DMS table spaces and YES for automatic storage table spaces.

**NO** Specifies that the auto-resize capability of a DMS table space or an automatic storage table space is to be disabled.

**YES** Specifies that the auto-resize capability of a DMS table space or an automatic storage table space is to be enabled.

### **INITIALSIZE** *integer*

Specifies the initial size, per database partition, of an automatic storage table space. This parameter is only valid for automatic storage table spaces. The integer value must be followed by K (for kilobytes), M (for megabytes), or G (for gigabytes). Note that the actual value used might be

slightly smaller than what was specified, because the database manager strives to maintain a consistent size across containers in the table space. Moreover, if the table space is auto-resizable and the initial size is not large enough to contain meta-data that must be added to the new table space, the database manager will continue to extend the table space by the value of **INCREASESIZE** until there is enough space. If the **INITIALSIZE** clause is not specified, the database manager determines an appropriate value. The value for *integer* must be at least 48 K.

**K** K (for kilobytes).

**M** M (for megabytes).

**G** G (for gigabytes).

#### **INCREASESIZE** *integer*

Specifies the amount, per database partition, by which a table space that is enabled for auto-resize will automatically be increased when the table space is full, and a request for space has been made. The integer value must be followed by either:

- PERCENT to specify the amount as a percentage of the table space size at the time that a request for space is made. When PERCENT is specified, the integer value must be between 0 and 100 (SQLSTATE 42615).
- K (for kilobytes), M (for megabytes), or G (for gigabytes) to specify the amount in bytes

Note that the actual value used might be slightly smaller or larger than what was specified, because the database manager strives to maintain consistent growth across containers in the table space. If the table space is auto-resizable, but the **INCREASESIZE** clause is not specified, the database manager determines an appropriate value.

#### **PERCENT**

Percent from 0 to 100.

**K** K (for kilobytes).

**M** M (for megabytes).

**G** G (for gigabytes).

#### **MAXSIZE**

Specifies the maximum size to which a table space that is enabled for auto-resize can automatically be increased. If the table space is auto-resizable, but the **MAXSIZE** clause is not specified, the default is NONE.

**NONE** Specifies that the table space is to be allowed to grow to file system capacity, or to the maximum table space size.

*integer* Specifies a hard limit on the size, per database partition, to which a DMS table space or an automatic storage table space can automatically be increased. The integer value must be followed by K (for kilobytes), M (for megabytes), or G (for

## CREATE DATABASE

gigabytes). Note that the actual value used might be slightly smaller than what was specified, because the database manager strives to maintain consistent growth across containers in the table space.

**K** K (for kilobytes).

**M** M (for megabytes).

**G** G (for gigabytes).

### **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** parameter 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 instance memory that is assigned to the database. However, if the <b>CREATE DATABASE</b> command invokes the configuration advisor and you do not specify a value for <b>mem_percent</b> , the percentage is calculated based on memory usage in the instance and the system up to a maximum of 25% of the instance memory.
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



Table 9. Valid input keywords and parameter values (continued)

Keyword	Valid values	Default value	Explanation
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)
bp_resizeable	yes, no	yes	Are buffer pools resizable?

**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** parameters.
- Specifying the **AUTOCONFIGURE** parameter 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.

## CREATE DATABASE

### Examples

Here are several examples of the **CREATE DATABASE** command:

Example 1:

```
CREATE DATABASE TESTDB3
```

Database TESTDB3 is created on the drive that is the value of database manager configuration parameter **dftdbpath**. By default, the storage group IBMSTOGROUP is created with the path **dftdbpath**.

Example 2:

```
CREATE DATABASE TESTDB7 ON C:,D:
```

Database TESTDB7 is created on drive C: (first drive in storage path list). The storage group IBMSTOGROUP has storage paths C: and D:.

Example 3:

```
CREATE DATABASE TESTDB15  
ON C:,D: DBPATH ON E:
```

Database TESTDB15 is created on drive E: (explicitly listed as **DBPATH**). The storage group IBMSTOGROUP has storage paths C: and D:.

### 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.  
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.

**Note:** If the change the database by path configuration parameter **newlogpath** is not set, the default for the location of log files configuration parameter **logpath** is the path shown by the **DBPATH ON** parameter. It is suggested that the **DBPATH ON** parameter be used when automatic storage is enabled to keep the database information separate from the database data.

- 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, if the **RESTRICTIVE** parameter is not selected.

The utilities bind file list contains two bind files that cannot be bound against previous version of the server:

- 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 server which is not at the latest level, 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.
- For information about privileges granted when creating a database, see: “Default privileges granted on creating a database”.

**Note:** If the **RESTRICTIVE** parameter is present it causes the **restrict\_access** database configuration parameter to be set to YES and no privileges or authorities are automatically granted to PUBLIC. For additional information, see the RESTRICTIVE parameter of the CREATE DATABASE command.

Automatic storage is a collection of storage paths associated with a storage group 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** parameter.

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.

## CREATE DATABASE

- *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 want 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 before 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.

In a partitioned database 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 SQL*nnnnn* 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 additional processing usage 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.

The combination of the code set and territory values must be valid. For a list of the supported combinations, see "Supported territory codes and code pages".

To specify a database path (instead of a drive) on a Windows operating system, you need to set the DB2 registry variable: DB2\_CREATE\_DB\_ON\_PATH=YES.

Use the COLLATE USING clause with a language-aware-collation or locale-sensitive-collation instead of UCA400\_NO, UCA400\_LSK, or UCA400\_LTH.

**Important:** Collations based on the Unicode Collation Algorithm of the Unicode Standard version 4.0.0 have been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Collations based on the Unicode Collation Algorithm of the Unicode Standard version 4.0.0 have been deprecated” in *What’s New for DB2 Version 10.1*.

**Related information:**

### CREATE TOOLS CATALOG

Creates the DB2 tools catalog tables in a new or existing database. The tools catalog contains information about the administrative tasks that are available to you.

The database must be local.

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 IBM Data Server Client.

#### Scope

The node from which this command is issued becomes the catalog node for the new database.

#### Authorization

One of the following authorities:

- 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—|—————|—————▶
  |—————|—————|—————|—————|—————▶
▶—[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 catalog1 use existing database toolsdb force
```

```
db2 create tools catalog catalog1 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).
- In partitioned database environments, the 32 KB REGULAR table space must exist on the catalog partition, otherwise the command (such as the following one) will fail when a table space is specified:

```
db2 create tools catalog catalog1 use existing tablespace user32Ksp
in database toolsdb
```

---

## DEACTIVATE DATABASE

Deactivates the specified database and stops all necessary database services.

### Scope

This command deactivates the target database on all members in the instance or, a specified member. If one or more of these members encounters an error during deactivation, a warning is returned. The database remains activated on those members.

### Authorization

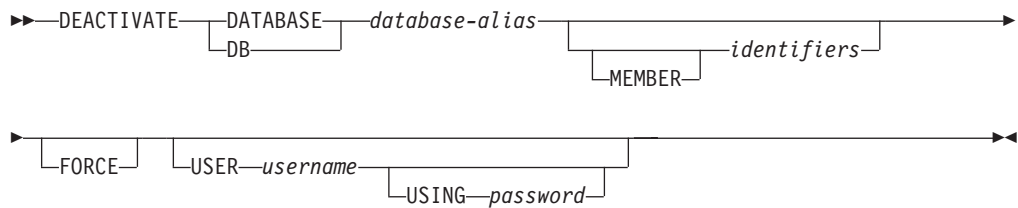
One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT

### Required connection

None

### Command syntax



### Command parameters

**DATABASE | DB** *database-alias*

Specifies the alias of the database to be stopped.

**MEMBER**

Specifies one or more members on which to deactivate the database.

*identifiers*

Specifies the numeric identifier of one or more members on which to deactivate the database. You can use the numeric identifier without specifying the **MEMBER** parameter keyword.

**FORCE**

Forces a database to deactivate even if in-doubt transactions exist.

**Note:** An error is returned if you try to deactivate a database without using the **FORCE** option with existing in-doubt transactions.

**USER** *username*

Specifies the user stopping the database.

**USING** *password*

Specifies the password for the user name.



### Usage notes

- An application issuing the **DEACTIVATE DATABASE** command cannot have an active database connection to any database.
- Databases initialized by the **ACTIVATE DATABASE** command can be shut down by issuing the **DEACTIVATE DATABASE** or **db2stop** command. If a database is initialized by the **ACTIVATE DATABASE** command, the last application disconnecting from the database does not shut down the database, and the **DEACTIVATE DATABASE** command must be used.
- In a DB2 pureScale environment, an active database on a member can only be deactivated on that member by issuing the **DEACTIVATE DATABASE** or **db2stop** command.

### Compatibilities

For compatibility with previous versions:

- **DBPARTITIONNUM** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.

**Related information:**

---

## DECOMPOSE XML DOCUMENT

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:

- One of the following authorizations:
  - CONTROL privilege on all target tables referenced in the set of annotated schema documents
  - DATAACCESS authority
- 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 privilege, as applicable, on any table referenced by the db2-xdb:expression or db2-xdb:condition annotation

If the **VALIDATE** option is specified, USAGE privilege on the XML schema is also required.

### 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
```

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"
```

## DECOMPOSE XML DOCUMENTS

Decomposes XML documents stored in a database column. The data from the XML documents is stored in columns of relational tables based on annotations specified in an XML schema.

The DECOMPOSE XML DOCUMENTS command invokes the XDB\_DECOMP\_XML\_FROM\_QUERY stored procedure to decompose one or more XML documents from a binary or XML column using a registered and decomposition-enabled XML schema.

### Authorization

One of the following groups of privileges is required:

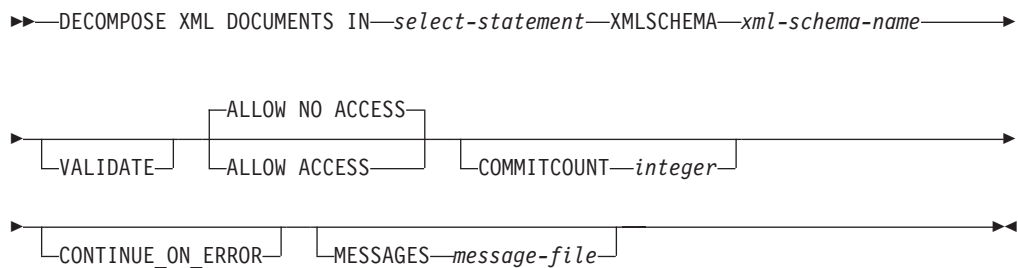
- All of the following privileges:
  - INSERT privilege on all target tables referenced in the annotated schema
  - SELECT privilege on the table, alias, or view containing the column holding the input documents
  - SELECT, INSERT, UPDATE, or DELETE privilege, as applicable, on any table referenced by the db2-xdb:expression or db2-xdb:condition annotation
- One of the following authorizations:
  - CONTROL privilege on all tables referenced in the set of annotated schema documents and on the table, alias, or view containing the column holding the input documents
  - DATAACCESS authority

If the **VALIDATE** option is specified, USAGE privilege on the XML schema is also required.

### Required connection

Database

### Command syntax



### Command parameters

#### DECOMPOSE XML DOCUMENTS IN *select-statement*

The *select-statement* conforms to the rules of an SQL SELECT statement, and must return a result set containing 2 columns. The first column is the document identifier. Each document identifier uniquely identifies an XML document to be decomposed. The column must be of character type or be castable to character type. The second column contains the XML documents to

be decomposed. The supported types for the document column are XML, BLOB, VARCHAR FOR BIT DATA, and LONG VARCHAR FOR BIT DATA. The column containing the XML documents must resolve to a column of an underlying base table, the column cannot be a generated column.

For example, the DOCID column in the following SELECT statement contains the unique identifiers for the XML documents stored in SALESDOC column.

```
SELECT DOCID, SALESDOC FROM SALESTAB
```

### **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**

Specifies that each input XML document is to be validated against *xml-schema-name*, then decomposed if the document is valid. If **VALIDATE** is not specified, input XML documents are not validated before decomposition.

If **VALIDATE** is not specified, it is the user's responsibility to validate the documents before calling the command. For example, the user can use XMLVALIDATE when inserting the XML documents into the column, or use an XML processor before inserting the documents. If an input XML document is not valid and **VALIDATE** is not specified, the decomposition results are undefined. See the related reference at the end of this topic for information about XML validation.

### **ALLOW**

Specifies whether access to the target tables specified in the XML Schema *xml-schema-name* are allowed during decomposition. ALLOW NO ACCESS is the default value.

#### **ALLOW ACCESS**

If ALLOW ACCESS is specified, when acquiring locks on the target table, the DECOMPOSE operation will wait and possibly timeout.

#### **ALLOW NO ACCESS**

If ALLOW NO ACCESS specified or used as the default value, the DECOMPOSE operation will acquire an exclusive lock (X) on all tables which have mappings specified in the XML schema. Not all target tables will necessarily participate during the decomposition of each document, but all target tables will be locked to lower the possibility of deadlock during a long unit of work.

### **COMMITCOUNT** *integer*

Specifies that after every *integer* successful document decompositions, a COMMIT is performed. A value of 0, or if the option is not specified, means that no COMMIT will ever be performed by the DECOMPOSE operation.

### **CONTINUE\_ON\_ERROR**

Specifies that the DECOMPOSE operation continues to the next document if a document-specific error occurs. Any changes to the database caused by an unsuccessfully decomposed document is undone before proceeding to the next document. If CONTINUE\_ON\_ERROR is not specified, the DECOMPOSE operation stops on the first document that cannot be successfully decomposed.

## DECOMPOSE XML DOCUMENTS

The DECOMPOSE operation does not continue on fatal errors and non-document specific errors even if the CONTINUE\_ON\_ERROR option is specified.

### MESSAGES *message-file*

The DECOMPOSE operation generates a UTF-8 encoded XML document that lists the input XML documents that were not successfully decomposed, along with the reason for their failure. The document containing the decomposition errors is generated only if there is at least one XML document that could not be successfully decomposed. Messages are translated according to server locale. *message-file* is the file that contains the XML document containing the decomposition information. If *message-file* is specified the file will be created on the system from where the CLP command is invoked. If the complete path of the file is not specified, it will be created in the current directory.

If this option is not specified, the decomposition information will be written to standard output.

Information about the decomposition of XML documents is displayed as an XML document that can optionally be sent to *message-file* specified by the parameter **MESSAGES**. The format of the XML document in *message-file* is as follows:

```
<?xml version='1.0' xmlns:xdb="http://www.ibm.com/xmlns/prod/db2/xdb1"?>
<xdb:errorReport>
  <xdb:document>
    <xdb:documentId>sssss</xdb:documentId>
    <xdb:errorMsg>qqqqq</xdb:errorMsg>
  </xdb:document>
  <xdb:document>
    .
    .
    .
  </xdb:document>
  .
  .
  .
</xdb:errorReport>
```

The documentId value *sssss* is the value from the first column specified by *select-statement*. The value identifies the XML document that was not successfully decomposed. The errorMsg value *qqqqq* is the error encountered during the attempt to decompose the document.

### Example

You could insert XML documents to be decomposed into a relational table, for example: ABC.SALESTAB. All the documents correspond to an XML schema registered as ABC.SALES, and the schema has been annotated with decomposition information and enabled for decomposition. Assuming the column name into which the documents are inserted is SALESDOC, and the corresponding ID is inserted into DOCID, invoke the DECOMPOSE XML DOCUMENTS command as follows:

```
DECOMPOSE XML DOCUMENTS IN 'SELECT DOCID, SALESDOC FROM SALESTAB'
XMLSCHEMA ABC.SALES
MESSAGES /home/myid/errors/errorreport.xml
```

## DEREGISTER

Deregisters the DB2 server from the network directory server.

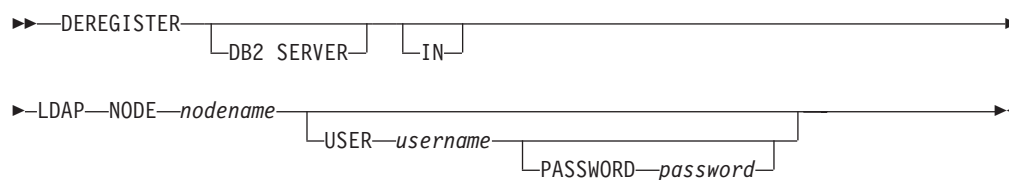
### Authorization

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** *nodename*

The node name is the value that was specified when the DB2 server was registered in LDAP.

### Usage notes

This command can only be issued for a remote machine when in the LDAP environment. When issued for a remote machine, the node name of the remote server must be specified.

The DB2 server is automatically deregistered when the instance is dropped.

---

## DESCRIBE

The **DESCRIBE** command displays metadata about the columns, indexes, and data partitions of tables or views. This command can also display metadata about the output of SELECT, CALL, or XQuery statements.

Use the **DESCRIBE** command to display information about any of the following items:

- Output of a SELECT, CALL, or XQuery statement
- Columns of a table or a view
- Indexes of a table or a view
- Data partitions of a table or view

### Authorization

The authorization required depends on the type of information you want to display using the **DESCRIBE** command.

- If the SYSTOOLSTMPSPACE table space exists, one of the authorities shown in the following table is required.

Object to display information about	Privileges or authorities required
Output of a SELECT statement or XQuery statement	Any of the following privileges or authorities for each table or view referenced in the SELECT statement: <ul style="list-style-type: none"><li>• SELECT privilege</li><li>• DATAACCESS authority</li><li>• DBADM authority</li><li>• SQLADM authority</li><li>• EXPLAIN authority</li></ul>
Output of a CALL statement	Any of the following privileges or authorities: <ul style="list-style-type: none"><li>• DATAACCESS authority</li><li>• EXECUTE privilege on the stored procedure</li></ul>



Object to display information about	Privileges or authorities required
Columns of a table or a view	<p>Any of the following privileges or authorities for the SYSCAT.COLUMNS system catalog table:</p> <ul style="list-style-type: none"> <li>• SELECT privilege</li> <li>• ACCESSCTRL authority</li> <li>• DATAACCESS authority</li> <li>• DBADM authority</li> <li>• SECADM authority</li> <li>• SQLADM authority</li> </ul> <p>If you want to use the <b>SHOW DETAIL</b> parameter, you also require any of these privileges or authorities on the SYSCAT.DATAPARTITIONEXPRESSION system catalog table.</p> <p>Because PUBLIC has all the privileges over declared temporary tables, you can use the command to display information about any declared temporary table that exists within your connection.</p>
Indexes of a table or a view	<p>Any of the following privileges or authorities on the SYSCAT.INDEXES system catalog table:</p> <ul style="list-style-type: none"> <li>• SELECT privilege</li> <li>• ACCESSCTRL authority</li> <li>• DATAACCESS authority</li> <li>• DBADM authority</li> <li>• SECADM authority</li> <li>• SQLADM authority</li> </ul> <p>If you want to use the <b>SHOW DETAIL</b> parameter, you also require EXECUTE privilege on the GET_INDEX_COLNAMES() UDF.</p> <p>Because PUBLIC has all the privileges over declared temporary tables, you can use the command to display information about any declared temporary table that exists within your connection.</p>

## DESCRIBE

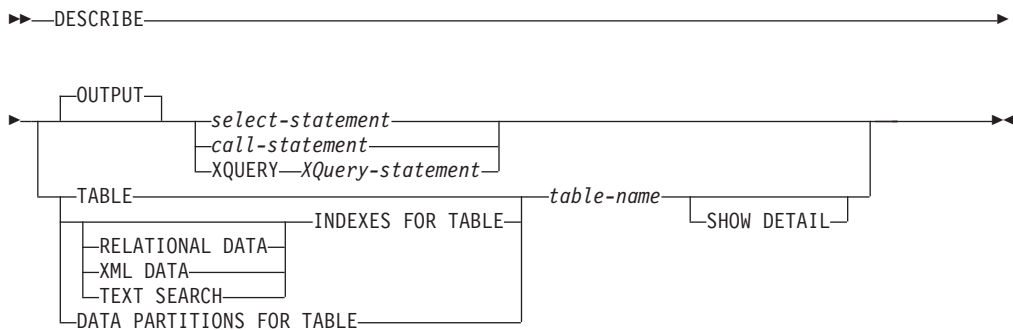
Object to display information about	Privileges or authorities required
Data partitions of a table or view	<p>Any of the following privileges or authorities on the SYSCAT.DATAPARTITIONS system catalog table:</p> <ul style="list-style-type: none"> <li>• SELECT privilege</li> <li>• ACCESSCTRL authority</li> <li>• DATAACCESS authority</li> <li>• DBADM authority</li> <li>• SECADM authority</li> <li>• SQLADM authority</li> </ul> <p>Because PUBLIC has all the privileges over declared temporary tables, you can use the command to display information about any declared temporary table that exists within your connection.</p>

- If the SYSTOOLSTMPSPACE table space does not exist, SYSADM or SYSCtrl authority is also required in addition to the one of the previously listed authorities.

### 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.

*select-statement* | *call-statement* | **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**. A DESCRIBE OUTPUT statement only returns information about an implicitly hidden column if the column is explicitly specified as part of the SELECT list of the final result table of the query described.

**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. Information about implicitly hidden columns is returned, but SHOW DETAIL must be used to indicate which columns are implicitly hidden.

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. You can use the fully qualified name in the form *schema.table-name* or you can just specify the *table-name* and default schema will be used automatically. An alias for the table cannot be used in place of the actual table.

The **DESCRIBE INDEXES FOR TABLE** command lists the following information about each index of the table or view:

- Index schema
- Index name
- Unique rule
- Number of columns
- Index type

If the **DESCRIBE INDEXES FOR TABLE** command is specified with the **SHOW DETAIL** option, the index name is truncated when the index name is greater than 18 bytes. If no index type option is specified, information for all index types is listed: relational data index, index over XML data, and Text Search index. The output includes the following additional information:

- Index ID for a relational data index, an XML path index, an XML regions index, or an index over XML data
- Data Type for an index over XML data
- Hashed for an index over XML data
- Max VARCHAR Length for an index over XML data
- XML Pattern specified for an index over XML data
- Codepage for a text search index
- Language for a text search index
- Format specified for a text search index
- Update minimum for a text search index
- Update frequency for a text search index
- Collection directory for a text search index
- Column names
- Whether the BUSINESS\_TIME WITHOUT OVERLAPS clause is specified

Specify an index type to list information for only a specific index type. Specifying multiple index types is not supported.

## DESCRIBE

### RELATIONAL DATA

If the **RELATIONAL DATA** index type option is specified without the **SHOW DETAIL** option, only the following information is listed:

- Index schema
- Index name
- Unique rule
- Number of columns

If **SHOW DETAIL** is specified, the column names information is also listed.

### XML DATA

If the **XML DATA** index type option is specified without the **SHOW DETAIL** option, only the following information is listed:

- Index schema
- Index name
- Unique rule
- Number of columns
- Index type

If **SHOW DETAIL** is specified, the following information for an index over XML data is also listed:

- Index ID
- Data type
- Hashed
- Max Varchar length
- XML Pattern
- Column names

### TEXT SEARCH

If the **TEXT SEARCH** index type option is specified without the **SHOW DETAIL** option, only the following information is listed:

- Index schema
- Index name

If **SHOW DETAIL** is specified, the following text search index information is also listed:

- Column name
- Codepage
- Language
- Format
- Update minimum
- Update frequency
- Collection directory

If the **TEXT SEARCH** option is specified and a text search option is not installed or not properly configured, an error (SQLSTATE 42724) is returned.

See DB2 Text Search for information listed in the columns.

### 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.

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

#### 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
- Hidden attribute
- Default
- Table partitioning type (for tables partitioned by range this output appears after the original output)
- Partitioning key columns (for tables partitioned by range this output appears after the original output)
- Identifier of table space used for the index
- Periods that are defined on the table (for temporal tables this output appears after the original output)
- Whether versioning is enabled on the table (for temporal tables this output appears after the original output)

## Examples

### Describing the output of a SELECT Statement

The following example shows how to describe a SELECT statement:

```
db2 describe output select * from staff
```

Column Information

Number of columns: 7

Data Type	Length	Column Name	Name 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 of a CALL Statement

Given a stored procedure created with the statement:

## DESCRIBE

```
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:

```
db2 describe output call give_bonus(123456, 987, ?, 15000.)
```

Column Information

Number of Columns: 2

Data Type	Length	Column Name	Name Length
497 INTEGER	4	CHEQUE	6
485 DECIMAL	6, 0	BONUS	5

If the procedure has one or more parameters of an array type, the output from the **DESCRIBE** command has one additional column, that indicates the maximum cardinality of array parameters. An empty value indicates that the parameter is not an array.

Given the array type and procedure created with the statements:

```
CREATE TYPE PRODUCT_LIST AS INTEGER ARRAY[100]  
CREATE TYPE CUSTOMER_LIST AS INTEGER ARRAY[1000]
```

```
CREATE PROCEDURE DISCONTINUE_PROD (IN PROD_LIST PRODUCT_LIST,  
                                   IN EFFECTIVE_DATE DATE,  
                                   OUT NUM_PENDING_ORDERS INTEGER,  
                                   OUT CUST_LIST CUSTOMER_LIST)
```

...

The following example shows how to describe the output of a **CALL** statement with array parameters. The only format difference with the previous example is the Max cardinality column.

```
db2 describe output call discontinue_prod(ARRAY[12, 34, 26], '04/13/2006',?)
```

Column Information

Number of Columns: 2

SQL type	Type length	Column name	Name length	Max cardinality
497 INTEGER	4	NUM_PENDING_ORDERS	17	
497 INTEGER	10	CUSTOMER_LIST	13	1000

### 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
```

Column Information

Number of Columns: 1

SQL type	Type length	Column name	Name 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 previous server that does not support the **XQUERY** option, the message DB21108E is returned to indicate that the functionality is not supported by the previous server.

### Describing a Table

The following example shows how to describe a table:

```
db2 describe table user1.department
```

Table: USER1.DEPARTMENT

Column name	Data Type schema	Data Type name	Column 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 after the existing output. For a non-partitioned table, the additional table heading is not displayed:

```
db2 describe table user1.employee show detail
```

Column name	Data Type schema	Column number	Data Type name	Column length	Hidden
FIRST	SYSIBM	0	CHARACTER	10	No
LAST	SYSIBM	1	CHARACTER	10	No
EMPLOYEEENUM	SYSTEM	2	CHARACTER	10	Implicitly

Table is partitioned by range (ordered on the following column/s):

```
-----
LAST
FIRST
EMPLOYEEENUM
```

### Describing a Table Index

The following example shows how to describe a table index. This command lists two relational data indexes, six xml data indexes, two text search indexes, and the system indexes:

```
db2 describe indexes for table user1.department
```

Index schema	Index name	Unique rule	Number of columns	Index type
SYSIBM	SQL070531145253450	D	-	XML DATA - REGIONS
SYSIBM	SQL070531145253620	U	1	XML DATA - PATH
USER1	RELIDX1	D	1	RELATIONAL DATA
USER1	RELIDX2	D	2	RELATIONAL DATA
SYSIBM	SQL070531145253650	P	1	RELATIONAL DATA
USER1	XMLIDX1	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154625650	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX2	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626000	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX3	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626090	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX4	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626190	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX5	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626290	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX6	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626400	D	1	XML DATA - VALUES PHYSICAL
USER1	TXTIDX1	-	1	TEXT SEARCH
USER1	TXTIDX2	-	1	TEXT SEARCH

The following command lists the relational data indexes for table USER1.DEPARTMENT:

```
db2 describe relational data indexes for table user1.department
```

## DESCRIBE

Index schema	Index name	Unique rule	Number of columns
SYSIBM	SQL070531145253650	P	1
USER1	RELIDX1	D	1
USER1	RELIDX2	D	2

The following command lists the indexes over XML data for table USER1.DEPARTMENT:

```
db2 describe xml data indexes for table user1.department
```

Index schema	Index name	Unique rule	Number of columns	Index type
SYSIBM	SQL070531145253450	D	-	XML DATA - REGIONS
SYSIBM	SQL070531145253620	U	1	XML DATA - PATH
USER1	XMLIDX1	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154625650	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX2	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626000	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX3	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626090	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX4	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626190	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX5	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626290	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX6	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626400	D	1	XML DATA - VALUES PHYSICAL

The following command lists the text search index information for table USER1.DEPARTMENT:

```
db2 describe text search indexes for table user1.department
```

Index schema	Index name
USER1	TXTIDX1
USER1	TXTIDX2

The following command lists information about both partitioned and nonpartitioned indexes on the partitioned table myDpartT:

```
db2 describe indexes for table myDpartT
```

Index schema	Index name	Unique rule	Number of columns	Index Partitioning
NEWTON	IDXNDP	D	1	N
NEWTON	IDXDP	D	1	P

### Describing Data Partitions

The following example shows how to describe data partitions:

```
db2 describe data partitions for table user1.sales
```

PartitionId	Inclusive (y/n) Low Value	Inclusive (y/n) 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, and includes an additional table showing the Partition ID and table space where the data for the data partition is stored, and the ID of the table space where the index is stored:



db2 describe data partitions for table user1.employee show detail

PartitionId	Inclusive (y/n) Low Value	Inclusive (y/n) 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	IndexTblSpId	AccessMode	Status
0	PARTx	3	43	50	F	
1	PARTNew	13	13	13	N	A
2	PART3	31	33	35	F	
3	PART4	23	34	23	N	A

## DETACH

---

### 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

---

## 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.

## DROP CONTACTGROUP

---

### 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.

## 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 authorities:

- SYSADM
- SYSCTRL

### Required connection

Instance. An explicit attachment is not required. If the database is listed as remote, an instance attachment to the remote database partition server is established for the duration of the command.

### Command syntax

```

▶▶ DROP DATABASE database-alias AT DBPARTITIONNUM
    DB
  
```

### 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 IBM InfoSphere® Warehouse, can be used in partitioned database environments, 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:

```

SAMPLE:
db2 drop database sample
  
```

### Usage notes

**DROP DATABASE** deletes all user data and log files, as well as any backup and restore history for the database. If the log files are needed for a rollforward recovery after a restore operation, or the backup history required to restore the database, these files should be saved before issuing this command.

## DROP DATABASE

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.

**Related information:**

---

## 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 before dropping a database partition from a partitioned database environment.

### 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.

---

# DROP TOOLS CATALOG

Drops the DB2 tools catalog tables for the specified catalog in the given database.

This command is not valid on a IBM Data Server 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 authorities:

- 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.

---

## 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] [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.

## 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.

Quick link to “File type modifiers for the export utility” on page 187.

### Authorization

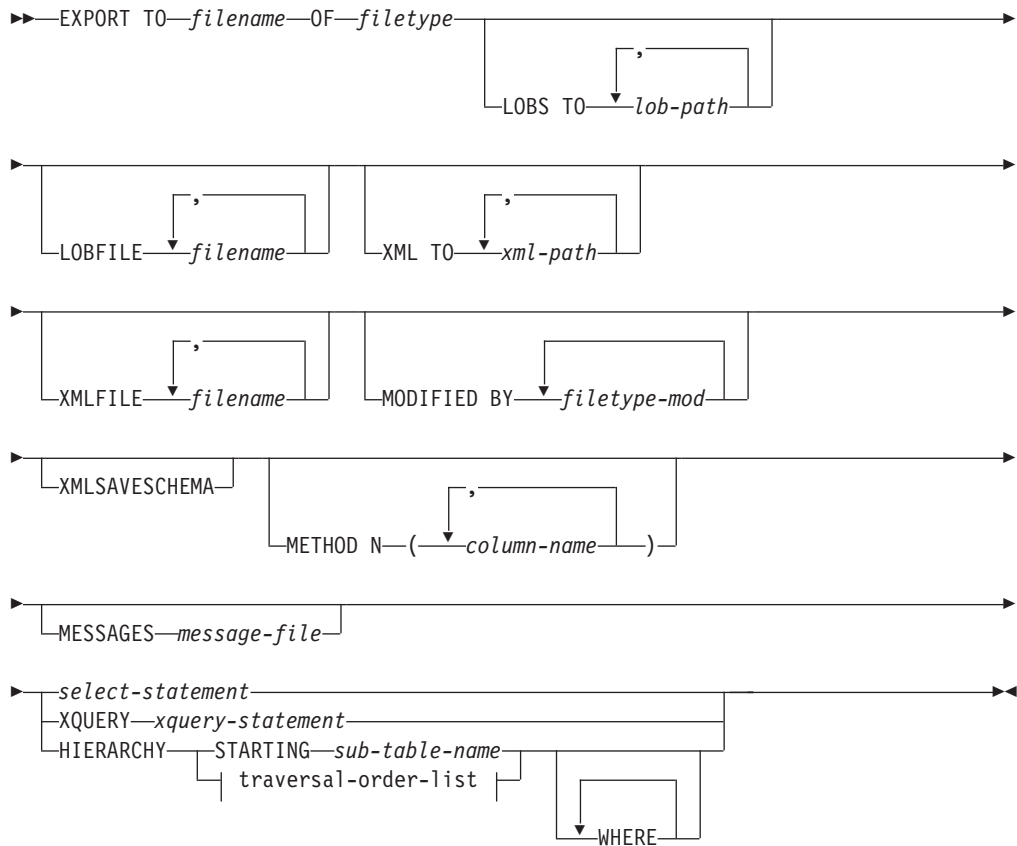
One of the following authorities:

- DATAACCESS authority
- 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 or DEL 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. 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 to start 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. The 3-digit sequence number in the LOB file name will grow to 4-digits once 999 is used, 4-digits will grow to 5-digits once 9999 is used, 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 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” on page 187.

### **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.
- IXF (Integration Exchange Format, PC version) is a proprietary binary format.

*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.

If the SELECT statement is in the form of SELECT \* FROM *tablename* and the table contains implicitly hidden columns, you must explicitly specify whether data for the hidden columns is included in the export operation. Use one of the following methods to indicate if data for hidden columns is included:

- Use one of the hidden column file type modifiers: specify **implicitlyhiddeninclude** when the export contains data for the hidden columns, or **implicitlyhiddenmissing** when the export does not.

```
db2 export to t.del of del modified by implicitlyhiddeninclude
select * from t
```

- Use the DB2\_DMU\_DEFAULT registry variable on the client-side to set the default behavior when data movement utilities encounter tables with implicitly hidden columns.

```
db2set DB2_DMU_DEFAULT=IMPLICITLYHIDDENINCLUDE
db2 export to t.del of del select * from t
```

**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 (XDM) instance. If more than one path is specified, then XDM instances are distributed evenly among the paths.

**XMLSAVESHEMA**

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 **XMLSAVESHEMA** 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.
- 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.

## EXPORT

- DB2 Connect can be used to export tables from DRDA servers such as DB2 for OS/390, DB2 for VM and VSE, and DB2 for OS/400<sup>®</sup>. Only PC/IXF export is supported.
- When exporting to the IXF format, if identifiers exceed the maximum size supported by the IXF format, the export will succeed but the resulting datafile cannot be used by a subsequent import operation using the CREATE mode. SQL27984W will be returned.
- When exporting to a diskette on Windows, and the table that has more data than the capacity of a single diskette, the system will prompt for another diskette, and multiple-part PC/IXF files (also known as multi-volume PC/IXF files, or logically split PC/IXF files), are generated and stored in separate diskettes. In each file, with the exception of the last, there is a DB2 CONTINUATION RECORD (or "AC" Record in short) written to indicate the files are logically split and where to look for the next file. The files can then be transferred to an AIX system, to be read by the import and load utilities. The export utility will not create multiple-part PC/IXF files when invoked from an AIX system. For detailed usage, see the **IMPORT** command or **LOAD** command.
- 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.
- When running Data Movement utilities such as **export** and **db2move**, the query compiler might determine that the underlying query will run more efficiently against an MQT than the base table or tables. In this case, the query will execute against a refresh deferred MQT, and the result of the utilities might not accurately represent the data in the underlying table.
- 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 behavior 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

```

## File type modifiers for the export utility

Table 10. 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 db2exp.001.123.456/ is stored in the data file, the LOB is located at offset 123 in the file db2exp.001, and is 456 bytes long.</p> <p>If you specify the <b>lobsinfile</b> modifier when using <b>EXPORT</b>, the LOB data is placed in the locations specified by the <b>LOBS TO</b> clause. Otherwise the LOB data is sent to the data file directory. The <b>LOBS TO</b> 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 <b>LOBS TO</b> or <b>LOBFILE</b> 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 db2exp.001.7.-1/.</p>
implicitlyhiddeninclude	<p>This modifier is used with SELECT * queries and specifies that the data in implicitly hidden columns is exported even though that data is not included in the result of the SELECT * query. This modifier cannot be used with the <b>implicitlyhiddenmissing</b> modifier.</p> <p>If this modifier is used and the query is not a SELECT *, then an error is returned (SQLCODE SQL3526N).</p>
implicitlyhiddenmissing	<p>This modifier is used with SELECT * queries and specifies that the data in implicitly hidden columns is not exported. This modifier cannot be used with the <b>implicitlyhiddeninclude</b> modifier.</p> <p>If this modifier is used and the query is not a SELECT *, then an error is returned (SQLCODE SQL3526N).</p>
xmlinsefiles	<p>Each XQuery Data Model (XDM) instance is written to a separate file. By default, multiple values are concatenated together in the same file.</p>
lobsinsefiles	<p>Each LOB value is written to a separate file. By default, multiple values are concatenated together in the same file.</p>

## EXPORT

Table 10. Valid file type modifiers for the export utility: All file formats (continued)

Modifier	Description
xmlnodeclaration	XDM instances are written without an XML declaration tag. By default, XDM instances are exported with an XML declaration tag at the beginning that includes an encoding attribute.
xmlchar	XDM instances are written in the character code page. Note that the character codepage is the value specified by the codepage file type modifier, or the application code page if it is not specified. By default, XDM instances are written out in Unicode.
xmlgraphic	If the xmlgraphic modifier is specified with the <b>EXPORT</b> 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 11. Valid file type modifiers for the export utility: DEL (delimited ASCII) file format

Modifier	Description
chardelx	<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.<sup>2</sup> 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 charde1""</pre> <p>The single quotation mark (') can also be specified as a character string delimiter as follows:</p> <pre>modified by charde1''</pre>
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 to this code page from 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 lobsinfile modifier.</p>
coldelx	<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.<sup>2</sup></p> <p>In the following example, coldel; causes the export utility to use the semicolon character (;) as a column delimiter for the exported data:</p> <pre>db2 "export to temp of del modified by coldel; select * from staff where dept = 20"</pre>
decplusblank	Plus sign character. Causes positive decimal values to be prefixed with a blank space instead of a plus sign (+). The default action is to prefix positive decimal values with a plus sign.
decptx	<i>x</i> 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. <sup>2</sup>
nochardel	<p>Column data will not be surrounded by character delimiters. This option should not be specified if the data is intended to be imported or loaded using DB2. 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 charde1x or nodoublede1. These are mutually exclusive options.</p>
nodoublede1	Suppresses recognition of double character delimiters. <sup>2</sup>

Table 11. Valid file type modifiers for the export utility: DEL (delimited ASCII) file format (continued)

Modifier	Description
striplzeros	<p>Removes the leading zeros from all exported decimal columns.</p> <p>Consider the following example:</p> <pre>db2 create table decimalTable ( c1 decimal( 31, 2 ) ) db2 insert into decimalTable values ( 1.1 )  db2 export to data of del select * from decimalTable  db2 export to data of del modified by STRIPLZEROS select * from decimalTable</pre> <p>In the first export operation, the content of the exported file data will be +00000000000000000000000000000001.10. In the second operation, which is identical to the first except for the striplzeros modifier, the content of the exported file data will be +1.10.</p>
timestampformat="x"	<p><i>x</i> is the format of the time stamp in the source file.<sup>4</sup> Valid time stamp elements are:</p> <pre>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 01 - 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 00 - 12 for a 12 hour system, and 00 - 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 00 - 59; mutually exclusive with M, minute) S - Second (one or two digits ranging from 0 - 59) SS - Second (two digits ranging from 00 - 59; mutually exclusive with S) SSSSS - Second of the day after midnight (5 digits ranging from 00000 - 86400; mutually exclusive with other time elements) U (1 to 12 times) - Fractional seconds(number of occurrences of U represent the number of digits with each digit ranging from 0 to 9) TT - Meridian indicator (AM or PM)</pre> <p>Following is an example of a time stamp format:</p> <pre>"YYYY/MM/DD HH:MM:SS.UUUUUU"</pre> <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="yyyy.mm.dd hh:mm tt" select * from schedule</pre>

## EXPORT

Table 12. 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.</p>

### Note:

1. The export utility does not issue a warning if an attempt is made to use unsupported file types with the **MODIFIED BY** option. If this is attempted, the export operation fails, and an error code is returned.
2. *Delimiter considerations for moving data* lists restrictions that apply to the characters that can be used as delimiter overrides.
3. 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.aaaaaaa" format

Data contained in any datetime columns specified in the SELECT statement for the export operation will also be in these formats.

4. 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)

5. All XDM 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 extension ".xml" appended to it.
6. All XDM instances are written with an XML declaration at the beginning that includes an encoding attribute, unless the XMLNODEDECLARATION file type modifier is specified.
7. By default, all XDM instances are written in Unicode unless the XMLCHAR or XMLGRAPHIC file type modifier is specified.
8. 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.

9. The export utility appends a numeric identifier to each LOB file or XML file. The identifier starts as 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 zeros (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`

10. It is possible to have the export utility export XDM instances that are not well-formed documents by specifying an XQuery. However, you will not be able to import or load these exported documents directly into an XML column, since XML columns can only contain complete documents.

**Related information:**

## 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 database partition server in the partitioned database environment.

### Authorization

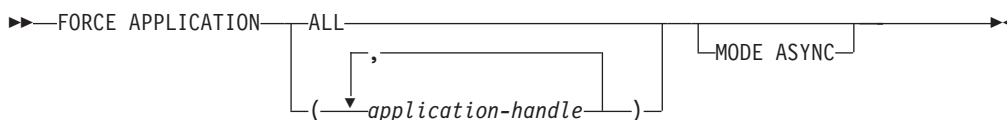
One of the following authorities:

- 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

#### FORCE 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

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.

The following types of users and applications cannot be forced:

- users creating a database
- system applications

In order to successfully force these types of users and applications, the database must be deactivated and/or the instance restarted.

After a **FORCE APPLICATION** has been issued, the database will still accept requests to connect. Additional forces might be required to completely force all users off.

**Related information:**

## 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.

**Important:** The DB2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in DB2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see “DB2 administration server (DAS) has been deprecated” at .

### 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.

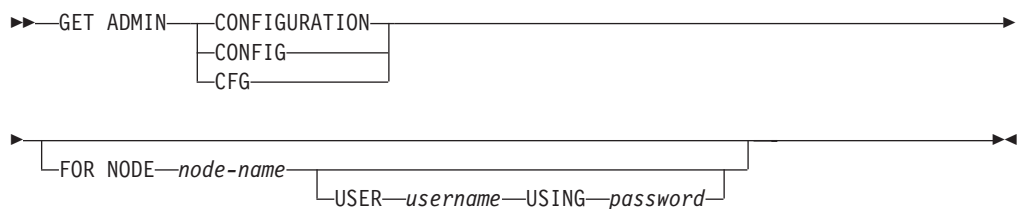
### Authorization

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** *node-name*

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 example 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.

## GET ALERT CONFIGURATION

Returns the alert configuration settings for health indicators for a particular instance.

**Important:** This command or API has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in DB2 pureScale environments. For more information, see “Health monitor has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

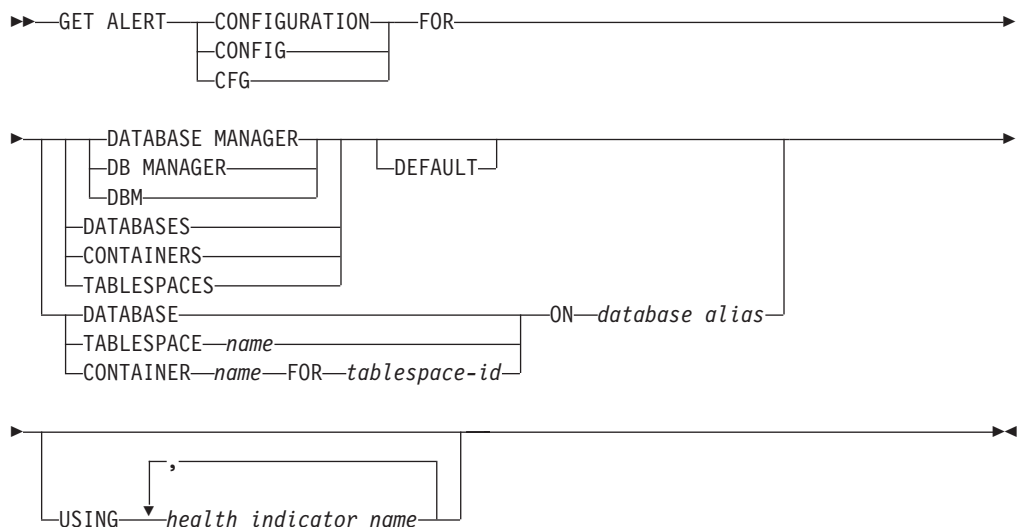
### Authorization

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.

**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 section 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
```

## GET ALERT CONFIGURATION

```
Threshold or State checking = Enabled
Indicator Name              = db2.mon_heap_util
Default                    = Yes
Type                      = Threshold-based
Warning                   = 85
Alarm                     = 95
Unit                      = %
Sensitivity                = 0
Formula                   = ((db2.mon_heap_cur_size/
                             db2.mon_heap_max_size)*100);
Actions                    = Disabled
Threshold or State checking = Enabled
```

The following section 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/sheapthres_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/
                             sheapthres_shr)*100);
Actions                    = Disabled
Threshold or State checking = Enabled

Indicator Name              = db.log_util
Default                    = Yes
Type                      = Threshold-based
```

## GET ALERT CONFIGURATION

```
Warning = 75
Alarm = 85
Unit = %
Sensitivity = 0
Formula = (db.total_log_used/
           (db.total_log_used+db.total_log_available)
           )*100;
Actions = Disabled
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
```

## GET ALERT CONFIGURATION

```

Indicator Name           = db.pkgcache_hitratio
Default                 = Yes
Type                   = Threshold-based
Warning                = 80
Alarm                  = 70
Unit                   = %
Sensitivity             = 0
Formula                = (1-
                        (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

```

## GET ALERT CONFIGURATION

```
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)
                        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
```

---

# 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 CLI and the applications using it. The file is divided into sections, each section corresponding to a database alias name.

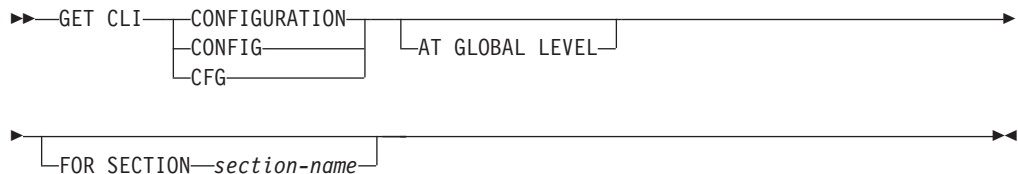
### Authorization

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.



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) using 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.

### 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 example 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.

## 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

### GET CONTACTGROUPS

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

---

## 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—▶

### Command parameters

None

### Examples

GET CONTACTS

Name	Type	Address	Max Page Length	Description
-----	-----	-----	-----	-----
joe	e-mail	joe@somewhere.com	-	-
joline	e-mail	joline@somewhereelse.com	-	-
john	page	john@relay.org	50	Support 24x7

---

# 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.

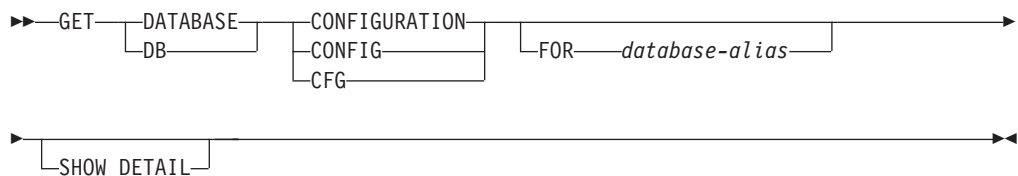
### Authorization

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.

This is a default clause when operating in the CLPPlus interface. **SHOW DETAIL** need not be called when using CLPPlus processor.

If the **SHOW DETAIL** option is not specified, this command only returns the value stored in DISK which might be different from the current value in memory.

### Examples

#### Note:

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 example is sample output after running the **GET DATABASE CONFIGURATION** command outside of a DB2 pureScale environment with a UNIX operating system:

```

Database Configuration for Database sample

Database configuration release level      = 0x0f00
Database release level                   = 0x0f00

Database territory                        = US
Database code page                       = 1208
Database code set                        = UTF-8
Database country/region code            = 1
Database collating sequence              = IDENTITY
Alternate collating sequence             (ALT_COLLATE) =
Number compatibility                     = OFF
Varchar2 compatibility                  = OFF
Date compatibility                       = OFF
Database page size                       = 4096

Dynamic SQL Query management            (DYN_QUERY_MGMT) = DISABLE

Statement concentrator                   (STMT_CONC) = OFF

Discovery support for this database      (DISCOVER_DB) = ENABLE

Restrict access                          = NO
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

Decimal floating point rounding mode     (DECFLT_ROUNDING) = ROUND_HALF_EVEN

Backup pending                           = NO

All committed transactions have been written to disk = 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

Self tuning memory                      (SELF_TUNING_MEM) = OFF
Size of database shared memory (4KB)     (DATABASE_MEMORY) = AUTOMATIC(40944)
Database memory threshold                (DB_MEM_THRESH) = 10
Max storage for lock list (4KB)          (LOCKLIST) = 4096
Percent. of lock lists per application   (MAXLOCKS) = 10
Package cache size (4KB)                 (PCKCACHESZ) = (MAXAPPLS*8)
Sort heap thres for shared sorts (4KB)   (SHEAPTHRES_SHR) = 5000
Sort list heap (4KB)                    (SORTHEAP) = 256

Database heap (4KB)                      (DBHEAP) = AUTOMATIC(1200)
Catalog cache size (4KB)                 (CATALOGCACHE_SZ) = (MAXAPPLS*5)
Log buffer size (4KB)                    (LOGBUFSZ) = 256
Utilities heap size (4KB)                (UTIL_HEAP_SZ) = 5000
Buffer pool size (pages)                 (BUFFPAGE) = 1000
SQL statement heap (4KB)                 (STMTHEAP) = AUTOMATIC(8192)
Default application heap (4KB)           (APPLHEAPSZ) = AUTOMATIC(256)
Application Memory Size (4KB)            (APPL_MEMORY) = AUTOMATIC(40000)
Statistics heap size (4KB)               (STAT_HEAP_SZ) = AUTOMATIC(4384)

Interval for checking deadlock (ms)      (DLCHKTIME) = 10000
Lock timeout (sec)                       (LOCKTIMEOUT) = -1

Changed pages threshold                  (CHNGPGS_THRESH) = 60
Number of asynchronous page cleaners     (NUM_IOCLEANERS) = AUTOMATIC(1)
Number of I/O servers                    (NUM_IOSERVERS) = AUTOMATIC(3)
Index sort flag                          (INDEXSORT) = YES
Sequential detect flag                   (SEQDETECT) = YES
Default prefetch size (pages)            (DFT_PREFETCH_SZ) = AUTOMATIC

Track modified pages                     (TRACKMOD) = NO

Default number of containers              = 1
Default tablespace extentsize (pages)    (DFT_EXTENT_SZ) = 32

Max number of active applications        (MAXAPPLS) = AUTOMATIC(40)
Average number of active applications    (AVG_APPLS) = AUTOMATIC(1)
Max DB files open per application        (MAXFILOP) = 61440

```

## GET DATABASE CONFIGURATION

```

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/db2inst1/db2inst1/SAMPLE/NODE0000/LOGSTREAM0000/
Overflow log path (OVERFLOWLOGPATH) =
Mirror log path (MIRRORLOGPATH) =
First active log file =
Block log on disk full (BLK_LOG_DSK_FUL) = NO
Block non logged operations (BLOCKNONLOGGED) = NO
Percent 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

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 target list (HADR_TARGET_LIST) =
HADR log write synchronization mode (HADR_SYNCMODE) = NEARSYNC
HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0
HADR spool log data limit (4KB) (HADR_SPOOL_LIMIT) = 0
HADR log replay delay (seconds) (HADR_REPLAY_DELAY) = 0
First log archive method (LOGARCHMETH1) = OFF
Archive compression for logarchmeth1 (LOGARCHCOMPR1) = OFF
Options for logarchmeth1 (LOGARCHOPT1) =
Second log archive method (LOGARCHMETH2) = OFF
Archive compression for logarchmeth2 (LOGARCHCOMPR2) = 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
Auto deletion of recovery objects (AUTO_DEL_REC_OBJ) = OFF

TSM management class (TSM_MGMTCLASS) =
TSM node name (TSM_NODENAME) =
TSM owner (TSM_OWNER) =
TSM password (TSM_PASSWORD) =

Automatic maintenance (AUTO_MAINT) = ON
Automatic database backup (AUTO_DB_BACKUP) = OFF
Automatic table maintenance (AUTO_TBL_MAINT) = ON
Automatic runstats (AUTO_RUNSTATS) = ON
Automatic statement statistics (AUTO_STMT_STATS) = ON
Automatic stat view statistics (AUTO_STATS_VIEWS) = OFF
Automatic statistics profiling (AUTO_STATS_PROF) = OFF
Automatic profile updates (AUTO_PROF_UPD) = OFF
Automatic reorganization (AUTO_REORG) = OFF

Auto-Revalidation (AUTO_REVAL) = DEFERRED
Currently Committed (CUR_COMMIT) = ON
CHAR output with DECIMAL input (DEC_TO_CHAR_FMT) = NEW
Enable XML Character operations (ENABLE_XMLCHAR) = YES
WLM Collection Interval (minutes) (WLM_COLLECT_INT) = 0
Monitor Collect Settings
Request metrics (MON_REQ_METRICS) = BASE
Activity metrics (MON_ACT_METRICS) = BASE
Object metrics (MON_OBJ_METRICS) = BASE
Unit of work events (MON_UOW_DATA) = NONE
UOW events with package list (MON_UOW_PKGLIST) = OFF
UOW events with executable list (MON_UOW_EXECLIST) = OFF
Lock timeout events (MON_LOCKTIMEOUT) = NONE
Deadlock events (MON_DEADLOCK) = WITHOUT_HIST
Lock wait events (MON_LOCKWAIT) = NONE
Lock wait event threshold (MON_LW_THRESH) = 5000000
Number of package list entries (MON_PKGLIST_SZ) = 32
Lock event notification level (MON_LCK_MSG_LVL) = 1

SMTP Server (SMTP_SERVER) =
SQL conditional compilation flags (SQL_CCFLAGS) =
Section actuals setting (SECTION_ACTUALS) = NONE
Connect procedure (CONNECT_PROC) =
Adjust temporal SYSTEM_TIME period (SYSTIME_PERIOD_ADJ) = NO
Log DDL Statements (LOG_DDL_STMTS) = NO
Log Application Information (LOG_APPL_INFO) = NO
Default Data Capture on new Schemas (DFT_SCHEMAS_DCC) = NO
Database is in write suspend state = NO

```



## GET DATABASE CONFIGURATION

The following example is sample output after running the **GET DATABASE CONFIGURATION** command in a DB2 pureScale environment with a Linux operating system:

```
Database Configuration for Database sample

Database configuration release level          = 0x0e00
Database release level                      = 0x0e00

Database territory                          = US
Database code page                          = 1208
Database code set                           = UTF-8
Database country/region code               = 1
Database collating sequence                 = IDENTITY
Alternate collating sequence                (ALT_COLLATE) =
Number compatibility                        = OFF
Varchar2 compatibility                     = OFF
Date compatibility                         = OFF
Database page size                          = 4096

Statement concentrator                      (STMT_CONC) = OFF

Discovery support for this database         (DISCOVER_DB) = ENABLE

Restrict access                             = NO
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

Decimal floating point rounding mode       (DECFLT_ROUNDING) = ROUND_HALF_EVEN

Backup pending                             = NO

All committed transactions have been written to disk = 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

Self tuning memory                        (SELF_TUNING_MEM) = ON
Size of database shared memory (4KB)      (DATABASE_MEMORY) = AUTOMATIC(60032)
Database memory threshold                 (DB_MEM_THRESH)  = 10
Max storage for lock list (4KB)           (LOCKLIST)       = AUTOMATIC(6575)
Percent. of lock lists per application    (MAXLOCKS)       = AUTOMATIC(60)
Package cache size (4KB)                  (PCKCACHESZ)    = AUTOMATIC(1533)
Sort heap thres for shared sorts (4KB)    (SHEAPTHRES_SHR) = AUTOMATIC(21919)
Sort list heap (4KB)                      (SORTHEAP)      = AUTOMATIC(1095)

Database heap (4KB)                       (DBHEAP)        = AUTOMATIC(2462)
Catalog cache size (4KB)                  (CATALOGCACHE_SZ) = 300
Log buffer size (4KB)                     (LOGBUFSZ)      = 256
Utilities heap size (4KB)                  (UTIL_HEAP_SZ)  = 52763
Buffer pool size (pages)                   (BUFFPAGE)      = 1000
SQL statement heap (4KB)                   (STMTHEAP)     = AUTOMATIC(4096)
Default application heap (4KB)             (APPLHEAPSZ)   = AUTOMATIC(256)
Application Memory Size (4KB)              (APPL_MEMORY)  = AUTOMATIC(40000)
Statistics heap size (4KB)                 (STAT_HEAP_SZ)  = AUTOMATIC(4384)

Interval for checking deadlock (ms)        (DLCHKTIME)    = 10000
Lock timeout (sec)                        (LOCKTIMEOUT)  = -1

Changed pages threshold                   (CHNGPGS_THRESH) = 80
Number of asynchronous page cleaners       (NUM_IOCLEANERS) = AUTOMATIC(1)
Number of I/O servers                     (NUM_IOSERVERS)  = AUTOMATIC(4)
Index sort flag                            (INDEXSORT)     = YES
Sequential detect flag                     (SEQDETECT)     = YES
Default prefetch size (pages)             (DFT_PREFETCH_SZ) = AUTOMATIC

Track modified pages                      (TRACKMOD)      = NO

Default number of containers               = 1
```

## GET DATABASE CONFIGURATION

```
Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32

Max number of active applications (MAXAPPLS) = AUTOMATIC(40)
Average number of active applications (AVG_APPLS) = AUTOMATIC(1)
Max DB files open per application (MAXFILOP) = 61440

Log file size (4KB) (LOGFILSIZ) = 1024
Number of primary log files (LOGPRIMARY) = 13
Number of secondary log files (LOGSECOND) = 4
Changed path to log files (NEWLOGPATH) =
Path to log files = /home/db2inst1/db2inst1/SAMPLE/NODE0000/LOGSTREAM0000/
Overflow log path (OVERFLOWLOGPATH) =
Mirror log path (MIRRORLOGPATH) =
First active log file =
Block log on disk full (BLK_LOG_DSK_FUL) = NO
Block non logged operations (BLOCKNONLOGGED) = NO
Percent 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 chkpt (SOFTMAX) = 520
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
HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0
HADR spool log data limit (4KB) (HADR_SPOOL_LIMIT) = 0
HADR log replay delay (seconds) (HADR_REPLAY_DELAY) = 0
First log archive method (LOGARCHMETH1) = OFF
Archive compression for logarchmeth1 (LOGARCHCOMPRI) = OFF
Options for logarchmeth1 (LOGARCHOPT1) =
Second log archive method (LOGARCHMETH2) = OFF
Archive compression for logarchmeth2 (LOGARCHCOMPRI2) = 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
Auto deletion of recovery objects (AUTO_DEL_REC_OBJ) = OFF

TSM management class (TSM_MGMTCLASS) =
TSM node name (TSM_NODENAME) =
TSM owner (TSM_OWNER) =
TSM password (TSM_PASSWORD) =

Automatic maintenance (AUTO_MAINT) = ON
Automatic database backup (AUTO_DB_BACKUP) = OFF
Automatic table maintenance (AUTO_TBL_MAINT) = ON
Automatic runstats (AUTO_RUNSTATS) = ON
Automatic statement statistics (AUTO_STMT_STATS) = ON
Automatic statistics profiling (AUTO_STATS_PROF) = OFF
Automatic profile updates (AUTO_PROF_UPD) = OFF
Automatic reorganization (AUTO_REORG) = OFF

Auto-Revalidation (AUTO_REVAL) = DEFERRED
CF Resource Configuration:
CF database memory size (4KB) (CF_DB_MEM_SZ) = AUTOMATIC(127488)
Group buffer pool size (4KB) (CF_GBP_SZ) = AUTOMATIC(80896)
Global lock memory size (4KB) (CF_LOCK_SZ) = AUTOMATIC(19200)
Shared communication area size (4KB) (CF_SCA_SZ) = AUTOMATIC(26368)

Catchup target for secondary CF (mins)(CF_CATCHUP_TRGT) = AUTOMATIC(15)

Currently Committed (CUR_COMMIT) = ON
CHAR output with DECIMAL input (DEC_TO_CHAR_FMT) = NEW
Enable XML Character operations (ENABLE_XMLCHAR) = YES
WLM Collection Interval (minutes) (WLM_COLLECT_INT) = 0
Monitor Collect Settings
```

```

Request metrics          (MON_REQ_METRICS) = BASE
Activity metrics        (MON_ACT_METRICS) = BASE
Object metrics         (MON_OBJ_METRICS) = BASE
Unit of work events    (MON_UOW_DATA) = NONE
Lock timeout events    (MON_LOCKTIMEOUT) = NONE
Deadlock events        (MON_DEADLOCK) = WITHOUT_HIST
Lock wait events       (MON_LOCKWAIT) = NONE
Lock wait event threshold (MON_LW_THRESH) = 5000000
Number of package list entries (MON_PKGLIST_SZ) = 32
Lock event notification level (MON_LCK_MSG_LVL) = 1

SMTP Server            (SMTP_SERVER) =
SQL conditional compilation flags (SQL_CCFLAGS) =
Section actuals setting (SECTION_ACTUALS) = NONE

Database is in write suspend state = NO
  
```

The following example shows the output of the command when you specify the **SHOW DETAIL** option in a DB2 pureScale environment. 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 sample

Description	Parameter	Current Value	Delayed Value
Database configuration release level		= 0x0e00	
Database release level		= 0x0e00	
Database territory		= US	
Database code page		= 1208	
Database code set		= UTF-8	
Database country/region code		= 1	
Database collating sequence		= IDENTITY	IDENTITY
Alternate collating sequence	(ALT_COLLATE)	=	
Number compatibility		= OFF	
Varchar2 compatibility		= OFF	
Date compatibility		= OFF	
Database page size		= 4096	4096
Statement concentrator	(STMT_CONC)	= OFF	OFF
Discovery support for this database	(DISCOVER_DB)	= ENABLE	ENABLE
Restrict access		= NO	
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
Decimal floating point rounding mode	(DECFLT_ROUNDING)	= ROUND_HALF_EVEN	ROUND_HALF_EVEN
Backup pending		= NO	
All committed transactions have been written to disk		= 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	
Self tuning memory	(SELF_TUNING_MEM)	= ON (Active)	ON
Size of database shared memory (4KB)	(DATABASE_MEMORY)	= AUTOMATIC(345584)	AUTOMATIC(345584)
Database memory threshold	(DB_MEM_THRESH)	= 10	10
Max storage for lock list (4KB)	(LOCKLIST)	= AUTOMATIC(6575)	AUTOMATIC(6575)
Percent. of lock lists per application	(MAXLOCKS)	= AUTOMATIC(60)	AUTOMATIC(60)
Package cache size (4KB)	(PCKCACHESZ)	= AUTOMATIC(1533)	AUTOMATIC(1533)
Sort heap thres for shared sorts (4KB)	(SHEAPTHRES_SHR)	= AUTOMATIC(21919)	AUTOMATIC(21919)
Sort list heap (4KB)	(SORTHEAP)	= AUTOMATIC(1095)	AUTOMATIC(1095)
Database heap (4KB)	(DBHEAP)	= AUTOMATIC(2462)	AUTOMATIC(2462)
Catalog cache size (4KB)	(CATALOGCACHE_SZ)	= 300	300
Log buffer size (4KB)	(LOGBUFSZ)	= 256	256
Utilities heap size (4KB)	(UTIL_HEAP_SZ)	= 52763	52763
Buffer pool size (pages)	(BUFFPAGE)	= 1000	1000
SQL statement heap (4KB)	(STMTHEAP)	= AUTOMATIC(6402)	AUTOMATIC(6402)
Default application heap (4KB)	(APPLHEAPSZ)	= AUTOMATIC(256)	AUTOMATIC(256)
Application Memory Size (4KB)	(APPL_MEMORY)	= AUTOMATIC(40016)	AUTOMATIC(40000)
Statistics heap size (4KB)	(STAT_HEAP_SZ)	= AUTOMATIC(4384)	AUTOMATIC(4384)
Interval for checking deadlock (ms)	(DLCHKTIME)	= 10000	10000
Lock timeout (sec)	(LOCKTIMEOUT)	= -1	-1
Changed pages threshold	(CHNGPGS_THRESH)	= 80	80
Number of asynchronous page cleaners	(NUM_IOCLEANERS)	= AUTOMATIC(3)	AUTOMATIC(3)
Number of I/O servers	(NUM_IOSERVERS)	= AUTOMATIC(6)	AUTOMATIC(6)
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

## GET DATABASE CONFIGURATION

```

Max number of active applications          (MAXAPPLS) = AUTOMATIC(40)          AUTOMATIC(40)
Average number of active applications     (AVG_APPLS) = AUTOMATIC(1)         AUTOMATIC(1)
Max DB files open per application         (MAXFILOP) = 61440                 61440

Log file size (4KB)                      (LOGFILSIZ) = 1024                 1024
Number of primary log files              (LOGPRIMARY) = 13                  13
Number of secondary log files            (LOGSECOND) = 4                    4
Changed path to log files                (NEWLOGPATH) =                     =
Path to log files                        = /home/db2inst1/db2inst1/SAMPLE/NODE0000/LOGSTREAM0000/ /home/db2inst1/db2inst1/SAMPLE/NODE0000/LOGSTREAM0000/
Overflow log path                        (OVERFLOWLOGPATH) =               =
Mirror log path                          (MIRRORLOGPATH) =                 =
First active log file                    =
Block log on disk full                   (BLK_LOG_DSK_FUL) = NO             NO
Block non logged operations              (BLOCKNONLOGGED) = NO              NO
Percent 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 ckcpt (SOFTMAX) = 520                520
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
HADR peer window duration (seconds)     (HADR_PEER_WINDOW) = 0             0
HADR spool log data limit (4KB)         (HADR_SPOOL_LIMIT) = 0             0
HADR log replay delay (seconds)         (HADR_REPLAY_DELAY) = 0            0
First log archive method                 (LOGARCHMETH1) = OFF              OFF
Archive compression for logarchmeth1     (LOGARCHCOMP1) = OFF              OFF
Options for logarchmeth1                 (LOGARCHOPT1) =                   =
Second log archive method                (LOGARCHMETH2) = OFF              OFF
Archive compression for logarchmeth2     (LOGARCHCOMP2) = 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 (RESTART)     SYSTEM (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
Auto deletion of recovery objects        (AUTO_DEL_REC_OBJ) = OFF           OFF

TSM management class                    (TSM_MGMTCLASS) =                  =
TSM node name                           (TSM_NODENAME) =                  =
TSM owner                                (TSM_OWNER) =                     =
TSM password                             (TSM_PASSWORD) =                  =

Automatic maintenance                   (AUTO_MAINT) = ON                  ON
Automatic database backup                (AUTO_DB_BACKUP) = OFF             OFF
Automatic table maintenance              (AUTO_TBL_MAINT) = ON              ON
Automatic runstats                       (AUTO_RUNSTATS) = ON              ON
Automatic statement statistics           (AUTO_STMT_STATS) = ON             ON
Automatic statistics profiling            (AUTO_STATS_PROF) = OFF           OFF
Automatic profile updates                 (AUTO_PROF_UPD) = OFF             OFF
Automatic reorganization                 (AUTO_REORG) = OFF                OFF

Auto-Revalidation                       (AUTO_REVAL) = DEFERRED            DEFERRED
CF Resource Configuration:
CF database memory size (4KB)            (CF_DB_MEM_SZ) = AUTOMATIC(114688) AUTOMATIC(114688)
Group buffer pool size (4KB)             (CF_GBP_SZ) = AUTOMATIC(72704)    AUTOMATIC(72704)
Global lock memory size (4KB)            (CF_LOCK_SZ) = AUTOMATIC(17408)   AUTOMATIC(17408)
Shared communication area size (4KB)     (CF_SCA_SZ) = AUTOMATIC(23552)    AUTOMATIC(23552)

Catchup target for secondary CF (mins)(CF_CATCHUP_TRGT) = AUTOMATIC(15)    AUTOMATIC(15)

Currently Committed                      (CUR_COMMIT) = ON                  ON
CHAR output with DECIMAL input           (DEC_TO_CHAR_FMT) = NEW            NEW
Enable XML Character operations           (ENABLE_XMLCHAR) = YES             YES
WLM Collection Interval (minutes)        (WLM_COLLECT_INT) = 0              0

Monitor Collect Settings
Request metrics                          (MON_REQ_METRICS) = BASE           BASE
Activity metrics                         (MON_ACT_METRICS) = BASE           BASE
Object metrics                           (MON_OBJ_METRICS) = BASE           BASE
Unit of work events                      (MON_UOW_DATA) = NONE             NONE
Lock timeout events                      (MON_LOCKTIMEOUT) = NONE           NONE
Deadlock events                          (MON_DEADLOCK) = WITHOUT_HIST     WITHOUT_HIST
Lock wait events                         (MON_LOCKWAIT) = NONE             NONE
Lock wait event threshold                 (MON_LW_THRESH) = 5000000         5000000

Number of package list entries            (MON_PKGLIST_SZ) = 32              32
Lock event notification level             (MON_LCK_MSG_LVL) = 1              1

SMTP Server                              (SMTP_SERVER) =                   =
SQL conditional compilation flags         (SQL_CCFLAGS) =                   =

Section actuals setting                  (SECTION_ACTUALS) = NONE           NONE

Database is in write suspend state        = NO

```

The following example is sample output after running the **GET DATABASE CONFIGURATION** command on a Windows operating system:

## GET DATABASE CONFIGURATION

### Database Configuration for Database

```
Database configuration release level      = 0x0d00
Database release level                   = 0x0d00

Database territory                        = US
Database code page                        = 1208
Database code set                         = UTF-8
Database country/region code             = 1
Database collating sequence              = IDENTITY
Alternate collating sequence              (ALT_COLLATE) =
Number compatibility                      = OFF
Varchar2 compatibility                   = OFF
Date compatibility                        = OFF

Database page size                       = 4096

Dynamic SQL Query management              (DYN_QUERY_MGMT) = DISABLE
Statement concentrator                    (STMT_CONC) = OFF

Discovery support for this database        (DISCOVER_DB) = ENABLE

Restrict access                           = NO
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

Decimal floating point rounding mode      (DECFLT_ROUNDING) = ROUND_HALF_EVEN

Backup pending                            = NO

All committed transactions have been written to disk = 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

Self tuning memory                        (SELF_TUNING_MEM) = ON
Size of database shared memory (4KB)      (DATABASE_MEMORY) = AUTOMATIC(60464)
Database memory threshold                 (DB_MEM_THRESH) = 10
Max storage for lock list (4KB)           (LOCKLIST) = AUTOMATIC(6200)
Percent. of lock lists per application    (MAXLOCKS) = AUTOMATIC(60)
Package cache size (4KB)                  (PCKCACHESZ) = AUTOMATIC(1533)
Sort heap thres for shared sorts (4KB)    (SHEAPTHRES_SHR) = AUTOMATIC(6728)
Sort list heap (4KB)                      (SORTHEAP) = AUTOMATIC(336)

Database heap (4KB)                       (DBHEAP) = AUTOMATIC(2283)
Catalog cache size (4KB)                  (CATALOGCACHE_SZ) = 300
Log buffer size (4KB)                     (LOGBUFSZ) = 256
Utilities heap size (4KB)                  (UTIL_HEAP_SZ) = 5115
Buffer pool size (pages)                   (BUFFPAGE) = 1000
SQL statement heap (4KB)                   (STMHEAP) = AUTOMATIC(4096)
Default application heap (4KB)             (APPLHEAPSZ) = AUTOMATIC(256)
Application Memory Size (4KB)             (APPL_MEMORY) = AUTOMATIC(40000)
Statistics heap size (4KB)                 (STAT_HEAP_SZ) = AUTOMATIC(4384)

Interval for checking deadlock (ms)        (DLCHKTIME) = 10000
Lock timeout (sec)                        (LOCKTIMEOUT) = -1
```

## GET DATABASE CONFIGURATION

```
Changed pages threshold          (CHNGPGS_THRESH) = 80
Number of asynchronous page cleaners (NUM_IOCLEANERS) = AUTOMATIC(3)
Number of I/O servers            (NUM_IOSERVERS) = AUTOMATIC(3)
Index sort flag                  (INDEXSORT) = YES
Sequential detect flag           (SEQDETECT) = YES
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(40)
Average number of active applications (AVG_APPLS) = AUTOMATIC(1)
Max DB files open per application (MAXFILOP) = 61440

Log file size (4KB)              (LOGFILSIZ) = 1024
Number of primary log files       (LOGPRIMARY) = 13
Number of secondary log files     (LOGSECOND) = 4
Changed path to log files         (NEWLOGPATH) =
Path to log files                 = D:\DB2\NODE0000\SQL00003\LOGSTREAM0000\
Overflow log path                 (OVERFLOWLOGPATH) =
Mirror log path                   (MIRRORLOGPATH) =
First active log file             =
Block log on disk full            (BLK_LOG_DSK_FUL) = NO
Block non logged operations       (BLOCKNONLOGGED) = NO
Percent 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) = 520
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
HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0
HADR spool log data limit (4KB)   (HADR_SPOOL_LIMIT) = 0
HADR log replay delay (seconds)   (HADR_REPLAY_DELAY) = 0
First log archive method          (LOGARCHMETH1) = OFF
Archive compression for logarchmeth1 (LOGARCHCOMPR1) = OFF
Options for logarchmeth1          (LOGARCHOPT1) =
Second log archive method         (LOGARCHMETH2) = OFF
Archive compression for logarchmeth2 (LOGARCHCOMPR2) = 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
Auto deletion of recovery objects (AUTO_DEL_REC_OBJ) = OFF

TSM management class             (TSM_MGMTCLASS) =
TSM node name                    (TSM_NODENAME) =
TSM owner                        (TSM_OWNER) =
```

## GET DATABASE CONFIGURATION

```

TSM password                (TSM_PASSWORD) =
Automatic maintenance       (AUTO_MAINT) = ON
Automatic database backup   (AUTO_DB_BACKUP) = OFF
Automatic table maintenance (AUTO_TBL_MAINT) = ON
Automatic runstats          (AUTO_RUNSTATS) = ON
Automatic statement statistics (AUTO_STMT_STATS) = ON
Automatic statistics profiling (AUTO_STATS_PROF) = OFF
Automatic profile updates   (AUTO_PROF_UPD) = OFF
Automatic reorganization    (AUTO_REORG) = OFF

Auto-Revalidation          (AUTO_REVAL) = DEFERRED
Currently Committed         (CUR_COMMIT) = ON
CHAR output with DECIMAL input (DEC_TO_CHAR_FMT) = NEW
Enable XML Character operations (ENABLE_XMLCHAR) = YES
WLM Collection Interval     (WLM_COLLECT_INT) = 0
Monitor Collect Settings

Request metrics             (MON_REQ_MATRICS) = BASE
Activity metrics            (MON_ACT_MATRICS) = BASE
Object metrics              (MON_OBJ_MATRICS) = BASE
Unit of work events         (MON_UOW_DATA) = NONE
Lock timeout events         (MON_LOCKTIMEOUT) = NONE
Deadlock events            (MON_DEADLOCK) = WITHOUT_HIST
Lock wait events            (MON_LOCKWAIT) = NONE
Lock wait event threshold   (MON_LW_THRESH) = 5000000

SMTP Server                 (SMTP_SERVER) =
  
```

The following example shows a portion of the output of the command when you specify the **SHOW DETAIL** option on a Windows operating system. 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

Description	Parameter	Current Value	Delayed Value
Database configuration release level		= 0x0d00	
Database release level		= 0x0d00	
Database territory		= US	
Database code page		= 1208	
Database code set		= utf-8	
Database country/region code		= 1	
Database collating sequence		= IDENTITY	IDENTITY
Alternate collating sequence	(ALT_COLLATE)	=	
Number compatibility		= OFF	
Varchar2 compatibility		= OFF	
Date compatibility		= OFF	
Database page size		= 4096	4096
Dynamic SQL Query management	(DYN_QUERY_MGMT)	= DISABLE	DISABLE
Statement concentrator	(STMT_CONC)	= OFF	OFF
Discovery support for this database	(DISCOVER_DB)	= ENABLE	ENABLE
Restrict access		= NO	
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
Decimal floating point rounding mode	(DECFLT_ROUNDING)	= ROUND_HALF_EVEN	ROUND_HALF_EVEN

## GET DATABASE CONFIGURATION

```

Backup pending                                     = NO

All committed transactions have been written to disk = No
Rollforward pending                               = NO
Restore pending                                   = NO

Multi-page file allocation enabled                 = YES

Log retain for recovery status                    = NO
User exit for logging status                      = NO

Self tuning memory                               (SELF_TUNING_MEM) = OFF                OFF
Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC(282400) AUTOMATIC(282400)
Database memory threshold (DB_MEM_THRESH) = 10                10
Max storage for lock list (4KB) (LOCKLIST) = 4096              4096
Percent. of lock lists per application (MAXLOCKS) = 10            10
Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)          (MAXAPPLS*8)
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 5000    5000
Sort list heap (4KB) (SORTHEAP) = 256                      256

Database heap (4KB) (DBHEAP) = AUTOMATIC(1200)              AUTOMATIC(1200)
Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*5)    (MAXAPPLS*5)
Log buffer size (4KB) (LOGBUFSZ) = 256                      256
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000             5000
Buffer pool size (pages) (BUFFPAGE) = 200                  200
SQL statement heap (4KB) (STMHEAP) = AUTOMATIC(6402)        AUTOMATIC(4096)
Default application heap (4KB) (APPLHEAPSZ) = AUTOMATIC(256) AUTOMATIC(256)
Application Memory Size (4KB) (APPL_MEMORY) = AUTOMATIC(40016) AUTOMATIC(40000)
Statistics heap size (4KB) (STAT_HEAP_SZ) = AUTOMATIC(4384)  AUTOMATIC(4384)

Interval for checking deadlock (ms) (DLCHKTIME) = 10000      10000
Lock timeout (sec) (LOCKTIMEOUT) = -1                       -1

Changed pages threshold (CHNGPGS_THRESH) = 60               60
Number of asynchronous page cleaners (NUM_IOCLEANERS) = AUTOMATIC(3) AUTOMATIC(3)
Number of I/O servers (NUM_IOSERVERS) = AUTOMATIC(3)        AUTOMATIC(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(40) AUTOMATIC(40)
Average number of active applications (AVG_APPLS) = AUTOMATIC(1) AUTOMATIC(1)
Max DB files open per application (MAXFILOP) = 61440         61440

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 = D:\DB2\NODE0000 D:\DB2\NODE0000
\SQL00001\LOGSTREAM0000\ \SQL00001\LOGSTREAM0000\

Overflow log path (OVERFLOWLOGPATH) =
Mirror log path (MIRRORLOGPATH) =
First active log file =
Block log on disk full (BLK_LOG_DSK_FUL) = NO                NO
Percent 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 ckcpt (SOFTMAX) = 100  100
Log retain for recovery enabled (LOGRETAIN) = OFF             OFF
User exit for logging enabled (USEREXIT) = OFF                OFF

HADR database role = STANDARD                                STANDARD

```



## GET DATABASE CONFIGURATION

```

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
HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0          0
HADR spool log data limit (4KB)     (HADR_SPOOL_LIMIT) = 0          0
HADR log replay delay (seconds)      (HADR_REPLAY_DELAY) = 0          0

First log archive method        (LOGARCHMETH1) = OFF          OFF
Archive compression for logarchmeth1 (LOGARCHCOMPR1) = OFF          OFF
Options for logarchmeth1        (LOGARCHOPT1) =
Second log archive method        (LOGARCHMETH2) = OFF          OFF
Archive compression for logarchmeth2 (LOGARCHCOMPR2) = 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)
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
Auto deletion of recovery objects  (AUTO_DEL_REC_OBJ) = OFF      OFF

TSM management class            (TSM_MGMTCLASS) =
TSM node name                   (TSM_NODENAME) =
TSM owner                       (TSM_OWNER) =
TSM password                    (TSM_PASSWORD) =

Automatic maintenance           (AUTO_MAINT) = ON           ON
  Automatic database backup      (AUTO_DB_BACKUP) = OFF      OFF
  Automatic table maintenance    (AUTO_TBL_MAINT) = ON       ON
  Automatic runstats             (AUTO_RUNSTATS) = ON        ON
  Automatic statement statistics (AUTO_STMT_STATS) = ON       ON
  Automatic statistics profiling (AUTO_STATS_PROF) = OFF     OFF
  Automatic profile updates      (AUTO_PROF_UPD) = OFF       OFF
  Automatic reorganization       (AUTO_REORG) = OFF          OFF
Auto-Revalidation              (AUTO_REVAL) = DEFERRED     DEFERRED
Currently Committed             (CUR_COMMIT) = ON           ON
CHAR output with DECIMAL input  (DEC_TO_CHAR_FMT) = NEW      NEW
Enable XML Character operations  (ENABLE_XMLCHAR) = YES       YES
WLM Collection Interval         (WLM_COLLECT_INT) = 0        0

Monitor Collect Settings
Request metrics                 (MON_REQ_MATRICS) = BASE     BASE
Activity metrics                (MON_ACT_MATRICS) = BASE     BASE
Object metrics                  (MON_OBJ_MATRICS) = BASE     BASE
Unit of work events             (MON_UOW_DATA) = NONE        NONE
Lock timeout events             (MON_LOCKTIMEOUT) = NONE     NONE
Deadlock events                 (MON_DEADLOCK) = WITHOUT_HIST WITHOUT_HIST
Lock wait events                (MON_LOCKWAIT) = NONE        NONE
Lock wait event threshold       (MON_LW_THRESH) = 0          5000000
SMTP Server                     (SMTP_SERVER) =

```

### Usage notes

If the database configuration file is invalid, 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.

## GET DATABASE CONFIGURATION

To retrieve information from all database partitions, use the SYSIBMADM.DBCFG administrative view.

The configuration parameter value returned by issuing the **GET DATABASE CONFIGURATION** command may vary slightly from the configuration parameter value allocated in DISK.

## 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 database manager configuration operations, but is required to perform remote database manager 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.

This is a default clause when operating in the CLPPlus interface. **SHOW DETAIL** need not be called when using CLPPlus processor.

### Examples

Both node type and platform determine which configuration parameters are listed.

The following example is a sample output after running the **GET DATABASE MANAGER CONFIGURATION** command on a Linux operating system with a DB2 pureScale instance:

```

Database Manager Configuration

Node type = Enterprise Server Edition with local and remote clients

Database manager configuration release level          = 0x0e00

CPU speed (millisec/instruction)                    (CPUSPEED) = 4.000000e-05
Communications bandwidth (MB/sec)                   (COMM_BANDWIDTH) = 1.000000e+02

Max number of concurrently active databases          (NUMDB) = 1
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) = /home/db2inst1/sqllib/java/jdk64

Diagnostic error capture level                       (DIAGLEVEL) = 3
Notify Level                                         (NOTIFYLEVEL) = 3
Diagnostic data directory path                       (DIAGPATH) = /home/hotel67/db2inst1/sqllib/db2dump/ $m
  
```

## GET DATABASE MANAGER CONFIGURATION

Current member resolved DIAGPATH = /home/hotel67/db2inst1/sqlllib/db2dump/DIAG0000  
Alternate diagnostic data directory path (ALT\_DIAGPATH) = /home/hotel67/db2inst1/sqlllib/db2altdump/ \$m  
Current member resolved ALT\_DIAGPATH = /home/hotel67/db2inst1/sqlllib/db2altdump/DIAG0000  
Size of rotating db2diag & notify logs (MB) (DIAGSIZE) = 0

### Default database monitor switches

Buffer pool (DFT\_MON\_BUFPOOL) = OFF  
Lock (DFT\_MON\_LOCK) = OFF  
Sort (DFT\_MON\_SORT) = OFF  
Statement (DFT\_MON\_STMT) = OFF  
Table (DFT\_MON\_TABLE) = OFF  
Timestamp (DFT\_MON\_TIMESTAMP) = ON  
Unit of work (DFT\_MON\_UOW) = OFF  
Monitor health of instance and databases (HEALTH\_MON) = OFF

SYSADM group name (SYSADM\_GROUP) = DB2ADMIN  
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  
Cluster manager = TSA

Database manager authentication (AUTHENTICATION) = SERVER  
Alternate authentication (ALTERNATE\_AUTH\_ENC) = NOT\_SPECIFIED  
Cataloging allowed without authority (CATALOG\_NOAUTH) = NO  
Trust all clients (TRUST\_ALLCLNTS) = YES  
Trusted client authentication (TRUST\_CLNTAUTH) = CLIENT  
Bypass federated authentication (FED\_NOAUTH) = NO

Default database path (DFTDBPATH) = /home/db2inst1

Database monitor heap size (4KB) (MON\_HEAP\_SZ) = AUTOMATIC(90)  
Java Virtual Machine heap size (4KB) (JAVA\_HEAP\_SZ) = 2048  
Audit buffer size (4KB) (AUDIT\_BUF\_SZ) = 0  
Size of instance shared memory (4KB) (INSTANCE\_MEMORY) = AUTOMATIC(1705741)  
Instance memory for restart light (%) (RSTRT\_LIGHT\_MEM) = AUTOMATIC(10)  
Backup buffer default size (4KB) (BACKBUFSZ) = 1024  
Restore buffer default size (4KB) (RESTBUFSZ) = 1024

Agent stack size (AGENT\_STACK\_SZ) = 1024  
Sort heap threshold (4KB) (SHEAPTHRES) = 0

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  
Agent pool size (NUM\_POOLAGENTS) = AUTOMATIC(100)  
Initial number of agents in pool (NUM\_INITAGENTS) = 0  
Max number of coordinating agents (MAX\_COORDAGENTS) = AUTOMATIC(200)  
Max number of client connections (MAX\_CONNECTIONS) = AUTOMATIC(MAX\_COORDAGENTS)

Keep fenced process (KEEPFENCED) = YES  
Number of pooled fenced processes (FENCED\_POOL) = AUTOMATIC(MAX\_COORDAGENTS)  
Initial number of fenced processes (NUM\_INITFENCED) = 0

Index re-creation time and redo index build (INDEXREC) = RESTART

## GET DATABASE MANAGER CONFIGURATION

```

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

SSL server keydb file                  (SSL_SVR_KEYDB) =
SSL server stash file                  (SSL_SVR_STASH) =
SSL server certificate label           (SSL_SVR_LABEL) =
SSL service name                       (SSL_SVCENAME) =
SSL cipher specs                       (SSL_CIPHERSPECS) =
SSL versions                           (SSL_VERSIONS) =
SSL client keydb file                  (SSL_CLNT_KEYDB) =
SSL client stash file                  (SSL_CLNT_STASH) =

Maximum query degree of parallelism    (MAX_QUERYDEGREE) = ANY
Enable intra-partition parallelism     (INTRA_PARALLEL) = NO

Maximum Asynchronous TQs per query    (FEDERATED_ASYNC) = 0

No. of int. communication buffers(4KB) (FCM_NUM_BUFFERS) = AUTOMATIC(4096)
No. of int. communication channels     (FCM_NUM_CHANNELS) = AUTOMATIC(2048)
Node connection elapse time (sec)      (CONN_ELAPSE) = 3
Max number of node connection retries (MAX_CONNRETRIES) = 3
Max time difference between nodes (min) (MAX_TIME_DIFF) = 1

db2start/db2stop timeout (min)        (START_STOP_TIME) = 10

CF Server Configuration:
Memory size (4KB)                      (CF_MEM_SZ) = AUTOMATIC
Number of worker threads                (CF_NUM_WORKERS) = AUTOMATIC
Number of connections                  (CF_NUM_CONNS) = AUTOMATIC
Diagnostic error capture level          (CF_DIAGLEVEL) = 2
Diagnostic data directory path          (CF_DIAGPATH) = /home/hotel67/db2inst1/sqllib/db2dump/ $m
Current member resolved CF_DIAGPATH    = /home/hotel67/db2inst1/sqllib/db2dump/DIAG0000/

```

The following output sample shows the information displayed when you specify the **SHOW DETAIL** option. The value that appears in the Delayed Value column is the value that will be in effect the next time you start the database manager instance.

```

Database Manager Configuration

Node type = Enterprise Server Edition with local and remote clients

Description          Parameter    Current Value    Delayed Value
-----
Database manager configuration release level      = 0x0e00

CPU speed (millisec/instruction)                (CPUSPEED) = 4.000000e-05    4.000000e-05
Communications bandwidth (MB/sec)                (COMM_BANDWIDTH) = 1.000000e+02    1.000000e+02

Max number of concurrently active databases       (NUMDB) = 1    1
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) = /home/db2inst1/sqllib/java/jdk64 /home/db2inst1/sqllib/java/jdk64

Diagnostic error capture level                    (DIAGLEVEL) = 3    3
Notify Level                                     (NOTIFYLEVEL) = 3    3
Diagnostic data directory path                   (DIAGPATH) = /home/hotel67/db2inst1/sqllib/db2dump/ $m
Current member resolved DIAGPATH                 = /home/hotel67/db2inst1/sqllib/db2dump/DIAG0000
Alternate diagnostic data directory path (ALT_DIAGPATH) = /home/hotel67/db2inst1/sqllib/db2altdump/ $m
Current member resolved ALT_DIAGPATH             = /home/hotel67/db2inst1/sqllib/db2altdump/DIAG0000
Size of rotating db2diag & notify logs (MB)     (DIAGSIZE) = 0    0

Default database monitor switches
Buffer pool                                     (DFT_MON_BUFPOOL) = OFF    OFF
Lock                                             (DFT_MON_LOCK) = OFF    OFF

```

## GET DATABASE MANAGER CONFIGURATION

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) = OFF	OFF
SYSADM group name	(SYSADM_GROUP) = DB2ADMIN	DB2ADMIN
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_SPECIFIED	NOT_SPECIFIED
Cluster manager	= TSA	TSA
Database manager authentication	(AUTHENTICATION) = SERVER	SERVER
Alternate authentication	(ALTERNATE_AUTH_ENC) = NOT_SPECIFIED	NOT_SPECIFIED
Cataloging allowed without authority	(CATALOG_NOAUTH) = NO	NO
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/db2inst1	/home/db2inst1
Database monitor heap size (4KB)	(MON_HEAP_SZ) = AUTOMATIC(90)	AUTOMATIC(90)
Java Virtual Machine heap size (4KB)	(JAVA_HEAP_SZ) = 2048	2048
Audit buffer size (4KB)	(AUDIT_BUF_SZ) = 0	0
Size of instance shared memory (4KB)	(INSTANCE_MEMORY) = AUTOMATIC(1705741)	AUTOMATIC(1705741)
Instance memory for restart light (%)	(RSTRT_LIGHT_MEM) = AUTOMATIC(10)	AUTOMATIC(10)
Agent stack size	(AGENT_STACK_SZ) = 1024	1024
Sort heap threshold (4KB)	(SHEAPTHRES) = 0	0
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
Workload impact by throttled utilities	(UTIL_IMPACT_LIM) = 10	10
Priority of agents	(AGENTPRI) = SYSTEM	SYSTEM
Agent pool size	(NUM_POOLAGENTS) = AUTOMATIC(100)	AUTOMATIC(100)
Initial number of agents in pool	(NUM_INITAGENTS) = 0	0
Max number of coordinating agents	(MAX_COORDAGENTS) = AUTOMATIC(200)	AUTOMATIC(200)
Max number of client connections	(MAX_CONNECTIONS) = AUTOMATIC(MAX_COORDAGENTS)	AUTOMATIC(MAX_COORDAGENTS)
Keep fenced process	(KEEPFENCED) = YES	YES
Number of pooled fenced processes	(FENCED_POOL) = AUTOMATIC(MAX_COORDAGENTS)	AUTOMATIC(MAX_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
SSL server keydb file	(SSL_SVR_KEYDB) =	
SSL server stash file	(SSL_SVR_STASH) =	
SSL server certificate label	(SSL_SVR_LABEL) =	
SSL service name	(SSL_SVCENAME) =	
SSL cipher specs	(SSL_CIPHERSPECS) =	
SSL versions	(SSL_VERSIONS) =	
SSL client keydb file	(SSL_CLNT_KEYDB) =	
SSL client stash file	(SSL_CLNT_STASH) =	
Maximum query degree of parallelism	(MAX_QUERYDEGREE) = ANY	ANY
Enable intra-partition parallelism	(INTRA_PARALLEL) = NO	NO
Maximum Asynchronous TQs per query	(FEDERATED_ASYNC) = 0	0

## GET DATABASE MANAGER CONFIGURATION

```
No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = AUTOMATIC(4096)          AUTOMATIC(4096)
No. of int. communication channels (FCM_NUM_CHANNELS) = AUTOMATIC(2048)          AUTOMATIC(2048)
Node connection elapse time (sec) (CONN_ELAPSE) = 3                            3
Max number of node connection retries (MAX_CONNRETRIES) = 3                    3
Max time difference between nodes (min) (MAX_TIME_DIFF) = 1                    1

db2start/db2stop timeout (min) (START_STOP_TIME) = 10                          10

CF Server Configuration:
Memory size (4KB) (CF_MEM_SZ) = AUTOMATIC(131072)          AUTOMATIC(131072)
Number of worker threads (CF_NUM_WORKERS) = AUTOMATIC(1)   AUTOMATIC(1)
Number of connections (CF_NUM_CONNS) = AUTOMATIC(19)       AUTOMATIC(19)
Diagnostic error capture level (CF_DIAGLEVEL) = 2          2
Diagnostic data directory path (CF_DIAGPATH) = /home/hotel67/db2inst1/sql1lib/ $m
Current member resolved CF_DIAGPATH = /home/hotel67/db2inst1/sql1lib/db2dump/DIAG0000/
```

The following example is a sample output after running the **GET DATABASE MANAGER CONFIGURATION** command on a Windows operating system:

### Database Manager Configuration

Node type = Enterprise Server Edition with local and remote clients

```
Database manager configuration release level          = 0x0c00

Maximum total of files open (MAXTOTFILOP) = 16000
CPU speed (millisec/instruction) (CPUSPEED) = 4.251098e-007
Communications bandwidth (MB/sec) (COMM_BANDWIDTH) = 1.000000e+002

Max number of concurrently active databases (NUMDB) = 8
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) =

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
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) = IBMkrb5
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
Cluster manager (CLUSTER_MGR) =

Database manager authentication (AUTHENTICATION) = SERVER
Cataloging allowed without authority (CATALOG_NOAUTH) = NO
Trust all clients (TRUST_ALLCLNTS) = YES
Trusted client authentication (TRUST_CLNTAUTH) = CLIENT
Bypass federated authentication (FED_NOAUTH) = NO
```

## GET DATABASE MANAGER CONFIGURATION

```
Default database path                (DFTDBPATH) = C:

Database monitor heap size (4KB)     (MON_HEAP_SZ) = AUTOMATIC
Java Virtual Machine heap size (4KB) (JAVA_HEAP_SZ) = 2048
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

Agent stack size                     (AGENT_STACK_SZ) = 16
Minimum committed private memory (4KB) (MIN_PRIV_MEM) = 32
Private memory threshold (4KB)       (PRIV_MEM_THRESH) = 20000

Sort heap threshold (4KB)            (SHEAPTHRES) = 0

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
Agent pool size                      (NUM_POOLAGENTS) = AUTOMATIC
Initial number of agents in pool     (NUM_INITAGENTS) = 0
Max number of coordinating agents    (MAX_COORDAGENTS) = AUTOMATIC
Max number of client connections     (MAX_CONNECTIONS) = AUTOMATIC

Keep fenced process                  (KEEPFENCED) = YES
Number of pooled fenced processes    (FENCED_POOL) = AUTOMATIC
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) = KEON14
SPM log size                          (SPM_LOG_FILE_SZ) = 256
SPM resync agent limit                (SPM_MAX_RESYNC) = 20
SPM log path                          (SPM_LOG_PATH) =

NetBIOS Workstation name             (NNAME) =

TCP/IP Service name                   (SVCENAME) = db2c_DB2
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

Maximum Asynchronous TQs per query   (FEDERATED_ASYNC) = 0

No. of int. communication buffers(4KB) (FCM_NUM_BUFFERS) = AUTOMATIC
No. of int. communication channels   (FCM_NUM_CHANNELS) = AUTOMATIC
Node connection elapse time (sec)    (CONN_ELAPSE) = 10
Max number of node connection retries (MAX_CONNRETRIES) = 5
Max time difference between nodes (min) (MAX_TIME_DIFF) = 60

db2start/db2stop timeout (min)      (START_STOP_TIME) = 10
```



## GET DATABASE MANAGER CONFIGURATION

The following output sample shows the information displayed when you specify the **SHOW DETAIL** option on a Windows operating system. The value that appears in the Delayed Value column is the value that will be in effect the next time you start the database manager instance.

```
db2 => get dbm cfg show detail
```

### Database Manager Configuration

Node type = Enterprise Server Edition with local and remote clients

Description	Parameter	Current Value	Delayed Value
-----			
Database manager configuration release level		= 0x0c00	
Maximum total of files open	(MAXTOTFILOP)	= 16000	16000
CPU speed (millisec/instruction)	(CPUSPEED)	= 4.251098e-007	4.251098e-007
Communications bandwidth (MB/sec)	(COMM_BANDWIDTH)	= 1.000000e+002	1.000000e+002
Max number of concurrently active databases	(NUMDB)	= 8	8
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)	=	
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)	=	
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)	= IBMkrb5	IBMkrb5
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_SPECIFIED	NOT_SPECIFIED
Cluster manager	(CLUSTER_MGR)	=	
Database manager authentication	(AUTHENTICATION)	= SERVER	SERVER
Cataloging allowed without authority	(CATALOG_NOAUTH)	= NO	NO
Trust all clients	(TRUST_ALLCLNTS)	= YES	YES
Trusted client authentication	(TRUST_CLNTAUTH)	= CLIENT	CLIENT

## GET DATABASE MANAGER CONFIGURATION

Bypass federated authentication	(FED_NOAUTH) = NO	NO
Default database path	(DFTDBPATH) = C:	C:
Database monitor heap size (4KB)	(MON_HEAP_SZ) = AUTOMATIC(66)	AUTOMATIC(66)
Java Virtual Machine heap size (4KB)	(JAVA_HEAP_SZ) = 2048	2048
Audit buffer size (4KB)	(AUDIT_BUF_SZ) = 0	0
Size of instance shared memory (4KB)	(INSTANCE_MEMORY) = AUTOMATIC(73728)	AUTOMATIC(73728)
Backup buffer default size (4KB)	(BACKBUFSZ) = 1024	1024
Restore buffer default size (4KB)	(RESTBUFSZ) = 1024	1024
Agent stack size	(AGENT_STACK_SZ) = 16	16
Sort heap threshold (4KB)	(SHEAPTHRES) = 0	0
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
Agent pool size	(NUM_POOLAGENTS) = AUTOMATIC(100)	AUTOMATIC(100)
Initial number of agents in pool	(NUM_INITAGENTS) = 0	0
Max number of coordinating agents	(MAX_COORDAGENTS) = AUTOMATIC(200)	AUTOMATIC(200)
Max number of client connections	(MAX_CONNECTIONS) = AUTOMATIC(MAX_COORDAGENTS)	AUTOMATIC(MAX_COORDAGENTS)
Keep fenced process	(KEEPFENCED) = YES	YES
Number of pooled fenced processes	(FENCED_POOL) = AUTOMATIC(MAX_COORDAGENTS)	AUTOMATIC(MAX_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) = KEON14	KEON14
SPM log size	(SPM_LOG_FILE_SZ) = 256	256
SPM resync agent limit	(SPM_MAX_RESYNC) = 20	20
SPM log path	(SPM_LOG_PATH) =	
NetBIOS Workstation name	(NNAME) =	
TCP/IP Service name	(SVCENAME) = db2c_DB2	db2c_DB2
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
Maximum Asynchronous TQs per query	(FEDERATED_ASYNC) = 0	0
No. of int. communication buffers(4KB)	(FCM_NUM_BUFFERS) = AUTOMATIC(4096)	AUTOMATIC(4096)
No. of int. communication channels	(FCM_NUM_CHANNELS) = AUTOMATIC(2048)	AUTOMATIC(2048)
Node connection elapse time (sec)	(CONN_ELAPSE) = 10	10
Max number of node connection retries	(MAX_CONNRETRIES) = 5	5
Max time difference between nodes (min)	(MAX_TIME_DIFF) = 60	60
db2start/db2stop timeout (min)	(START_STOP_TIME) = 10	10

## 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 drop and re-create the instance 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 instance startup time and do not reflect any automatic increasing or decreasing that might have occurred during runtime.
- Configuration parameters **max\_connections**, **max\_coordagents** and **num\_poolagents** are set to AUTOMATIC.
- Configuration parameters **maxagents** and **maxcagents** are deprecated. The following deprecated functions are the result:
  - CLP and the db2CfgSet API will tolerate updates to these parameters, however these updates will be ignored by DB2.
  - CLP will no longer display these database configuration parameters when the client and server are on the DB2 v9.5 code base. If the server is DB2 v9.5, earlier version clients will see a value of 0 output for these parameters. If the client is DB2 v9.5, but the server is before DB2 v9.5, these parameters will be displayed with the assigned values.
  - db2CfgGet API will tolerate requests for SQLF\_KTN\_MAXAGENTS and SQLF\_KTN\_MAXCAGENTS, but they will return 0 if the server is DB2 v9.5.
  - The behavior of the db2AutoConfig API will depend on the **db2VersionNumber** passed in to the API. If the version is DB2 v9.5 or beyond, **maxagents** will not be returned, but for versions before this it will.
  - The **AUTOCONFIGURE** CLP command will display a value for **maxagents** with requests from an earlier version client (current and recommended value of 0). For current version client requests, **maxagents** will be displayed with an appropriate value.
  - The AUTOCONFIGURE ADMIN\_CMD will not return information about **maxagents** when the server is DB2 v9.5 and beyond.
  - Updates to **maxagents** or **maxcagents** through the ADMIN\_CMD will return successfully but have no effect on the server if the server is DB2 v9.5 or later.
  - Queries of database manager configuration parameters using the DBMCFG administrative view will not return rows for **maxagents** or **maxcagents** if the server is DB2 v9.5 or beyond.

In a future release, these configuration parameters may be removed completely.

### GET DATABASE MANAGER MONITOR SWITCHES

Displays the status of the database system monitor switches that are currently active. Monitor switches instruct the database manager to collect database activity information.

Each application using the database system monitor interface has its own set of monitor switches. A database system monitor switch is ON when any of the monitoring applications has turned it ON. This command is used to determine what data the database system monitor is currently collecting. It is possible that more data is being collected than is specified in the default monitor switches that are set in the database manager configuration because an application has a switch set to ON in its view.

#### Scope

This command is invoked on the currently attached member and only returns information for that member by default. The currently attached member defaults to the host on which the command is run. If there are multiple members per host, the currently attached member is the first member listed in the `db2nodes.cfg` file on that host.

To run the command on a member that is not the current attach member, specify the **AT MEMBER** option to run this command on a specific member, or specify the **GLOBAL** option to run this command on all members to receive an aggregated result.

To change the current attach member from which to run any subsequent commands, use the **ATTACH\_MEMBER** parameter of the **SET CLIENT** command. You must **DETACH** and then again **ATTACH** your application for this client setting to take effect.

#### Authorization

One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON

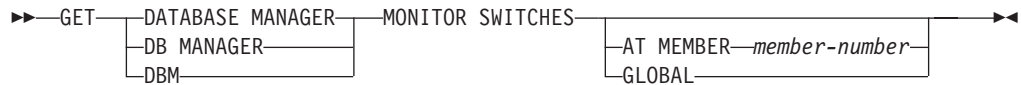
#### Required connection

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 MEMBER *member-number*

Specifies the member for which the status of the database manager monitor switches is to be displayed.

### GLOBAL

Returns the status of the database manager monitor switches from all members.

## Examples

The following is sample output from **GET DATABASE MANAGER MONITOR SWITCHES**:

DBM System Monitor Information Collected

```

Switch list for member 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**
- **apl\_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**

## GET DATABASE MANAGER MONITOR SWITCHES

- `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 previous versions:

- `DBPARTITIONNUM` or `NODE` can be substituted for `MEMBER`, except when the `DB2_ENFORCE_MEMBER_SYNTAX` registry variable is set to `ON`.

---

## 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 example 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:  $(\text{db2.sort\_heap\_allocated} / \text{SHEAPTHRES}) * 100$ . 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`.

### 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 email 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

```
▶▶ GET HEALTH NOTIFICATION CONTACT LIST ▶▶
    |_____|
    |_____|
    NOTIFICATION
```

#### Command Parameters

None

#### Examples

Issuing the command **GET NOTIFICATION LIST** results in a report similar to the following output:

Name	Type
Joe Brown	Contact
Support	Contact group



## 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.

**Important:** This command or API has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in DB2 pureScale environments. For more information, see “Health monitor has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

### Scope

This command is invoked on the currently attached member and only returns information for that member by default. The currently attached member defaults to the host on which the command is run. If there are multiple members per host, the currently attached member is the first member listed in the `db2nodes.cfg` file on that host.

To run the command on a member that is not the current attach member, specify the **AT MEMBER** option to run this command on a specific member, or specify the **GLOBAL** option to run this command on all members to receive an aggregated result.

To change the current attach member from which to run any subsequent commands, use the **ATTACH\_MEMBER** parameter of the **SET CLIENT** command. You must **DETACH** and then again **ATTACH** your application for this client setting to take effect.

### Authorization

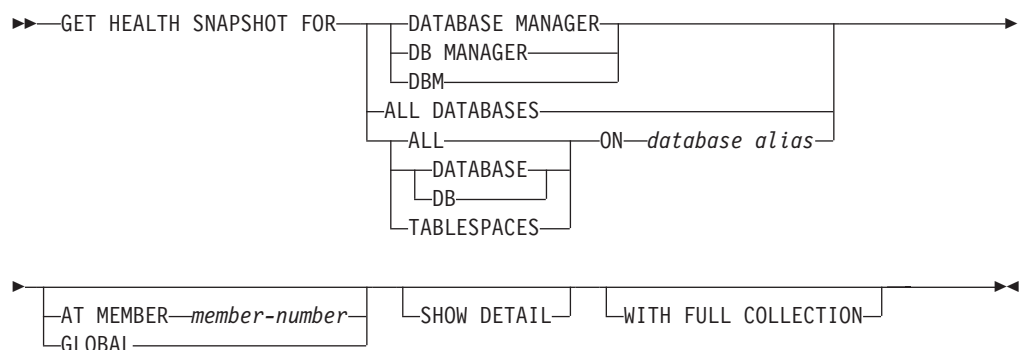
None

### 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 HEALTH SNAPSHOT

### 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 MEMBER *member-number*

Returns results for the member specified.

#### GLOBAL

Returns the health monitor status for all members.

#### 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 (that is, manual reorg or automation failed) will be displayed in a get health snapshot report.

### Examples

The following section 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 members in DB2 instance = 1
Start Database Manager timestamp = 02/17/2004 12:17:21.000119
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
```

## Usage notes

When the **GET HEALTH SNAPSHOT** command returns a recommendation to reorganize the data or index on a data partitioned table, the recommendation is only at the table level and not specific to any individual partitions of the table. Starting with DB2 Version 9.7 Fix Pack 1, the data or the partitioned indexes of a specific data partition can be reorganized using the **REORG INDEXES/TABLE** command or the db2Reorg API. To determine if only specific data partitions of a data partitioned table need to be reorganized, use the **REORGCHK** command to retrieve statistics and reorganization recommendations for the data partitions of the data partitioned table. Use the **REORG TABLE** or **REORG INDEXES ALL** command with the **ON DATA PARTITION** clause to reorganize the data or the partitioned indexes of a specific data partition.

## Compatibilities

For compatibility with previous versions:

- **DBPARTITIONNUM** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.

### 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 example is sample output from **GET INSTANCE**:

The current database manager instance is: smith

## GET MONITOR SWITCHES

Displays the status of the database system monitor switches for the current session. Monitor switches instruct the database 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.

### Scope

This command is invoked on the currently attached member and only returns information for that member by default. The currently attached member defaults to the host on which the command is run. If there are multiple members per host, the currently attached member is the first member listed in the `db2nodes.cfg` file on that host.

To run the command on a member that is not the current attach member, specify the **AT MEMBER** option to run this command on a specific member, or specify the **GLOBAL** option to run this command on all members to receive an aggregated result.

To change the current attach member from which to run any subsequent commands, use the **ATTACH\_MEMBER** parameter of the **SET CLIENT** command. You must **DETACH** and then again **ATTACH** your application for this client setting to take effect.

### Authorization

One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- 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

```

▶▶ GET MONITOR SWITCHES [ AT MEMBER member-number ] [ GLOBAL ]

```

### Command parameters

**AT MEMBER** *member-number*

Specifies the member for which the status of the monitor switches is to be displayed.

## GET MONITOR SWITCHES

### GLOBAL

Returns the status of the monitor switches from all members.

### Examples

The following is sample output from **GET MONITOR SWITCHES**:

#### Monitor Recording Switches

```
Switch list for member 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 previous versions:

- `DBPARTITIONNUM` or `NODE` can be substituted for `MEMBER`, except when the `DB2_ENFORCE_MEMBER_SYNTAX` registry variable is set to `ON`.

## 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.

**Important:** This command or API has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in DB2 pureScale environments. For more information, see “Health monitor has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

### Scope

This command is invoked on the currently attached member and only returns information for that member by default. The currently attached member defaults to the host on which the command is run. If there are multiple members per host, the currently attached member is the first member listed in the `db2nodes.cfg` file on that host.

To run the command on a member that is not the current attach member, specify the **AT MEMBER** option to run this command on a specific member, or specify the **GLOBAL** option to run this command on all members to receive an aggregated result.

To change the current attach member from which to run any subsequent commands, use the **ATTACH\_MEMBER** parameter of the **SET CLIENT** command. You must **DETACH** and then again **ATTACH** your application for this client setting to take effect.

In a partitioned database or DB2 pureScale environment, this command can be invoked from any member defined in the `db2nodes.cfg` file. It acts only on that member 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 MEMBER──member-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 *tblspacename*

Returns recommendation for a health indicator that has entered an alert state on the specified table space and database.

### CONTAINER *containername*

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.

### ON *database-alias*

Specifies a database.

### AT MEMBER *member-number*

Specifies the member number at which the health indicator has entered an alert state. If a member number is not specified and **GLOBAL** is not specified, the command will return information for the currently connected member.

### GLOBAL

Retrieves recommendations for the specified health indicator across all members. In cases where the recommendations are the same on different members, those recommendations are returned as a single set of recommendations that solve the health indicator on the affected members.

## 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  $(\text{pool\_cur\_size} / (4096 * U))$  where `U` is the required utilization rate. For example, if your required utilization rate is 60% of the warning

## GET RECOMMENDATIONS FOR HEALTH INDICATOR

threshold level, which you have set at 75%, then  
 $U = 0.6 * 0.75 = 0.45$  (or 45%).

Execute the following commands at the DB2 server:

```
CONNECT TO SAMPLE;  
UPDATE DB CFG USING DBHEAP 149333;  
CONNECT_RESET;  
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 about 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.

### 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.

When the **GET RECOMMENDATIONS FOR HEALTH INDICATOR** command returns a recommendation to reorganize the data or index on a data partitioned table, the recommendation is only at the table level and not specific to any individual data partitions of the table. Starting with DB2 Version 9.7 Fix Pack 1, the data or the partitioned indexes of a specific data partition can be reorganized using the **REORG INDEXES/TABLE** command or the db2Reorg API. To determine if only specific data partitions of a data partitioned table need to be reorganized, use the **REORGCHK** command to retrieve statistics and reorganization recommendations for the data partitions of the data partitioned table. Use the **REORG TABLE** or **REORG INDEXES ALL** command with the **ON DATA PARTITION** clause to reorganize the data or the partitioned indexes of a specific data partition.

## GET ROUTINE

Retrieves a routine SQL Archive (SAR) file for a specified SQL routine.

### Authorization

EXECUTE privilege on SYSFUN.GET\_ROUTINE\_SAR procedure

### 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" is SQLCODE

## GET ROUTINE

and "42704" is SQLSTATE, 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.

---

## 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

This command is invoked on the currently attached member and only returns information for that member by default. The currently attached member defaults to the host on which the command is run. If there are multiple members per host, the currently attached member is the first member listed in the `db2nodes.cfg` file on that host.

To run the command on a member that is not the current attach member, specify the **AT MEMBER** option to run this command on a specific member, or specify the **GLOBAL** option to run this command on all members to receive an aggregated result.

To change the current attach member from which to run any subsequent commands, use the **ATTACH\_MEMBER** parameter of the **SET CLIENT** command. You must **DETACH** and then again **ATTACH** your application for this client setting to take effect.

### Authorization

one of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON

### Required connection

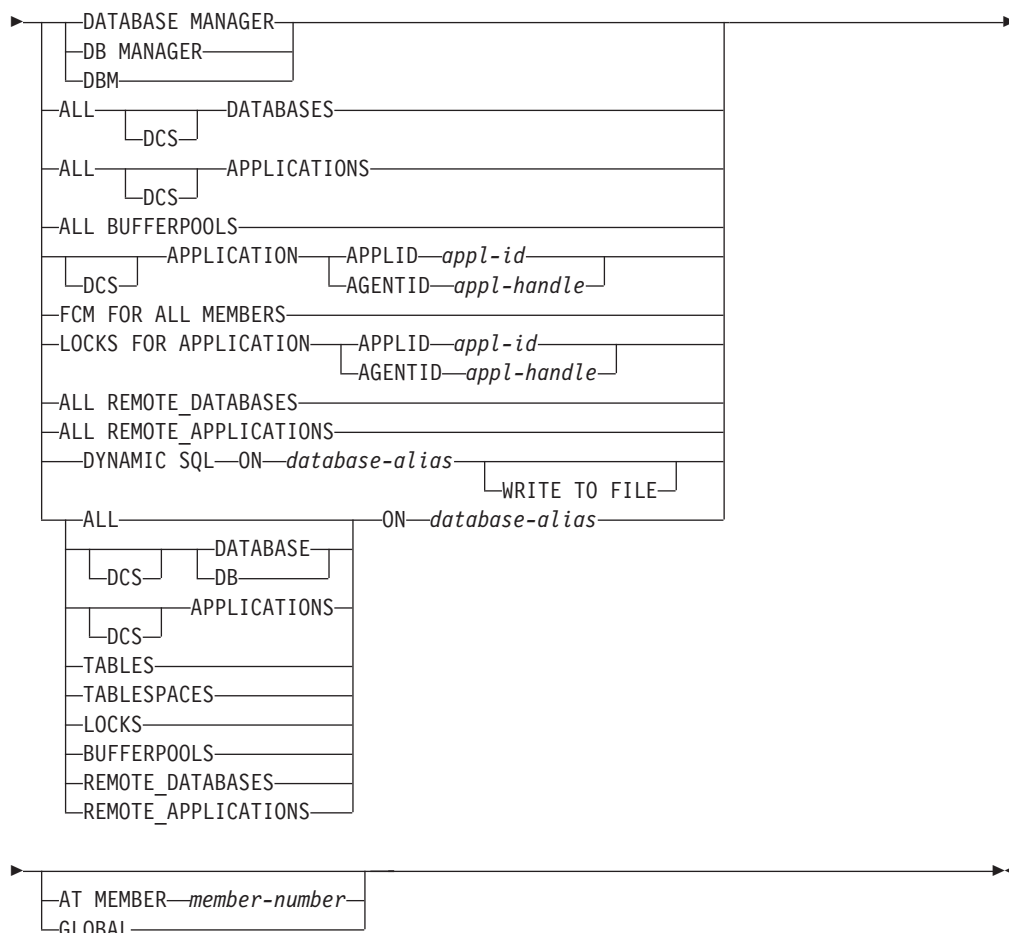
Instance. If there is no instance attachment, a default instance attachment is created.

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 member.

#### ALL APPLICATIONS

Provides information about all active applications that are connected to a database on the current member.

#### 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 MEMBERS**

Provides Fast Communication Manager (FCM) statistics between the member against which the **GET SNAPSHOT** command was issued and the other members in the partitioned database environment or in a DB2 pureScale 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 member.

**ALL REMOTE\_APPLICATIONS**

Provides information about all active remote applications that are connected to the current member.

**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 MEMBER** *member-number*

Returns results for the member specified.

**GLOBAL** Returns snapshot results for all members.

### Examples

- To request snapshot information about the database manager, issue:  
get snapshot for database manager

The following is a sample output listing from the preceding command:

#### Database Manager Snapshot

```
Node name                =
Node type                 = Enterprise Server Edition with local and remote clients
Instance name            = DB2
Number of members in DB2 instance = 1
Database manager status  = Active

Product name             = DB2 v9.5.0.535
Service level            = s070101 (NT32)

Private Sort heap allocated = 0
Private Sort heap high water mark = 0
Post threshold sorts     = Not Collected
Piped sorts requested    = 0
Piped sorts accepted     = 0

Start Database Manager timestamp = 01/10/2007 15:18:36.241035
Last reset timestamp       =
Snapshot timestamp       = 01/10/2007 15:28:26.989789

Remote connections to db manager = 3
Remote connections executing in db manager = 0
Local connections          = 1
Local connections executing in db manager = 0
Active local databases    = 1

High water mark for agents registered = 0
Agents registered         = 8
Idle agents               = 0

Committed private Memory (Bytes) = 8912896

Switch list for member number 0
Buffer Pool Activity Information (BUFFERPOOL) = OFF
Lock Information                 (LOCK) = ON 01/10/2007 15:22:43.145437
Sorting Information              (SORT) = OFF
SQL Statement Information       (STATEMENT) = OFF
Table Activity Information       (TABLE) = OFF
```



Take Timestamp Information (TIMESTAMP) = ON 01/10/2007 15:18:36.241035  
 Unit of Work Information (UOW) = OFF

Agents assigned from pool = 3  
 Agents created from empty pool = 11  
 Agents stolen from another application = 0  
 High water mark for coordinating agents = 9  
 Hash joins after heap threshold exceeded = 0  
 OLAP functions after heap threshold exceeded = 0

Total number of gateway connections = 0  
 Current number of gateway connections = 0  
 Gateway connections waiting for host reply = 0  
 Gateway connections waiting for client request = 0  
 Gateway connection pool agents stolen = 0

Node FCM information corresponds to = 1  
 Free FCM buffers = 13425  
 Total FCM buffers = 13425  
 Free FCM buffers low water mark = 13420  
 Maximum number of FCM buffers = 28640  
 Free FCM channels = 8036  
 Total FCM channels = 8055  
 Free FCM channels low water mark = 8036  
 Maximum number of FCM channels = 28640  
 Number of FCM nodes = 3

Node Number	Total Buffers Sent	Total Buffers Received	Connection Status
1	1	1	Active
2	1	0	Active
10	1	0	Active

Node FCM information corresponds to = 2  
 Free FCM buffers = 13425  
 Total FCM buffers = 13425  
 Free FCM buffers low water mark = 13420  
 Maximum number of FCM buffers = 28640  
 Free FCM channels = 8036  
 Total FCM channels = 8055  
 Free FCM channels low water mark = 8036  
 Maximum number of FCM channels = 28640  
 Number of FCM nodes = 3

Node Number	Total Buffers Sent	Total Buffers Received	Connection Status
1	0	1	Active
2	1	1	Active
10	0	0	Active

Node FCM information corresponds to = 10  
 Free FCM buffers = 13425  
 Total FCM buffers = 13425  
 Free FCM buffers low water mark = 13420  
 Maximum number of FCM buffers = 28640  
 Free FCM channels = 8036  
 Total FCM channels = 8055  
 Free FCM channels low water mark = 8036  
 Maximum number of FCM channels = 28640  
 Number of FCM nodes = 3

Node Number	Total Buffers Sent	Total Buffers Received	Connection Status
1	0	1	Active
2	1	1	Active
10	0	0	Active

## GET SNAPSHOT

1	0	1	Active
2	0	0	Active
10	1	1	Active

Memory usage for database manager:

Node number	= 0
Memory Pool Type	= Other Memory
Current size (bytes)	= 11534336
High water mark (bytes)	= 11599872
Configured size (bytes)	= 34275328
Node number	= 0
Memory Pool Type	= Database Monitor Heap
Current size (bytes)	= 65536
High water mark (bytes)	= 65536
Configured size (bytes)	= 327680
Node number	= 0
Memory Pool Type	= FCMBP Heap
Current size (bytes)	= 655360
High water mark (bytes)	= 655360
Configured size (bytes)	= 851968

- To request snapshot information about an application with agent ID 29:  
get snapshot for application agentid 29

The following is a sample output listing from the preceding command, assuming the lock and statement monitor switches are ON:

### Application Snapshot

Application handle	= 29
Application status	= Lock-wait
Status change time	= Not Collected
Application code page	= 819
Application country/region code	= 1
DUOW correlation token	= *LOCAL.jwr.070222182152
Application name	= db2bp
Application ID	= *LOCAL.jwr.070222182152
Sequence number	= 00001
TP Monitor client user ID	=
TP Monitor client workstation name	=
TP Monitor client application name	=
TP Monitor client accounting string	=
Connection request start timestamp	= 02/22/2007 13:21:52.587168
Connect request completion timestamp	= 02/22/2007 13:21:53.291779
Application idle time	=
CONNECT Authorization ID	= JWR
Client login ID	= jwr
Configuration NNAME of client	= gilera
Client database manager product ID	= SQL09050
Process ID of client application	= 843852
Platform of client application	= AIX 64BIT
Communication protocol of client	= Local Client
Inbound communication address	= *LOCAL.jwr
Database name	= SAMPLE
Database path	= /home/jwr/jwr/NODE0000/SQL00001/
Client database alias	= SAMPLE
Input database alias	=
Last reset timestamp	=
Snapshot timestamp	= 02/22/2007 13:22:39.766300
Authorization level granted	=
User authority:	

```

DBADM authority
CREATETAB authority
BINDADD authority
CONNECT authority
CREATE_NOT_FENC authority
LOAD authority
IMPLICIT_SCHEMA authority
CREATE_EXT_RT authority
QUIESCE_CONN authority
Group authority:
  SYSADM authority
  CREATETAB authority
  BINDADD authority
  CONNECT authority
  IMPLICIT_SCHEMA authority
Coordinator member number           = 0
Current member number               = 0
Coordinator agent process or thread ID = 1801
Current Workload ID                 = 1
Agents stolen                       = 0
Agents waiting on locks              = 1
Maximum associated agents            = 1
Priority at which application agents work = 0
Priority type                        = Dynamic

Lock timeout (seconds)              = -1
Locks held by application            = 4
Lock waits since connect             = 1
Time application waited on locks (ms) = 20268
Deadlocks detected                  = 0
Lock escalations                    = 0
Exclusive lock escalations           = 0
Number of Lock Timeouts since connected = 0
Total time UOW waited on locks (ms) = Not Collected

Total sorts                         = 0
Total sort time (ms)                = Not Collected
Total sort overflows                 = 0

Buffer pool data logical reads       = Not Collected
Buffer pool data physical reads       = Not Collected
Buffer pool temporary data logical reads = Not Collected
Buffer pool temporary data physical reads = Not Collected
Buffer pool data writes               = Not Collected
Buffer pool index logical reads        = Not Collected
Buffer pool index physical reads       = Not Collected
Buffer pool temporary index logical reads = Not Collected
Buffer pool temporary index physical reads = Not Collected
Buffer pool index writes               = Not Collected
Buffer pool xda logical reads          = Not Collected
Buffer pool xda physical reads         = Not Collected
Buffer pool temporary xda logical reads = Not Collected
Buffer pool temporary xda physical reads = Not Collected
Buffer pool xda writes                = Not Collected
Total buffer pool read time (milliseconds) = Not Collected
Total buffer pool write time (milliseconds) = Not Collected
Time waited for prefetch (ms)        = Not Collected
Unread prefetch pages                 = Not Collected
Direct reads                          = Not Collected
Direct writes                          = Not Collected
Direct read requests                  = Not Collected
Direct write requests                 = Not Collected
Direct reads elapsed time (ms)        = Not Collected
Direct write elapsed time (ms)        = Not Collected

Number of SQL requests since last commit = 3
Commit statements                     = 0

```

## GET SNAPSHOT

```
Rollback statements = 0
Dynamic SQL statements attempted = 3
Static SQL statements attempted = 0
Failed statement operations = 0
Select SQL statements executed = 1
Xquery statements executed = 0
Update/Insert/Delete statements executed = 0
DDL statements executed = 0
Inactive stmt history memory usage (bytes) = 0
Internal automatic rebinds = 0
Internal rows deleted = 0
Internal rows inserted = 0
Internal rows updated = 0
Internal commits = 1
Internal rollbacks = 0
Internal rollbacks due to deadlock = 0
Binds/precompiles attempted = 0
Rows deleted = 0
Rows inserted = 0
Rows updated = 0
Rows selected = 0
Rows read = 95
Rows written = 0

UOW log space used (Bytes) = Not Collected
Previous UOW completion timestamp = Not Collected
Elapsed time of last completed uow (sec.ms) = Not Collected
UOW start timestamp = Not Collected
UOW stop timestamp = Not Collected
UOW completion status = Not Collected

Open remote cursors = 0
Open remote cursors with blocking = 0
Rejected Block Remote Cursor requests = 0
Accepted Block Remote Cursor requests = 1
Open local cursors = 1
Open local cursors with blocking = 1
Total User CPU Time used by agent (s) = 0.019150
Total System CPU Time used by agent (s) = 0.001795
Host execution elapsed time = 0.012850

Package cache lookups = 2
Package cache inserts = 1
Application section lookups = 3
Application section inserts = 1
Catalog cache lookups = 11
Catalog cache inserts = 8
Catalog cache overflows = 0
Catalog cache high water mark = 0

Workspace Information

Shared high water mark = 0
Total shared overflows = 0
Total shared section inserts = 0
Total shared section lookups = 0
Private high water mark = 0
Total private overflows = 0
Total private section inserts = 0
Total private section lookups = 0

Most recent operation = Fetch
Cursor name = SQLCUR201
Most recent operation start timestamp = 02/22/2007 13:22:19.497439
Most recent operation stop timestamp =
Agents associated with the application = 1
```

## GET SNAPSHOT

```

Number of hash joins                = 0
Number of hash loops                 = 0
Number of hash join overflows        = 0
Number of small hash join overflows  = 0
Number of OLAP functions             = 0
Number of OLAP function overflows    = 0

Statement type                       = Dynamic SQL Statement
Statement                             = Fetch
Section number                       = 201
Application creator                   = NULLID
Package name                         = SQLC2G11
Consistency Token                    = AAAAAANBX
Package Version ID                   =
Cursor name                          = SQLCUR201
Statement member number               = 0
Statement start timestamp             = 02/22/2007 13:22:19.497439
Statement stop timestamp              =
Elapsed time of last completed stmt(sec.ms) = 0.000289
Total Statement user CPU time         = 0.002172
Total Statement system CPU time       = 0.001348
SQL compiler cost estimate in timerons = 14
SQL compiler cardinality estimate     = 57
Degree of parallelism requested       = 1
Number of agents working on statement = 1
Number of subagents created for statement = 1
Statement sorts                      = 0
Total sort time                      = 0
Sort overflows                       = 0
Rows read                            = 0
Rows written                         = 0
Rows deleted                         = 0
Rows updated                         = 0
Rows inserted                        = 0
Rows fetched                         = 0
Buffer pool data logical reads        = Not Collected
Buffer pool data physical reads       = Not Collected
Buffer pool temporary data logical reads = Not Collected
Buffer pool temporary data physical reads = Not Collected
Buffer pool index logical reads       = Not Collected
Buffer pool index logical reads       = Not Collected
Buffer pool temporary index logical reads = Not Collected
Buffer pool temporary index physical reads = Not Collected
Buffer pool xda logical reads         = Not Collected
Buffer pool xda physical reads        = Not Collected
Buffer pool temporary xda logical reads = Not Collected
Buffer pool temporary xda physical reads = Not Collected
Blocking cursor                      = YES
Dynamic SQL statement text:
select * from org

Agent process/thread ID               = 1801

Memory usage for application:

    Memory Pool Type                  = Application Heap
        Current size (bytes)          = 65536
        High water mark (bytes)       = 65536
        Configured size (bytes)       = 1048576

Agent process/thread ID               = 1801
Agent Lock timeout (seconds)          = -1
Memory usage for agent:

    Memory Pool Type                  = Other Memory
        Current size (bytes)          = 589824
        High water mark (bytes)       = 786432

```

## GET SNAPSHOT

```
Configured size (bytes) = 34359738368
ID of agent holding lock = 34
Application ID holding lock = *LOCAL.jwr.070222182158
Lock name = 0x0002000E00000000000000000054
Lock attributes = 0x00000000
Release flags = 0x00000001
Lock object type = Table
Lock mode = Exclusive Lock (X)
Lock mode requested = Intention Share Lock (IS)
Name of tablespace holding lock = USERSPACE1
Schema of table holding lock = JWR
Name of table holding lock = ORG
Data Partition Id of table holding lock = 0
Lock wait start timestamp = 02/22/2007 13:22:19.497833
```

- To request snapshot information about all of the databases:

```
get snapshot for all databases
```

The following is a sample output listing from the preceding command:

### Database Snapshot

```
Database name = SAMPLE
Database path = C:\DB2\NODE0000\SQL00001\
Input database alias =
Database status = Active
Catalog database partition number = 0
Catalog network node name =
Operating system running at database server= NT
Location of the database = Local
First database connect timestamp = 06/21/2007 14:46:49.771064
Last reset timestamp =
Last backup timestamp =
Snapshot timestamp = 06/21/2007 14:51:50.235993

Number of automatic storage paths = 1
Automatic storage path = C:
Node number = 0

High water mark for connections = 6
Application connects = 4
Secondary connects total = 4
Applications connected currently = 1
Appls. executing in db manager currently = 0
Agents associated with applications = 5
Maximum agents associated with applications= 6
Maximum coordinating agents = 6

Number of Threshold Violations = 0
Locks held currently = 0
Lock waits = 0
Time database waited on locks (ms) = Not Collected
Lock list memory in use (Bytes) = 2256
Deadlocks detected = 0
Lock escalations = 0
Exclusive lock escalations = 0
Agents currently waiting on locks = 0
Lock Timeouts = 0
Number of indoubt transactions = 0

Total Private Sort heap allocated = 0
Total Shared Sort heap allocated = 0
Shared Sort heap high water mark = 0
Post threshold sorts (shared memory) = Not Collected
Total sorts = 0
Total sort time (ms) = Not Collected
Sort overflows = 0
```

```

Active sorts = 0

Buffer pool data logical reads = Not Collected
Buffer pool data physical reads = Not Collected
Buffer pool temporary data logical reads = Not Collected
Buffer pool temporary data physical reads = Not Collected
Asynchronous pool data page reads = Not Collected
Buffer pool data writes = Not Collected
Asynchronous pool data page writes = Not Collected
Buffer pool index logical reads = Not Collected
Buffer pool index physical reads = Not Collected
Buffer pool temporary index logical reads = Not Collected
Buffer pool temporary index physical reads = Not Collected
Asynchronous pool index page reads = Not Collected
Buffer pool index writes = Not Collected
Asynchronous pool index page writes = Not Collected
Buffer pool xda logical reads = Not Collected
Buffer pool xda physical reads = Not Collected
Buffer pool temporary xda logical reads = Not Collected
Buffer pool temporary xda physical reads = Not Collected
Buffer pool xda writes = Not Collected
Asynchronous pool xda page reads = Not Collected
Asynchronous pool xda page writes = Not Collected
Total buffer pool read time (milliseconds) = Not Collected
Total buffer pool write time (milliseconds) = Not Collected
Total elapsed asynchronous read time = Not Collected
Total elapsed asynchronous write time = Not Collected
Asynchronous data read requests = Not Collected
Asynchronous index read requests = Not Collected
Asynchronous xda read requests = Not Collected
No victim buffers available = Not Collected
LSN Gap cleaner triggers = Not Collected
Dirty page steal cleaner triggers = Not Collected
Dirty page threshold cleaner triggers = Not Collected
Time waited for prefetch (ms) = Not Collected
Unread prefetch pages = Not Collected
Direct reads = Not Collected
Direct writes = Not Collected
Direct read requests = Not Collected
Direct write requests = Not Collected
Direct reads elapsed time (ms) = Not Collected
Direct write elapsed time (ms) = Not Collected
Database files closed = Not Collected
Vectored IOs = Not Collected
Pages from vectored IOs = Not Collected
Block IOs = Not Collected
Pages from block IOs = Not Collected

Host execution elapsed time = Not Collected

Commit statements attempted = 0
Rollback statements attempted = 0
Dynamic statements attempted = 6
Static statements attempted = 3
Failed statement operations = 0
Select SQL statements executed = 0
Xquery statements executed = 0
Update/Insert/Delete statements executed = 0
DDL statements executed = 0
Inactive stmt history memory usage (bytes) = 0

Internal automatic rebinds = 0
Internal rows deleted = 0
Internal rows inserted = 0
Internal rows updated = 0
Internal commits = 6
Internal rollbacks = 0

```

## GET SNAPSHOT

```
Internal rollbacks due to deadlock           = 0
Number of MDC table blocks pending cleanup = 0

Rows deleted                                = 0
Rows inserted                               = 0
Rows updated                               = 0
Rows selected                              = 0
Rows read                                   = 98
Binds/precompiles attempted               = 0

Log space available to the database (Bytes)= 20400000
Log space used by the database (Bytes)     = 0
Maximum secondary log space used (Bytes)   = 0
Maximum total log space used (Bytes)       = 0
Secondary logs allocated currently         = 0
Log pages read                             = 0
Log read time (sec.ns)                     = 0.000000004
Log pages written                           = 0
Log write time (sec.ns)                    = 0.000000004
Number write log IOs                       = 0
Number read log IOs                        = 0
Number partial page log IOs                = 0
Number log buffer full                     = 0
Log data found in buffer                   = 0
Appl id holding the oldest transaction     = 93
Log to be redone for recovery (Bytes)      = 0
Log accounted for by dirty pages (Bytes)   = 0

Node number                                 = 0
File number of first active log            = 0
File number of last active log             = 2
File number of current active log          = 0
File number of log being archived          = Not applicable

Package cache lookups                      = 6
Package cache inserts                      = 0
Package cache overflows                    = 0
Package cache high water mark (Bytes)     = 196608
Application section lookups                = 6
Application section inserts                = 0

Catalog cache lookups                      = 37
Catalog cache inserts                      = 10
Catalog cache overflows                    = 0
Catalog cache high water mark              = 65536
Catalog cache statistics size              = 0

Workspace Information

Shared high water mark                     = 0
Corresponding shared overflows             = 0
Total shared section inserts               = 0
Total shared section lookups               = 0
Private high water mark                    = 0
Corresponding private overflows            = 0
Total private section inserts              = 0
Total private section lookups              = 0

Number of hash joins                       = 0
Number of hash loops                       = 0
Number of hash join overflows              = 0
Number of small hash join overflows        = 0
Post threshold hash joins (shared memory) = 0
Active hash joins                          = 0

Number of OLAP functions                   = 0
Number of OLAP function overflows          = 0
```



```

Active OLAP functions                = 0

Statistic fabrications                = Not Collected
Synchronous runstats                 = Not Collected
Asynchronous runstats                 = Not Collected
Total statistic fabrication time (milliseconds) = Not Collected
Total synchronous runstats time (milliseconds) = Not Collected

Memory usage for database:

Node number                          = 0
  Memory Pool Type                    = Backup/Restore/Util Heap
    Current size (bytes)               = 65536
    High water mark (bytes)            = 65536
    Configured size (bytes)            = 20512768

Node number                          = 0
  Memory Pool Type                    = Package Cache Heap
    Current size (bytes)               = 196608
    High water mark (bytes)            = 196608
    Configured size (bytes)            = 402653184

Node number                          = 0
  Memory Pool Type                    = Other Memory
    Current size (bytes)               = 131072
    High water mark (bytes)            = 131072
    Configured size (bytes)            = 20971520

Node number                          = 0
  Memory Pool Type                    = Catalog Cache Heap
    Current size (bytes)               = 65536
    High water mark (bytes)            = 65536
    Configured size (bytes)            = 402653184

Node number                          = 0
  Memory Pool Type                    = Buffer Pool Heap
    Secondary ID                       = 1
    Current size (bytes)               = 2424832
    High water mark (bytes)            = 2424832
    Configured size (bytes)            = 402653184

Node number                          = 0
  Memory Pool Type                    = Buffer Pool Heap
    Secondary ID                       = System 32k buffer pool
    Current size (bytes)               = 851968
    High water mark (bytes)            = 851968
    Configured size (bytes)            = 402653184

Node number                          = 0
  Memory Pool Type                    = Buffer Pool Heap
    Secondary ID                       = System 16k buffer pool
    Current size (bytes)               = 589824
    High water mark (bytes)            = 589824
    Configured size (bytes)            = 402653184

Node number                          = 0
  Memory Pool Type                    = Buffer Pool Heap
    Secondary ID                       = System 8k buffer pool
    Current size (bytes)               = 458752
    High water mark (bytes)            = 458752
    Configured size (bytes)            = 402653184

Node number                          = 0
  Memory Pool Type                    = Buffer Pool Heap
    Secondary ID                       = System 4k buffer pool
    Current size (bytes)               = 393216

```

## GET SNAPSHOT

```
High water mark (bytes)           = 393216
Configured size (bytes)           = 402653184

Node number                        = 0
Memory Pool Type                   = Shared Sort Heap
Current size (bytes)               = 0
High water mark (bytes)           = 0
Configured size (bytes)           = 20512768

Node number                        = 0
Memory Pool Type                   = Lock Manager Heap
Current size (bytes)               = 327680
High water mark (bytes)           = 327680
Configured size (bytes)           = 393216

Node number                        = 0
Memory Pool Type                   = Database Heap
Current size (bytes)               = 10551296
High water mark (bytes)           = 10551296
Configured size (bytes)           = 12582912

Node number                        = 0
Memory Pool Type                   = Application Heap
Secondary ID                       = 97
Current size (bytes)               = 65536
High water mark (bytes)           = 65536
Configured size (bytes)           = 1048576

Node number                        = 0
Memory Pool Type                   = Application Heap
Secondary ID                       = 96
Current size (bytes)               = 65536
High water mark (bytes)           = 65536
Configured size (bytes)           = 1048576

Node number                        = 0
Memory Pool Type                   = Application Heap
Secondary ID                       = 95
Current size (bytes)               = 65536
High water mark (bytes)           = 65536
Configured size (bytes)           = 1048576

Node number                        = 0
Memory Pool Type                   = Application Heap
Secondary ID                       = 94
Current size (bytes)               = 65536
High water mark (bytes)           = 65536
Configured size (bytes)           = 1048576

Node number                        = 0
Memory Pool Type                   = Application Heap
Secondary ID                       = 93
Current size (bytes)               = 65536
High water mark (bytes)           = 65536
Configured size (bytes)           = 1048576

Node number                        = 0
Memory Pool Type                   = Applications Shared Heap
Current size (bytes)               = 65536
High water mark (bytes)           = 65536
Configured size (bytes)           = 20512768
```

User authority represents all authorizations and roles granted to the user, and Group authority represents all authorizations and roles granted to the group.

- 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

- To request fast communication manager (FCM) statistics, issue the following command:

get snapshot for fcm for all dbpartitionnums

The following is a sample output listing from the preceding command:

#### FCM Snapshot

```
Node FCM information corresponds to      = 1
Free FCM buffers                        = 10740
Total FCM buffers                       = 10740
Free FCM buffers low water mark         = 10740
Maximum number of FCM buffers           = 28640
Free FCM channels                       = 6265
Total FCM channels                      = 6265
Free FCM channels low water mark        = 6265
Maximum number of FCM channels          = 28640
Snapshot timestamp                      = 02/17/2010 15:54:57.094901
Number of FCM nodes                     = 3
```

Node Number	Total Buffers Sent	Total Buffers Received	Connection Status
1	2	2	Active
2	1	1	Active
10	1	1	Active

## 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 file 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. For a partitioned table, information for each reorganized data partition is returned.

- To obtain snapshot information from all members (which is different than the aggregate result of all members), the snapshot administrative views should be used.
- In a partitioned database environment or in a DB2 pureScale environment, specifying the command with the **GLOBAL** option will return a value for the High

## GET SNAPSHOT

water mark for connections parameter which represents the greatest high water mark for connections among all the members and not the sum of the individual high water marks of all the members. For example:

- Member 1 has 5 applications connected currently and the high water mark for connections is 5.
- Member 2 has 4 applications connected currently and the high water mark for connections is 6.

In the previous example, the High water mark for connections value is 6, and the Applications connected currently value is 9.

- If you are using storage groups, the snapshot monitor reports information only for the default storage group. To view information about the storage paths in all database storage groups use the ADMIN\_GET\_STORAGE\_PATHS table function.

### Compatibilities

For compatibility with previous versions:

- **DBPARTITIONNUM** or **NODE** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.
- **DBPARTITIONNUMS** or **NODES** can be substituted for **MEMBERS**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.
- The new registry variable in Version 9.5, **DB2\_SYSTEM\_MONITOR\_SETTINGS** impacts the behavior of monitoring the CPU usage on Linux. If you need to use the method of reading CPU usage that returns both system and user CPU usage times on Linux, perform one of the following actions.

On Linux on RHEL4 and SLES9:

```
db2set DB2_SYSTEM_MONITOR_SETTINGS=DISABLE_CPU_USAGE:FALSE
```

On Linux on RHEL5 and SLES10:

```
db2set DB2_SYSTEM_MONITOR_SETTINGS=OLD_CPU_USAGE:TRUE
```

---

## HELP

Permits the user to invoke help from the Information Center.

This command is not available on UNIX operating systems.

### Authorization

None

### Required connection

None

### Command syntax

▶—HELP—▶

### Examples

The following example shows how to use 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.

### Usage notes

The command line processor will not know if the command succeeds or fails, and cannot report error conditions.

---

## 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 | R

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** registry 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.

---

## 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.

Quick link to “File type modifiers for the import utility” on page 282.

### Authorization

- **IMPORT** using the **INSERT** option requires one of the following authorities:
  - DATAACCESS authority
  - 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 authorities:
  - DATAACCESS authority
  - 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 authorities:
  - DATAACCESS authority
  - 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 authorities:
  - DBADM authority
  - 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
    - CREATEIN 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 authorities:
  - DBADM authority
  - 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 authorities:
  - DATAACCESS authority
  - CONTROL privilege on every sub-table in the hierarchy

## IMPORT

- 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 meet these criteria:
  - It is part of the security policy protecting the table
  - It was granted to the session authorization ID for write access

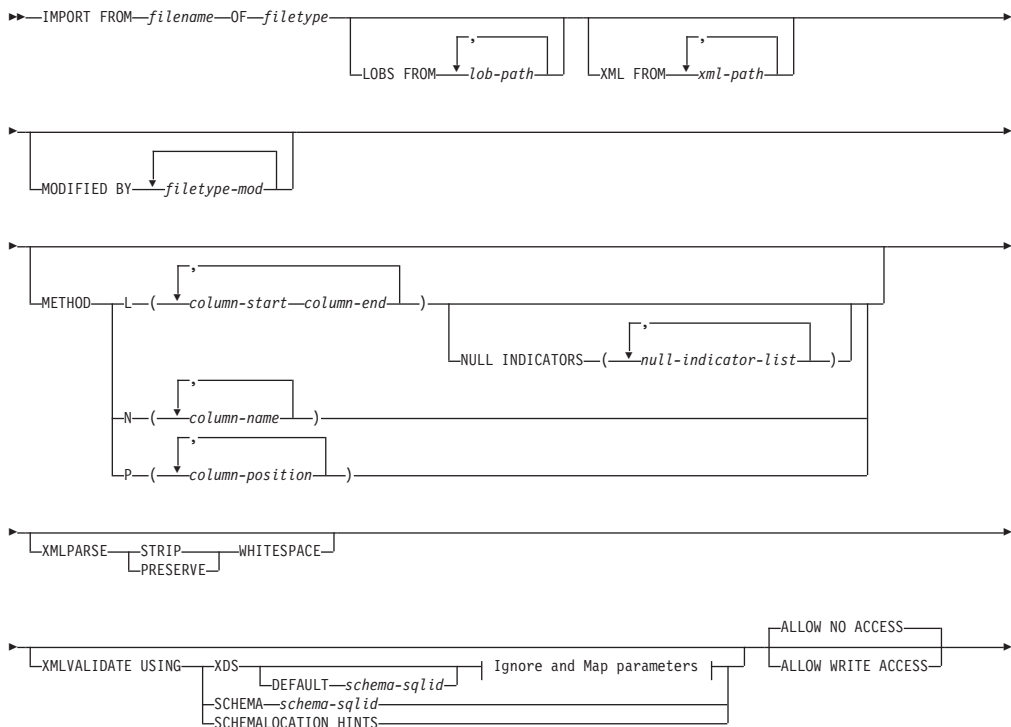
The label on the row to insert, the user's LBAC credentials, the security policy definition, and the LBAC rules determine the label on the row.

- If the **REPLACE** or **REPLACE\_CREATE** option is specified, the session authorization ID must have the authority to drop the table.
- To import data into a nickname, the session authorization ID must have the privilege to access and use a specified data source in pass-through mode.
- If the table has row access control activated, then **IMPORT REPLACE** on that table would require the ability to drop the table. Specifically, you must have either **CONTROL** or **DBADM** on the table.

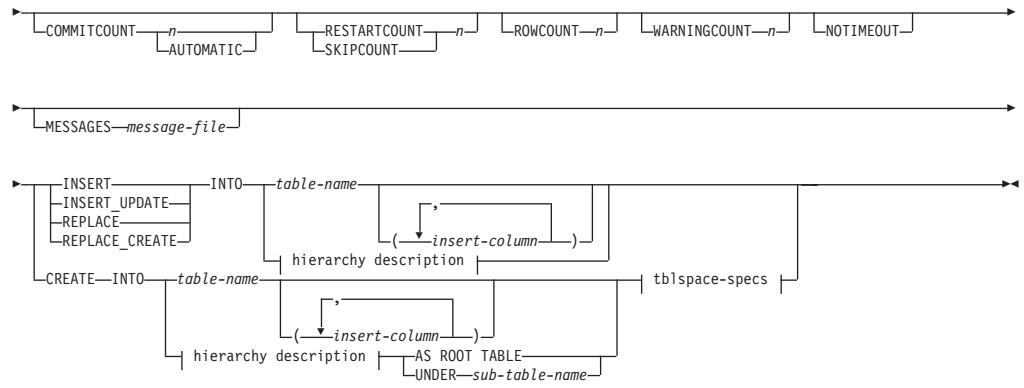
### Required connection

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.

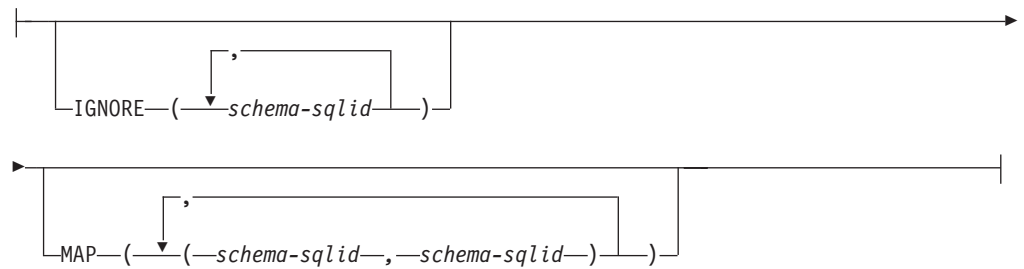
### Command syntax







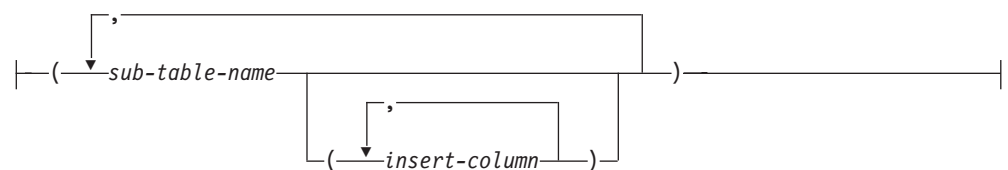
**Ignore and Map parameters:**



**hierarchy description:**



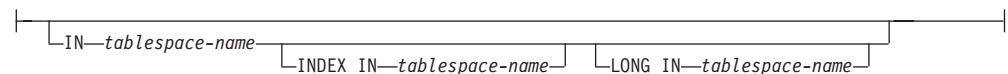
**sub-table-list:**



**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.

The ability of the import operation to avoid running out of active log space is affected by the DB2 registry variable **DB2\_FORCE\_APP\_ON\_MAX\_LOG**:

- If **DB2\_FORCE\_APP\_ON\_MAX\_LOG** is set to FALSE and the **COMMITCOUNT AUTOMATIC** command option is specified, the import utility will be able to automatically avoid running out of active log space.
- If **DB2\_FORCE\_APP\_ON\_MAX\_LOG** is set to FALSE and the **COMMITCOUNT *n*** command option is specified, the import utility will attempt to resolve the log full condition if it encounters an SQL0964C (Transaction Log Full) while inserting or updating a record. It will perform an unconditional commit and then will reattempt to insert or update the record. If this does not help resolve the issue (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. To avoid

processing the rows that were already committed when you retry the import operation, use the **RESTARTCOUNT** or **SKIPCOUNT** command parameters.

- If **DB2\_FORCE\_APP\_ON\_MAX\_LOG** is set to TRUE (which is the default), the import operation will fail if it encounters an SQL0964C while inserting or updating a record. This can occur irrespective of whether you specify **COMMITCOUNT** **AUTOMATIC** or **COMMITCOUNT** *n*.

The application is forced off the database and the current unit of work is rolled back. To avoid processing the rows that were already committed when you retry the import operation, use the **RESTARTCOUNT** or **SKIPCOUNT** command parameters.

## CREATE

**Note:** The **CREATE** parameter is deprecated and may be removed in a future release. For additional details, see “IMPORT command options **CREATE** and **REPLACE\_CREATE** are deprecated”.

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 more than 254, **CREATE** might fail because the rows are too long. See “Imported table re-creation” 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.

## IMPORT

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 created temporary 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 an earlier 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.

If the database table contains implicitly hidden columns, you must specify whether data for the hidden columns is included in the import operation. Use one of the following methods to indicate if data for hidden columns is included:

- Use *insert-column* to explicitly specify the columns into which data is to be inserted.

```
db2 import from delfile1 of del
    insert into table1 (c1, c2, c3,...)
```

- Use one of the hidden column file type modifiers: specify **implicitlyhiddeninclude** when the input file contains data for the hidden columns, or **implicitlyhiddenmissing** when the input file does not.

```
db2 import from delfile1 of del modified by implicitlyhiddeninclude
    insert into table1
```

- Use the DB2\_DMU\_DEFAULT registry variable on the client-side to set the default behavior when data movement utilities encounter tables with implicitly hidden columns.

```
db2set DB2_DMU_DEFAULT=IMPLICITLYHIDDENINCLUDE
db2 import from delfile1 of del insert into table1
```

**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 LOBSINFILE behavior.

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.

**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 in the data file to be imported. The case of these column names must match the case of the corresponding names in the system catalogs. Each table column

## IMPORT

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.

**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” on page 282.

### **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
- IXF (Integration Exchange Format, PC version) is a binary format that is used exclusively by DB2.

### **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 honor the CREATE TABLE statement's NOT LOGGED INITIALLY (NLI) clause or the ALTER TABLE statement's ACTIVE NOT LOGGED INITIALLY clause.

This option cannot be used to import data into system-period temporal tables.

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 re-create it, then invoke the import with INSERT statement.

This limitation applies to DB2 Universal Database™ Version 7 and DB2 UDB Version 8

### REPLACE\_CREATE

**Note:** The **REPLACE\_CREATE** parameter is deprecated and may be removed in a future release. For additional details, see “IMPORT command options CREATE and REPLACE\_CREATE are deprecated”.

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 *Imported table re-creation* 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  $n$  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

## IMPORT

*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 required. 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 behavior 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 in the following section.

## 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.

Data records in DATAFILE1 (DEL format):

## IMPORT

```
"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. However, row 3 is assigned the user-supplied identity value of 100 and row 4 is assigned the the user-supplied identity value of 101.

```
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:

```
XMLVALIDATE USING XDS
  DEFAULT S1.SCHEMA_A
  MAP ((S1.SCHEMA_A, S2.SCHEMA_B), (S2.SCHEMA_B, S1.SCHEMA_A))
```

## IMPORT

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 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.

Updates from the `IMPORT` command will always be committed at the end of an `IMPORT` task. The `IMPORT` command can also perform automatic commits during its execution to reduce the size of the lock list and the active log space. The `IMPORT` command will roll back if the active log becomes full during `IMPORT` processing.

- 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.
- Offline import does not perform automatic `COMMITs` if any of the following conditions are true:
  - The target is a view, not a table
  - Compound inserts are used
  - Buffered inserts are used
- By default, online import performs automatic commit 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

## IMPORT

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.

You 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 re-create 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 recreating 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 60 KB), the message file returned to the user on the client might be missing messages from the middle of the import operation. The first 30 KB of message information and the last 30 KB 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** or **DEL** 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 created temporary table, a declared temporary table, or a summary table.

Views cannot be created through the import utility.

Importing a multiple-part PC/IXF file whose individual parts are copied from a Windows system to an AIX system is supported. Only the name of the first file must be specified in the **IMPORT** command. For example, **IMPORT FROM data.ixf OF IXF INSERT INTO TABLE1**. The file **data.002**, etc should be available in the same directory as **data.ixf**.

On the Windows operating system:

- Importing logically split PC/IXF files is not supported.
- Importing bad format PC/IXF 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.

If the database table contains implicitly hidden columns, you must specify whether data for the hidden columns is included in the import operation.

## Federated considerations

When using the **IMPORT** command and the **INSERT**, **UPDATE**, or **INSERT\_UPDATE** command parameters, you must ensure that you have **CONTROL** privilege on the participating nickname. You must ensure that the nickname you want 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.

Some data sources, such as ODBC, do not support importing into nicknames.

## File type modifiers for the import utility

Table 13. Valid file type modifiers for the import utility: All file formats

Modifier	Description
compound= <i>x</i>	<p><i>x</i> is a number between 1 and 100 inclusive. Uses nonatomic compound SQL to insert the data, and <i>x</i> 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 <b>COMMITCOUNT</b>, or the number of rows in the data file if <b>COMMITCOUNT</b> is not specified. It is therefore recommended that the <b>COMMITCOUNT</b> option be specified to avoid transaction log overflow.</p> <p>This modifier is incompatible with <b>INSERT_UPDATE</b> mode, hierarchical tables, and the following modifiers: <b>usedefaults</b>, <b>identitymissing</b>, <b>identityignore</b>, <b>generatedmissing</b>, and <b>generatedignore</b>.</p>
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 <b>generatedmissing</b> 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 <b>generatedignore</b> 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 <b>GENERATED ALWAYS</b> and <b>GENERATED BY DEFAULT</b> identity columns. This means that for <b>GENERATED ALWAYS</b> columns, no rows will be rejected. This modifier cannot be used with the <b>identitymissing</b> 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 <b>GENERATED ALWAYS</b> and <b>GENERATED BY DEFAULT</b> identity columns. This modifier cannot be used with the <b>identityignore</b> modifier.
<b>implicitlyhiddeninclude</b>	If this modifier is specified, the utility assumes that the input data file contains data for the implicitly hidden columns and this data will also be imported. This modifier cannot be used with the <b>implicitlyhiddenmissing</b> modifier. See the Note: section for information about the precedence when multiple modifiers are specified.
<b>implicitlyhiddenmissing</b>	If this modifier is specified, the utility assumes that the input data file does not contain data for the implicitly hidden columns and the utility will generate values for those hidden columns. This modifier cannot be used with the <b>implicitlyhiddeninclude</b> modifier. See the Note: section for information about the precedence when multiple modifiers are specified.



Table 13. Valid file type modifiers for the import utility: All file formats (continued)

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 <code>db2exp.001.123.456/</code> is stored in the data file, the LOB is located at offset 123 in the file <code>db2exp.001</code>, and is 456 bytes long.</p> <p>The <b>LOBS FROM</b> clause specifies where the LOB files are located when the “lobsinfile” modifier is used. The <b>LOBS FROM</b> clause will implicitly activate the LOBSINFILE behavior. The <b>LOBS FROM</b> clause conveys to the <b>IMPORT</b> 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 <code>db2exp.001.7.-1/</code>.</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"> <li>• If a default value can be specified for a column, the default value is loaded</li> <li>• If the column is nullable, and a default value cannot be specified for that column, a NULL is loaded</li> <li>• If the column is not nullable, and a default value cannot be specified, an error is returned, and the utility stops processing.</li> </ul>
norowwarnings	Suppresses all warnings about rejected rows.
periodignore	This modifier informs the import utility that data for the period columns is present in the data file but should be ignored. When this modifier is specified, all period column values are generated by the utility. This modifier cannot be used with the <b>periodmissing</b> modifier.
periodmissing	If this modifier is specified, the utility assumes that the input data file contains no data for the period columns. When this modifier is specified, all period column values are generated by the utility. This modifier cannot be used with the <b>periodignore</b> modifier.
rowchangetimestampignore	This modifier informs the import utility that data for the row change timestamp column is present in the data file but should be ignored. This results in all ROW CHANGE TIMESTAMP being generated by the utility. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT columns. This means that for GENERATED ALWAYS columns, no rows will be rejected. This modifier cannot be used with the <b>rowchangetimestampmissing</b> modifier.
rowchangetimestampmissing	If this modifier is specified, the utility assumes that the input data file contains no data for the row change timestamp 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 columns. This modifier cannot be used with the <b>rowchangetimestampignore</b> modifier.

## IMPORT

Table 13. 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. <b>IMPORT</b> 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. <b>IMPORT</b> 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. <b>Note:</b> 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>
transactionidignore	<p>This modifier informs the import utility that data for the TRANSACTION START ID column is present in the data file but should be ignored. When this modifier is specified, the value for the TRANSACTION START ID column is generated by the utility. This modifier cannot be used with the <b>transactionidmissing</b> modifier.</p>
transactionidmissing	<p>If this modifier is specified, the utility assumes that the input data file contains no data for the TRANSACTION START ID columns. When this modifier is specified, the value for the TRANSACTION START ID column is generated by the utility. This modifier cannot be used with the <b>transactionidignore</b> modifier.</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"> <li>• 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.</li> <li>• For DEL/ASC files: A row that does not have enough columns, or is not long enough for the original specification.</li> </ul> <p><b>Note:</b> 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"> <li>• For DEL/ASC files: If the column is nullable, a NULL is loaded. If the column is not nullable, the utility rejects the row.</li> </ul>

Table 14. 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 input 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"> <li>• For pure DBCS (graphic) mixed DBCS, and EUC, delimiters are restricted to the range of x00 to x3F, inclusive.</li> <li>• 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.</li> </ul> <p><b>Note:</b></p> <ol style="list-style-type: none"> <li>1. The codepage modifier cannot be used with the lobsinfile modifier.</li> <li>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.</li> </ol>
dateformat=" <i>x</i> "	<p><i>x</i> is the format of the date in the source file.<sup>2</sup> 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 01 - 12;  mutually exclusive with M)  D - Day (one or two digits ranging from 1 - 31)  DD - Day (two digits ranging from 01 - 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>

## IMPORT

Table 14. Valid file type modifiers for the import utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
timeformat="x"	<p><i>x</i> is the format of the time in the source file.<sup>2</sup> Valid time elements are:</p> <ul style="list-style-type: none"><li>H - Hour (one or two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system)</li><li>HH - Hour (two digits ranging from 00 - 12 for a 12 hour system, and 00 - 24 for a 24 hour system; mutually exclusive with H)</li><li>M - Minute (one or two digits ranging from 0 - 59)</li><li>MM - Minute (two digits ranging from 00 - 59; mutually exclusive with M)</li><li>S - Second (one or two digits ranging from 0 - 59)</li><li>SS - Second (two digits ranging from 00 - 59; mutually exclusive with S)</li><li>SSSSS - Second of the day after midnight (5 digits ranging from 00000 - 86400; mutually exclusive with other time elements)</li><li>TT - Meridian indicator (AM or PM)</li></ul> <p>A default value of 0 is assigned for each element that is not specified. Some examples of time formats are:</p> <ul style="list-style-type: none"><li>"HH:MM:SS"</li><li>"HH.MM TT"</li><li>"SSSSS"</li></ul>

Table 14. 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.<sup>2</sup> Valid time stamp 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 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 01 - 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 00 - 12  for a 12 hour system, and 00 - 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 00 - 59;  mutually exclusive with M, minute)  S - Second (one or two digits ranging from 0 - 59)  SS - Second (two digits ranging from 00 - 59;  mutually exclusive with S)  SSSSS - Second of the day after midnight (5 digits  ranging from 00000 - 86400; mutually  exclusive with other time elements)  U (1 to 12 times)  - Fractional seconds(number of occurrences of U represent the  number of digits with each digit ranging from 0 to 9  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>
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><b>Restrictions</b></p> <p>The usegraphiccodepage 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 usegraphiccodepage modifier is also ignored by the double-byte character large objects (DBCLOBs) in files.</p>

## IMPORT

Table 14. Valid file type modifiers for the import utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
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 code page is the value specified by the codepage file type modifier, or the application code page 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><b>Note:</b> If the xmlgraphic modifier is specified with the <b>IMPORT</b> 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 15. 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>
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.<sup>3</sup></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>

Table 15. Valid file type modifiers for the import utility: ASC (non-delimited ASCII) file format (continued)

Modifier	Description
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, striptblanks 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 striptnulls. These are mutually exclusive options. This option replaces the obsolete t option, which is supported for earlier 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 striptblanks. These are mutually exclusive options. This option replaces the obsolete padwithzero option, which is supported for earlier compatibility only.</p>

Table 16. Valid file type modifiers for the import utility: DEL (delimited ASCII) file format

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.<sup>34</sup> 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, chardel'' 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>
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.<sup>34</sup></p> <p>In the following example, coldel; 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>
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.<sup>34</sup></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>

## IMPORT

Table 16. Valid file type modifiers for the import utility: DEL (delimited ASCII) file format (continued)

Modifier	Description
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&lt;row delimiter&gt; "Vincent,&lt;row delimiter&gt;, is a manager", ... ... 4005,44.37&lt;row delimiter&gt;</pre> <p>With the delprioritychar modifier specified, there will be only two rows in this data file. The second &lt;row delimiter&gt; will be interpreted as part of the first data column of the second row, while the first and the third &lt;row delimiter&gt; 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 &lt;row delimiter&gt;.</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 charde1x, delprioritychar or nodoublede1. These are mutually exclusive options.</p>
nodoublede1	<p>Suppresses recognition of double character delimiters.</p>

Table 17. 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>
indexschema= <i>schema</i>	<p>Uses the specified <i>schema</i> for the index name during index creation. If <i>schema</i> is not specified (but the keyword <i>indexschema</i> is specified), uses the connection user ID. If the keyword is not specified, uses the schema in the IXF file.</p>
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>



Table 17. Valid file type modifiers for the import utility: IXF file format (continued)

Modifier	Description
forcecreate	Specifies that the table should be created with possible missing or limited information after returning SQL3311N during an import operation.

Table 18. IMPORT behavior when using codepage and usegraphiccodepage

codepage=N	usegraphiccodepage	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. <b>Warning:</b> 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.  <b>Warning:</b> 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.  <b>Warning:</b> Graphic data will be corrupted when imported into the database if N is a single-byte code page.

**Note:**

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.

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)

## IMPORT

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. Character values provided for the `chardel`, `codel`, or `decp` file type modifiers 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 statements:

```
... modified by codel# ...
... modified by codel0x23 ...
... modified by codelX23 ...
```

4. *Delimiter considerations 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:
  - `indexif`
  - `indexschema`
  - `dldelfiletype`
  - `nodefaults`
  - `usedefaults`
  - `no_type_idfiletype`
  - `generatedignore`
  - `generatedmissing`
  - `identityignore`
  - `identitymissing`
  - `lobsinfile`
6. The **CREATE** mode is not supported for XML columns.
7. 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.
8. 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.
9. Rows containing documents that are not well-formed will be rejected.
10. 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:
  - `DBADM` authority
  - `USAGE` privilege on the XML schema to be used in the validation

11. When multiple modifiers suffixed with **ignore**, **include**, **missing**, and **override** are specified, they are applied in the order that they are listed. In the following statement, data for implicitly hidden columns that are not identity columns is included in the input data. While data for all identity columns, regardless of their implicitly hidden status, is not.

```
db2 import from delfile1 of del modified by
    implicitlyhiddeninclude identitymissing insert into table1
```

However, changing the order of the file type modifiers in the following statement means that data for all implicitly hidden columns (including hidden identity columns) is included in the input data. While data for identity columns that are not implicitly hidden is not.

```
db2 import from delfile1 of del modified by
    identitymissing implicitlyhiddeninclude insert into table1
```

If the DB2\_DMU\_DEFAULT registry variable is set to **IMPLICITLYHIDDENINCLUDE**, then:

```
db2set DB2_DMU_DEFAULT=IMPLICITLYHIDDENINCLUDE
db2 import from delfile1 of del modified by identitymissing insert into table1
```

is equivalent to:

```
db2 import from delfile1 of del modified by
    implicitlyhiddeninclude identitymissing insert into table1
```

**Related information:**

---

## INGEST

Ingest data from an input file or pipe into a DB2 table.

### Authorization

The privileges held by the authorization ID used to connect to the database must include:

- At least one of the following authorities:
  - DATAACCESS authority
  - CONTROL privilege on the target table
  - SELECT and INSERT privilege on the target table if the **INGEST** command specifies the INSERT statement (including as part of a MERGE statement)
  - SELECT and UPDATE privilege on the target table if the **INGEST** command specifies the UPDATE statement (including as part of a MERGE statement)
  - SELECT and DELETE privilege on the target table if the **INGEST** command specifies the DELETE statement (including as part of a MERGE statement)
  - INSERT, SELECT, and DELETE privilege on the target table if the **INGEST** command specifies the REPLACE clause
- SELECT privilege on the following catalog views:
  - SYSCAT.COLUMNS
  - SYSCAT.DATATYPES
  - SYSCAT.INDEXES
  - SYSCAT.INDEXCOLUSE
  - SYSCAT.SECURITYPOLICIES (if the target table has a security label column)
  - SYSCAT.TABDEP
  - SYSCAT.TABLES
  - SYSCAT.VIEWS

**Note:** Users have these privileges by default unless the database was created with the RESTRICTIVE clause.

- EXECUTE privilege on the following procedures:
  - SYSPROC.DB\_PARTITIONS (only when server is not a DB2 pureScale server)
  - SYSPROC.DB2\_GET\_INSTANCE\_INFO (only when server is a DB2 pureScale server)
  - SYSPROC.MON\_GET\_CONNECTION (V9.7 and later) or SYSIBMADM.APPLICATIONS (V9.5)
- If the target table has any triggers, the authorization ID must have sufficient privileges to execute the operations that the triggers specify.
- To insert into or update a table that has protected columns, the authorization ID must have LBAC credentials that allow write access to all protected columns in the table. Otherwise, the command fails and an error is returned.
- If an UPDATE or MERGE statement requires reading a protected column, the authorization ID must have LBAC credentials that allow read access to the column. Otherwise, the command fails and an error is returned.
- To insert into or update a table that has protected rows, the authorization ID must hold an LBAC credential that meets these criteria:
  - The LBAC credential is part of the security policy protecting the table.

- If the security policy was defined as RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL, then the LBAC credential must be granted to the authorization ID for write access

The security label on the row to be inserted, the authorization ID's LBAC credentials, the security policy definition, and the LBAC rules determine whether insert or update can be performed on the table with protected rows.

- If the **INGEST** command specifies the RESTART NEW (the default) or RESTART CONTINUE option, then SELECT, INSERT, UPDATE, and DELETE privilege on the restart table
- If the **INGEST** command specifies the RESTART TERMINATE option, then SELECT and DELETE privilege on the restart table.
- If the **INGEST** command specifies the EXCEPTION TABLE option, then INSERT privilege on the exception table

In addition, the SQL statement on the **INGEST** command is subject to the same fine grained access controls (FGAC) that it would be if the user running the ingest utility accessed the table outside of the ingest utility.

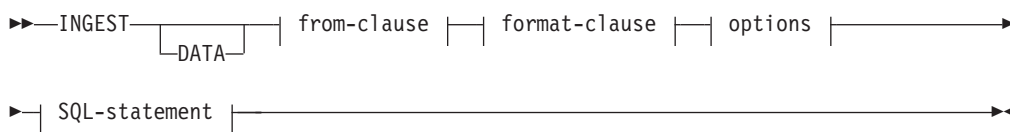
The user running the CLP must have the following file permissions:

- read access to the control file
- if the **INGEST** command specifies the MESSAGES option, then:
  - write access to the directory containing the messages file
  - write access to the file if the messages file already exists
- If the **INGEST** command specifies the DUMPFIL option, then:
  - write access to the directory containing the dump file
  - write access to the dump file if the dump file already exists

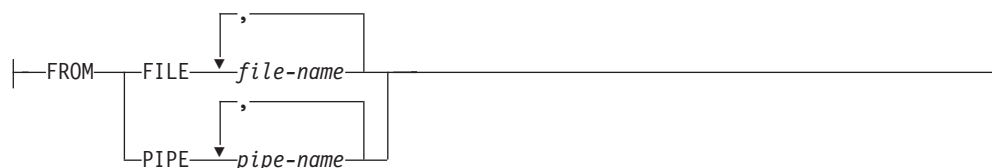
### Required connection

Database

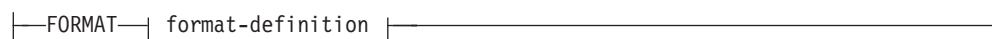
### Command syntax



#### from-clause:

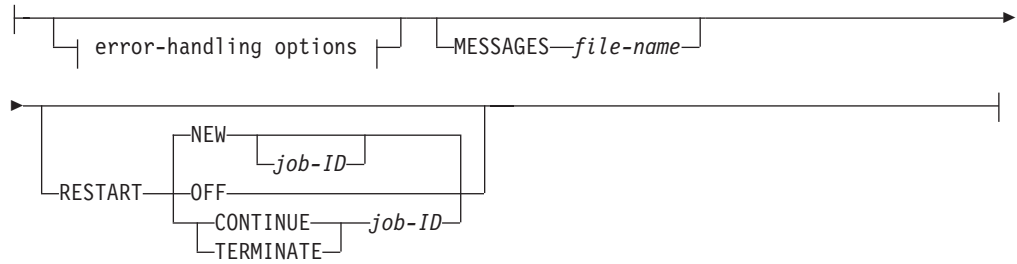


#### format-clause:

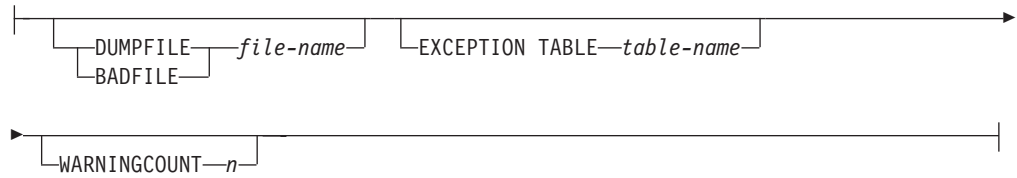


# INGEST

## options:



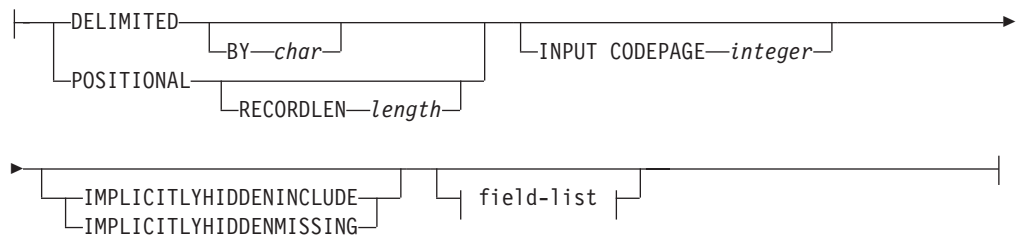
## error-handling options:



## SQL-statement:



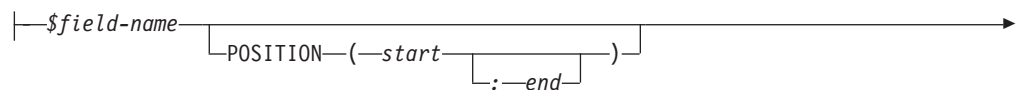
## format-definition:

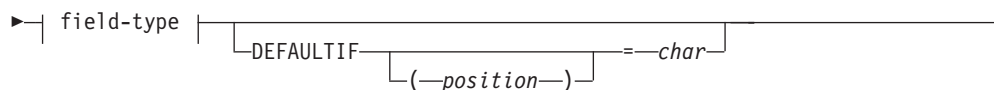


## field-list:

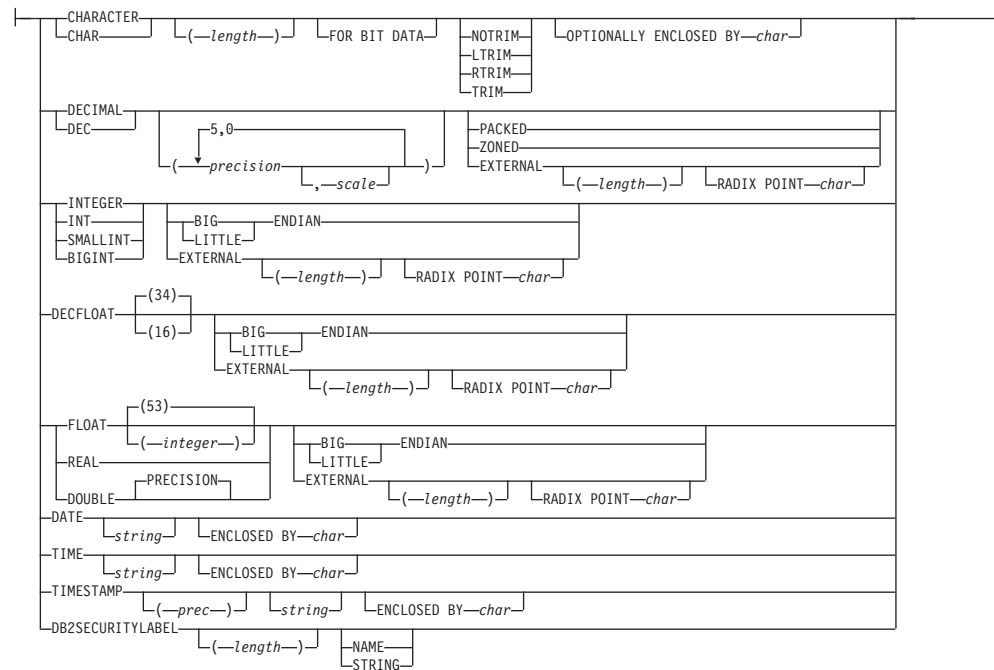


## field-definition:

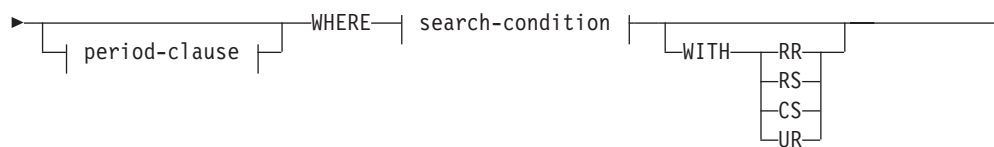




**field-type:**



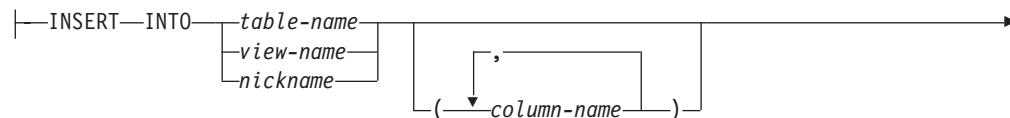
**DELETE-statement:**



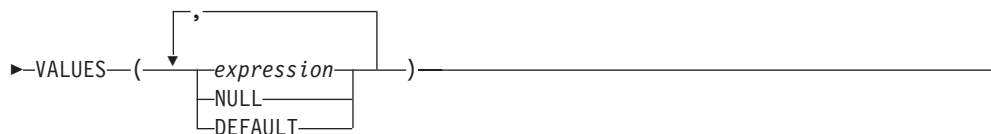
**period-clause:**



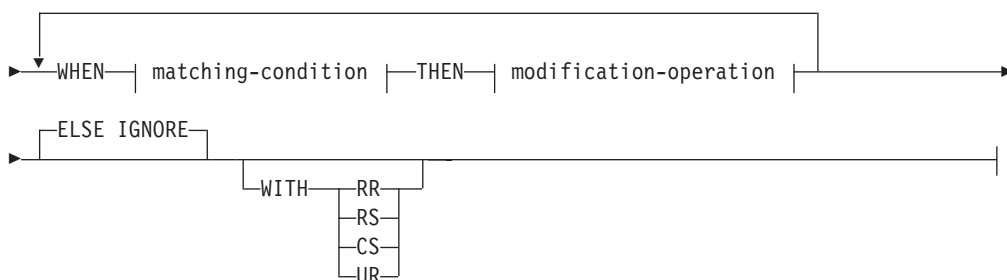
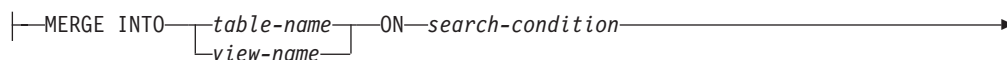
**INSERT-statement:**



# INGEST



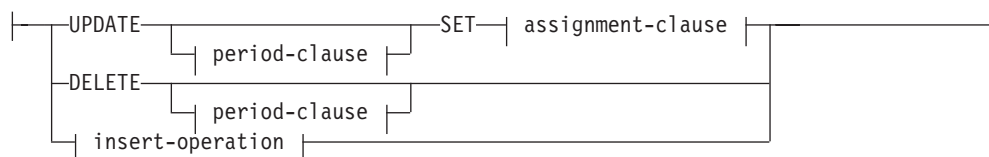
## MERGE-statement:



## matching-condition:



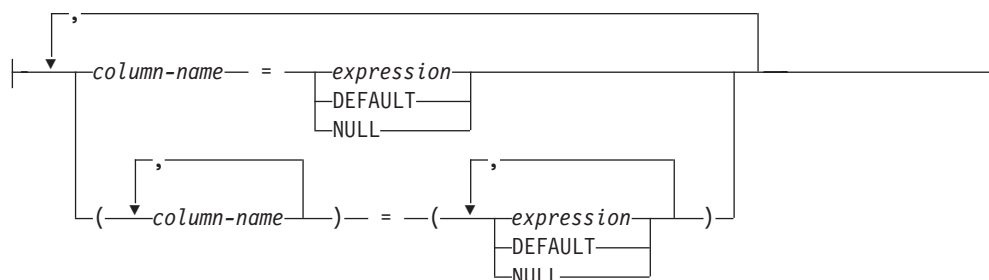
## modification-operation:



## period-clause:

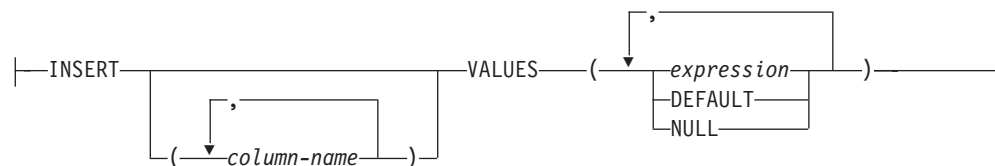


## assignment-clause:

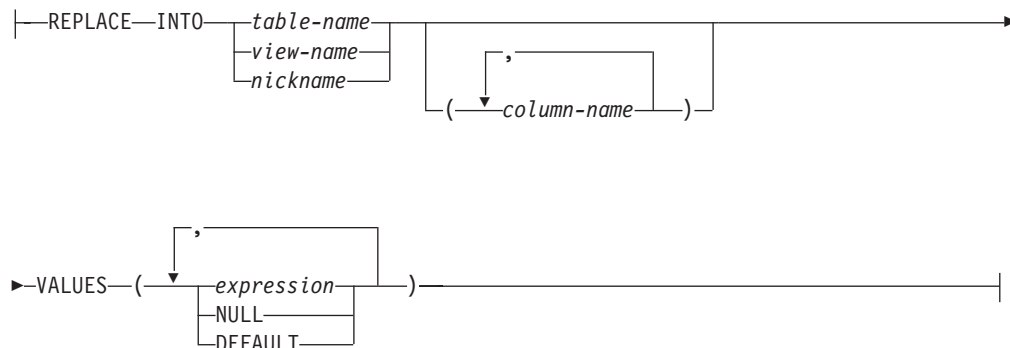




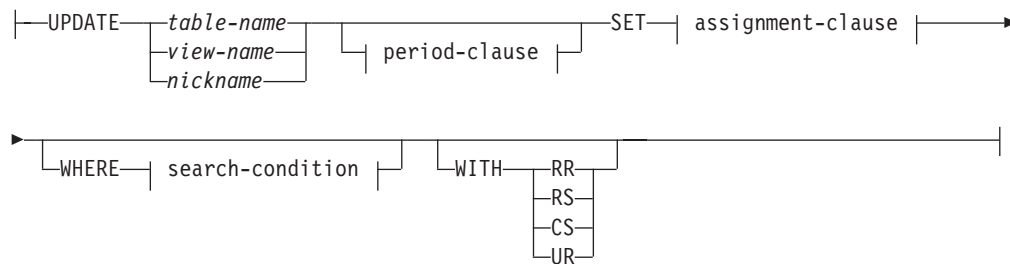
**insert-operation:**



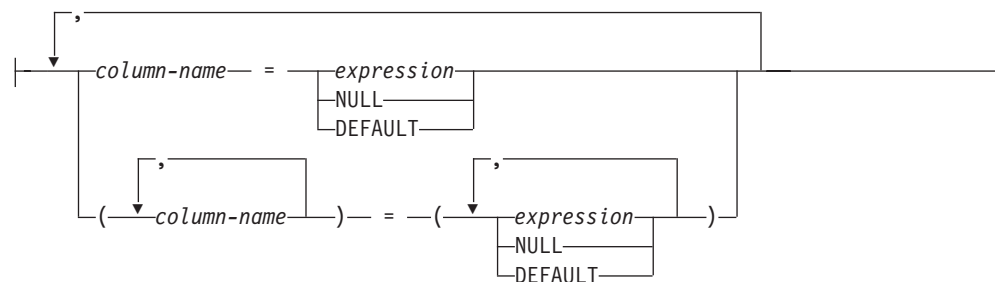
**REPLACE-statement:**



**UPDATE-statement:**

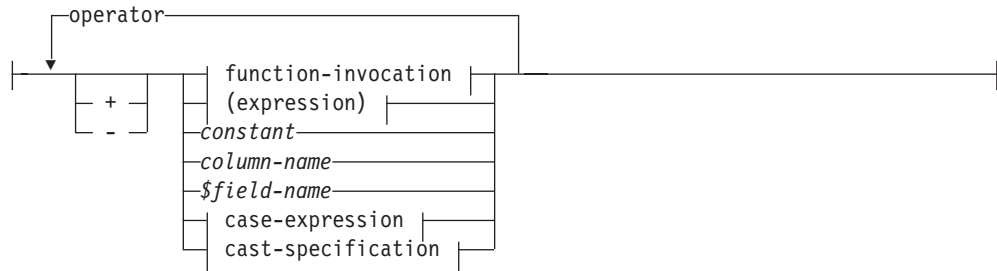


**assignment-clause:**



**expression:**

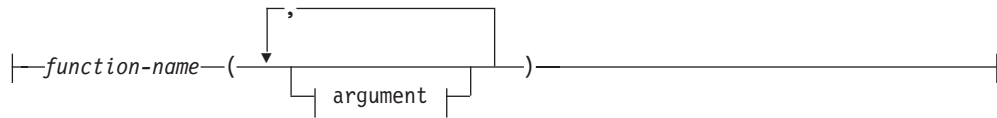
# INGEST



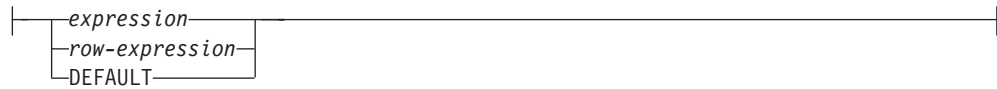
## operator:



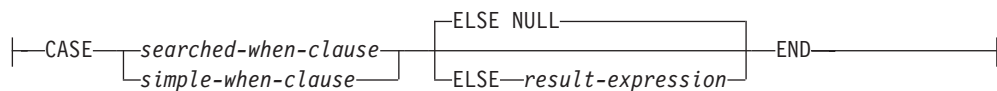
## function-invocation:



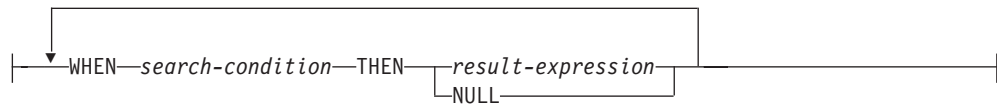
## argument:



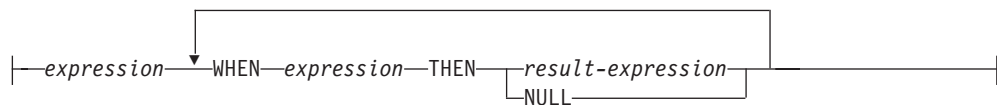
## case-expression:



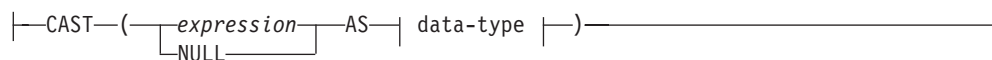
## searched-when-clause:



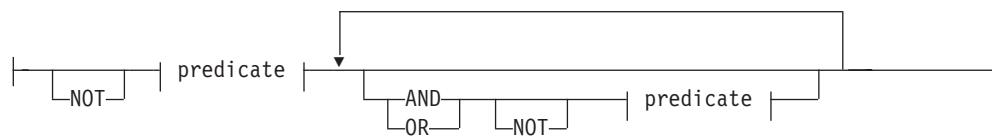
## simple-when-clause:



**cast-specification:**



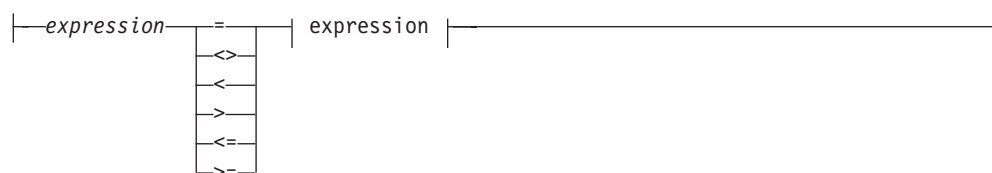
**search-condition:**



**predicate:**



**basic-predicate:**



**NULL-predicate:**



**Command parameters**

**FROM**

Specifies the source of the input data. The ingest utility always reads the input data from the client.

**FILE *file-name***

Specifies that data is to be read from the specified files. For the syntax of file names, see "File and pipe names" on page 323.

**PIPE *pipe-name***

Specifies that data is to be read from the specified pipes. For the syntax of pipe names, see "File and pipe names" on page 323.

**FORMAT *format-definition***

The data is in the format given by the specified format definition. The syntax of a format definition is given in the "format-definition" section described previously.

**DUMPFILe or BADFILE *file-name***

Specifies that rows rejected by the formatters are to be written to the specified file. The formatters reject rows due to the following types of errors:

- numbers that are invalid or out of range (based on the field type)
- dates, times, and timestamps that do not fit the specified format
- Any other errors detected by the formatter

If a relative path is specified, it is relative to the current directory and the default drive. On Linux and UNIX platforms, if the file does not exist, the ingest utility creates the file with the permission flags specified by the user's umask setting. The ingest utility uses the umask setting that was in effect at the time the DB2 CLP backend process (db2bp) started, which is usually at the first connection. If the file already exists, the ingest utility appends the records to the end of the file.

The ingest utility writes each record in the same format as it appeared in the input. This means that after correcting the errors, you can re-run the **INGEST** command using the dump file as input. However, the ingest utility might not write records in the same order that they appeared across multiple input files or within each input file. If you want the dump file to contain records in the same order that they appeared within each input file, set the *num\_formatters* configuration parameter to 1. In this case:

- If the operation is INSERT or REPLACE, records from different input files might still appear out of order but all the records from a given input file appears in the same order that they were within that file.
- If the operation is UPDATE, DELETE, or MERGE, all records are in the same order that they were across all input files and within each input file

If an error occurs opening or writing to the file, the ingest utility issues an error message, including the input source and line number, and ends the command. In this case, the record is not saved anywhere. (If the input is from a file, the user can get the original input record from the file.)

The dump file is guaranteed to be complete only if the **INGEST** command completes normally in a single run. If the **INGEST** command fails and the restarted command succeeds, the combination of dump files from the two commands might be missing records or might contain duplicate records.

If this parameter is not specified, the ingest utility issues an error message for each rejected row, but does not save the row anywhere.

**Note:** If you specify this parameter and any rejected input records contain sensitive data, these records appear in the dump file. You must protect the file from unauthorized access as needed.

### EXCEPTION TABLE *table-name*

Specifies that rows inserted by the ingest utility and rejected by DB2 with certain SQLSTATES are to be written to the specified table. DB2 could reject rows due to the following types of errors. Note that each of these errors indicates bad data in the input file:

- For character data, right truncation occurred; for example, an update or insert value is a string that is too long for the column, or a datetime value cannot be assigned to a host variable, because it is too small.
- A null value, or the absence of an indicator parameter was detected; for example, the null value cannot be assigned to a host variable, because no indicator variable is specified.
- A numeric value is out of range.
- An invalid datetime format was detected; that is, an invalid string representation or value was specified.

- The character value for a CAST specification or cast scalar function is invalid.
- A character is not in the coded character set.
- The data partitioning key value is not valid.
- A resulting row did not satisfy row permissions.
- An insert or update value is null, but the column cannot contain null values.
- The insert or update value of the FOREIGN KEY is not equal to any value of the parent key of the parent table.
- A violation of the constraint imposed by a unique index or a unique constraint occurred.
- The resulting row of the INSERT or UPDATE does not conform to the check constraint definition.
- The value cannot be converted to a valid security label for the security policy protecting the table.
- This authorization ID is not allowed to perform the operation on the protected table.
- The component element is not defined in security label component.
- The specified security label name cannot be found for the specified security policy.
- The data type, length, or value of the argument to routine is incorrect.

The specified table must be defined as described in the “Exception tables” topic in the related links section. The restrictions listed in that section and in message SQL3604N also apply, with the following exception:

- If the target table contains a DB2SECURITYLABEL column, the exception table column corresponding to the DB2SECURITYLABEL column must be of type VARCHAR with a length of at least 128. (Type LONG VARCHAR is also permissible but has been deprecated.)

The ingest utility uses the exception table only when the operation is INSERT or REPLACE. If the operation is UPDATE, MERGE, or DELETE, the ingest utility ignores this clause and does not return an error or warning.

If an error occurs inserting into the table, the ingest utility issues an error message, including the input source and line number, and continues. In this case, the record is not saved anywhere. (If the input is from a file, the user can get the original input record from the file.)

If the table name is not fully qualified, the ingest utility uses the value of CURRENT SCHEMA for the schema name. If the EXCEPTION TABLE parameter is not specified, the ingest utility issues an error message for each rejected row but does not save the row anywhere.

#### **WARNINGCOUNT** *n*

Specifies that the **INGEST** command is to stop after *n* warning and error messages have been issued. This parameter covers only certain types of errors. For more information, see Categories of errors. The number is the total for all input sources. If this limit is reached in the middle of a transaction, the transaction is rolled back. However, rows that were committed to the target table before this limit was reached remain in the table. If the number is 0, the **INGEST** command continues regardless of the number of warnings and errors. The range is 0 - 2,147,483,647 (max 32-bit signed integer). The default is 0.

**MESSAGES** *file-name*

Specifies the file to receive informational, warning, and error messages.

If a relative path is specified, it is relative to the current directory and the default drive. On Linux and UNIX platforms, if the file does not exist, the ingest utility creates the file with the permission flags specified by the user's umask setting. The ingest utility uses the umask setting that was in effect at the time the DB2 CLP backend process (db2bp) started, which is usually at the first connection. If the file already exists, the ingest utility appends to the end of the file. If this parameter is not specified, the ingest utility writes messages to standard output.

Even when this parameter is specified, the ingest utility writes messages to standard output in the following cases:

- syntax errors
- an input file is not found or not readable
- target or exception table is not found
- the dump file or messages file cannot be opened
- other errors detected at start-up

Also, the summary messages issued at the end (showing number of rows read, inserted, and so on, and the total number of warnings and errors) are always written to standard output.

If an error occurs opening or writing to the messages file, the ingest utility issues an error message and ends the command.

For examples of messages from the **INGEST** command, see "Messages from the INGEST command" on page 322.

**RESTART NEW** *job-ID*

Specifies that if the **INGEST** command fails before completing, it can be restarted from the point of the last commit by specifying the **RESTART CONTINUE** option on a later **INGEST** command. The *job-ID* is a string of up to 128 bytes that uniquely identifies the **INGEST** command. This job-ID must be unique across all **INGEST** commands in the current database that specified the **RESTART** option and are not yet complete. (These could be commands that are still running or that failed before completing.) Once the **INGEST** command completes, you can reuse the job-ID on the **RESTART** parameter of a later **INGEST** command. If the job-ID is omitted, the ingest utility generates one. This option is the default.

Before using this option, you must have created the restart log table. For more information, see "Restarting a failed INGEST command" in the following related links section.

**RESTART OFF**

Specifies that no restart information is to be saved. If the **INGEST** command fails before completing, it cannot be restarted using the **RESTART CONTINUE** option. If you want to rerun the command to completion, you need to restore the target table to its state before running the **INGEST** command, and rerun the **INGEST** command with the same input data.

**RESTART CONTINUE** *job-ID*

Specifies that the ingest utility is to restart a previous **INGEST** command that specified the **RESTART NEW** option and failed before completing. The *job-ID* specified on this option must match the job-ID specified on the previous **INGEST** command. This restarted command is also restartable. Once the restarted command completes, you can reuse the job-ID on the

RESTART NEW parameter of a later **INGEST** command. For more information, see “Restarting a failed INGEST command” in the following related links section.

#### **RESTART TERMINATE** *job-ID*

Specifies that the ingest utility is to clean up the restart information for a previous **INGEST** command that specified the RESTART NEW option and failed before completing. The string specified on this option must match the string specified on the previous **INGEST** command.

This option is intended for when a previous restartable **INGEST** command failed, and you plan to never resume the job. Once you specify this option, the **INGEST** command that failed earlier can no longer be restarted. You can, however, reuse the string on the RESTART NEW parameter of a later **INGEST** command.

**Note:** Unlike with the TERMINATE parameter of the **LOAD** command, data that the original **INGEST** command committed before its failure is still in the target table. For more information, see “Terminating a failed INGEST command” in the following related links section.

#### **DELIMITED BY** *char*

Specifies that each input record is a sequence of text fields delimited by the specified character. The delimiter can be any single-byte character except the following values:

- the null character (X'00')
- blank (X'20')
- carriage return or line feed (X'0D' or X'0A')
- dot (X'2E'), regardless of what is specified on the RADIX POINT clause
- the decimal point (specified by the RADIX POINT clause; the default is the dot .)
- the string delimiter (specified by the ENCLOSED BY clause; the default is the double quote ")

If the input data code page is a DBCS or EUC code page, the character must be in the range X'01' to X'3F', inclusive. Note that this excludes the vertical bar (| or X'7C'). If the input data code page is not a single-byte code page, the character must be in the range X'01' to X'7F', inclusive. If the input code page is a single-byte code page, the character can be in the range X'01' to X'FF'.

The default is a comma ,.

#### **POSITIONAL**

Specifies that each input record is a sequence of text and/or binary fields identified by their position in the input record. The position of each field is given by the POSITION clause in the field definition.

#### **RECORDLEN** *length*

Specifies that the input is fixed-length records and each record has the specified length in bytes. Any carriage return or line feed characters (X'0D' or X'0A') in the record are assumed to be part of the data. Note that the record length that some editors display does not include these characters, so you might need to add 1 or 2 to the record length that the editor displays. If the input contains binary data, this parameter is required.

The range is 1 - 32,767. If RECORDLEN specifies a length less than the sum of the field lengths specified on each field definition, the ingest utility issues

an error message and exits. If RECORDLEN specifies a length greater than the maximum field end position, the ingest utility ignores the bytes past the end of the field with the maximum end position. If the last record is too short, the ingest utility issues an error message and rejects the record.

If this parameter is omitted, the ingest utility recognizes both of the following as the record (row) delimiter, regardless of platform:

- line feed (X'0A')
- one or more carriage returns followed by a line feed (X'0D0A')

#### **INPUT CODEPAGE** *integer*

Specifies the code page of the input data. The ingest utility supports the code pages listed in the “Supported territory codes and code pages” topic in the *Related links* section, except the following values :

- 1200 (UTF-16)
- any code pages listed as "host" code pages

The default is the DB2 application code page.

If the input data is not in the code page specified by this parameter, the ingest utility does not issue an error. However, if the ingest utility or DB2 needs to convert the input data into another code page, the utility might convert some characters to the substitute character (Unicode U+001A) or other characters that you did not intend. For more information, see the topic on “Code page considerations for the ingest utility”.

#### **IMPLICITLYHIDDENINCLUDE**

Specifies that implicitly hidden columns are included in the default column list. If the DB2\_DMU\_DEFAULT registry variable on the client-side is also specified, then this keyword takes precedence. This keyword is only allowed for INSERT statements where there is not a column list following the table name. If the SQL statement is not an INSERT or there is a column list after the table name, then an error is returned (SQLCODE -2918).

#### **IMPLICITLYHIDDENMISSING**

Specifies that implicitly hidden columns are omitted from the default column list. If the DB2\_DMU\_DEFAULT registry variable on the client-side is also specified, then this keyword takes precedence. This keyword is only allowed for INSERT statements where there is not a column list following the table name. If the SQL statement is not an INSERT or there is a column list after the table name, then an error is returned (SQLCODE -2918).

#### **field-list**

Specifies the list of field definitions. If the list is omitted, it defaults as described in Default field definition list.

#### *\$field-name*

Field names can be from 2 to 129 bytes in length (including the dollar sign) and follow the same rules as SQL identifiers. You can also specify delimited field names by specifying a dollar sign followed by a delimited name, for example, "\$My Field Name".

If the SQL statement specifies a table name column name that starts with a dollar sign, specify the table name or column name as a delimited identifier. For example, the ingest utility interprets the following identifiers as indicated:



Table 19. The meaning of field name identifiers

Identifier	Meaning
\$name	field "\$NAME"
\$NAME	field "\$NAME"
\$"My Name"	field "\$My Name"
"\$name"	column "\$name" (in lowercase)
"\$NAME"	column "\$NAME" (in upper case)
"\$My Name"	column "\$My Name"

If you specify the **INGEST** command on the system command line in UNIX, in order to avoid the field names being seen as environment variables, you need to put the command in single quotation marks. For more information, see "Using INGEST on the system command line (UNIX)" on page 320.

#### **POSITION** (*start:end*)

This clause is valid only when the format is POSITIONAL. It specifies the starting position and optionally the ending position of the field within the record. The units are bytes. The first byte in the record is at position 1. If the ending position is specified, it must be greater than or equal to the starting position.

Positions specified on different fields can overlap. For example, if the full name of an American state is in positions 1 - 10, you could define the following two fields:

- \$state\_full\_name POSITION(1:10)
- \$state\_abbreviation POSITION(1:2)

If there is data outside the specified field positions, the ingest utility ignores it. For example, if the field definition list specifies two fields at POSITION(1:10) and POSITION(21:30), the ingest utility ignores data in positions 11:20 and any data past position 30 to the end of the record.

If the POSITION clause is omitted, the starting position is assumed to be the next byte following the previous field, or position 1 if this is the first field. If the ending position is omitted, the utility uses a default or issues an error, as described in "Rules and defaults for field lengths" on page 326.

#### **field-type**

Specifies the data type of the field.

#### **CHARACTER** (*length*)

Character data. The code page is specified by the INPUT CODEPAGE parameter. The default is the application code page.

For information about the default length, see "Rules and defaults for field lengths" on page 326.

#### **NOTRIM, LTRIM, RTRIM, TRIM**

Specifies that leading blanks (LTRIM), trailing blanks (RTRIM), or both leading and trailing blanks (TRIM) in the field and not enclosed by the string delimiter are not part of the string. When the format is DELIMITED, the default is TRIM. When the format is POSITIONAL, the default is NOTRIM. If this parameter is specified on any character field, it must have the same value on all other character fields.

#### **OPTIONALLY ENCLOSED BY** *char*

This clause is valid only when the format is DELIMITED. The

string can be enclosed by the specified character. The restrictions are the same as for the field delimiter. In addition, the character cannot be the same as the field delimiter or the decimal point. If this parameter is specified on any character, date, time, or timestamp field, it must have the same value on all other character, date, time, and timestamp fields.

The utility uses the following algorithm to determine whether the value starts with the delimiter:

- If the TRIM or LTRIM option is specified (or defaults), the utility tests the first character in the field after trimming of leading blanks.
- If the RTRIM or NOTRIM option is specified, the utility tests the first character in the field. (In other words, when RTRIM or NOTRIM is specified, the utility does not skip over leading blanks to locate the string delimiter.)

Except for record delimiters, the utility considers all characters between the enclosing characters to be part of the string. If the string contains the enclosing character, it must be doubled inside the string. For example, if the string is don't know and the enclosing character is a single quote, specify 'don''t know'. If the string does not end with the enclosing character, the utility considers the opening delimiter to be part of the string.

The utility issues warning SQL3114W under the following conditions:

- TRIM or RTRIM is specified (or defaults) and there are extra non-blank characters between the closing delimiter and the next field delimiter (or end of record).
- NOTRIM or LTRIM is specified and there are extra characters (including blanks) between the closing delimiter and the next field delimiter (or end of record).

When this parameter is not specified, the default is `OPTIONALLY ENCLOSED BY ''`. In that case, if the value does not start with a double quote, the utility considers any character (other than the field delimiter) around the string to be part of the string.

#### FOR BIT DATA

Specifies that the field contains bit data and that the ingest utility is not to convert it from the input data code page to the ingest utility application code page. If this field is used to assign to a column that specifies FOR BIT DATA, it is strongly recommended that the field definition also specify FOR BIT DATA. Otherwise, the value assigned to the column is unpredictable, because of how the ingest utility sometimes optimizes code page conversion.

#### INTEGER/SMALLINT/BIGINT EXTERNAL (*length*)

A character string of up to *length* bytes representing a number. The string can be an integer, decimal, or floating point constant. If the string specifies a decimal or floating point value, the ingest utility converts it to the specified integer type by truncating the fractional part. If the string is not a valid number, or the number is out of range, the ingest utility issues an error message and rejects the record. For more information, see "Handling of invalid numeric data" on page 332.

If EXTERNAL is specified on any integer field, it must be specified on all other integer, decfloat, and float fields.

For information about default length, see “Rules and defaults for field lengths” on page 326.

**DECIMAL** (*precision, scale*) **EXTERNAL** (*length*)

A character string of up to *length* bytes representing a decimal number. The string can be an integer, decimal, or floating point constant. The ingest utility converts it to a decimal value with the specified precision and scale. If the string specifies a decimal value with a scale larger than the scale specified on the field type, the ingest utility truncates the fractional part of the value with no warning or error. If the string specifies a decimal value with a scale smaller than the scale specified on the field type, the ingest utility pads the fractional part of the value with zeros. If the string is not a valid number, or the number is out of range (other than the scale), the ingest utility issues an error message and rejects the record. For more information, see “Handling of invalid numeric data” on page 332.

The precision can range from 1 to 31. The scale can range from 0 to the precision. The default is (5,0). For more information about default length, see “Rules and defaults for field lengths” on page 326.

If EXTERNAL is specified on any decimal field, it must be specified on all other decimal fields.

**DECFLOAT** (16|34) **EXTERNAL** (*length*)

A character string of up to *length* bytes representing a decimal floating point number. The string can be an integer, decimal, or floating point constant, or a decimal floating point special value. The ingest utility converts it to a decimal floating point value of the specified precision. If the string is not a valid number or decimal floating point special value, or the number is out of range, the ingest utility issues an error message and rejects the record. For more information, see “Handling of invalid numeric data” on page 332. The precision can be either 16 or 34 (the default).

For more information about the default length, see “Rules and defaults for field lengths” on page 326.

If EXTERNAL is specified on any decfloat field, it must be specified on all other integer, decfloat, and float fields.

**FLOAT**(*integer*) **EXTERNAL**(*length*)

A character string of up to *length* bytes representing a floating point number. The string can be an integer, decimal, or floating point constant. If the string is not a valid number, or the number is out of range, the ingest utility issues an error message and rejects the record. For more information, see “Handling of invalid numeric data” on page 332.

The value of the integer must be in the range 1 through 53. Values 1 through 24 indicate single precision, and values 25 through 53 indicate double precision. The default is 53. You can also specify:

**REAL** For single-precision floating-point.

**DOUBLE, DOUBLE PRECISION, or FLOAT**

For double-precision floating-point.

If EXTERNAL is specified on any float field, it must be specified on all other integer, decfloat, and float fields.

For information about the default length, see “Rules and defaults for field lengths” on page 326.

**RADIX POINT** *char*

The character to use as the decimal point. The restrictions are the same as those for the field delimiter. In addition, the character cannot be the same as the field delimiter or the string delimiter. If this parameter is specified on any integer, decimal, decfloat, or float field, it must have the same value on all other integer, decimal, decfloat, and float fields. The default is a dot ..

**INTEGER, SMALLINT, or BIGINT**

An integer of the specified type in binary. These types can be specified only when the format is POSITIONAL and the RECORDLEN parameter specified. The following can also be specified:

**BIG | LITTLE ENDIAN**

Specifies whether the integer is in big endian format (most significant byte at low address) or little endian format (least significant byte at low address). This option applies to integer, float, and decfloat types, but not decimal types. The default is the default format on the current hardware platform where the ingest utility is running.

**DECIMAL** (*precision, scale*) **PACKED | ZONED**

A decimal number of the specified precision and scale in packed or zoned format. These types can be specified only when the format is POSITIONAL and the RECORDLEN parameter is specified. The precision can range from 1 to 31. The scale can range from 0 to the precision. The default is (5,0). The PACKED/ZONED setting must be the same on all other decimal fields. The default is PACKED.

**DECFLOAT(16|34)**

A decimal floating point number of the specified precision in binary. The precision can be either 16 or 34 (the default). This type can be specified only when the format is POSITIONAL and the RECORDLEN parameter is specified.

**FLOAT(integer)**

A single- or double-precision floating point number of the specified size in binary (IEEE floating point format). The value of the integer must be in the range 1 through 53. Values 1 through 24 indicate single precision and values 25 through 53 indicate double precision. The default is 53. These types can be specified only when the format is POSITIONAL and the RECORDLEN parameter is specified. You can also specify:

**REAL** For single-precision floating-point.

**DOUBLE, DOUBLE PRECISION, or FLOAT**

For double-precision floating-point.

**DATE, TIME, TIMESTAMP**

A date, time, or timestamp specified as a string. If the database has the **date\_compat** database configuration parameter set to ON, DATE is equivalent to TIMESTAMP(0).

For information about the default length, see “Rules and defaults for field lengths” on page 326.

*precision*

Specifies the precision of a timestamp. The range is 0 - 12. The

default is 6. If the input data specifies a timestamp with a precision larger than the precision specified on the field type, the ingest utility truncates the fractional part of the timestamp with no warning or error (provided that the timestamp has the format specified on the format string). If the input data specifies a timestamp with a precision smaller than the precision specified on the field type, the ingest utility pads the fractional part of the timestamp with zeros.

*string* Specifies the format of the date, time, or timestamp value. The elements of the format string are those currently supported by the **IMPORT** and **LOAD** commands. For more information, see the descriptions of the `dateformat`, `timeformat`, and `timestampformat` file type modifiers in the “IMPORT command” topic.

If the format is `POSITIONAL` and the string is longer than the length specified by the `POSITION` parameter, the ingest utility issues an error. For example, if the string is 'yyyy-mm-dd' (10 bytes), but the position is 1:6, the ingest utility issues an error.

The date, time, or timestamp value in the input data must match the format, except that you can omit sub-fields on the right. For example, if the timestamp format is "yyyy/mm/dd hh:mm:ss.uuu", the following timestamps are valid:

```
2010/06/03 12:34
2010/06/03 12:34:56
2010/06/03 12:34:56.1
2010/06/03 12:34:56.12
2010/06/03 12:34:56.123
```

However, "2010/06/03 12:34:56.1234" is not valid. When you omit sub-fields, they default as described in the descriptions of the `dateformat`, `timeformat`, and `timestampformat` file type modifiers in the “IMPORT command” topic.

There is one exception to the previous description of omitting sub-fields: When the format is `POSITIONAL` and the field is the last field in the record and the record ends before the field's end position, the utility issues a warning and uses `NULL` for the value.

If the date, time, or timestamp value is not in the specified format (other than sub-fields missing from the right), the ingest utility issues an error message and rejects the row.

The following rules also apply

- If this parameter is specified on any `DATE` field, it must also be specified and have the same value on all other `DATE` fields. If the database has the `date_compat` database configuration parameter on (in which case, field type `DATE` is equivalent to `TIMESTAMP(0)`), it must also be specified and have the same value on all other `TIMESTAMP` fields.
- If it is specified on any `TIME` field, it must also be specified and have the same value on all other `TIME` fields.
- If it is specified on any `TIMESTAMP` field, it must also be specified and have the same value on all other `TIMESTAMP` fields. If the database has the `date_compat` database configuration parameter on (in which case, field type `DATE` is equivalent to `TIMESTAMP(0)`), it must also be specified and have the same value on all other `DATE` fields.

If the format string is not specified, the ingest utility recognizes various formats, as shown in Table 20.

Table 20. Default values for DATE, TIME, and TIMESTAMP

Field type	Default format
DATE	<ul style="list-style-type: none"> <li>If the <code>date_compat</code> database configuration parameter is set to OFF, then:                             <ul style="list-style-type: none"> <li>'yyyymmdd' (delimited format only)</li> <li>'yyyy-mm-dd'</li> <li>'mm/dd/yyyy'</li> <li>'dd.mm.yyyy'</li> </ul> </li> <li><b>Note:</b> In all but the first format, the month and day can also be specified as a single digit, for example, 'yyyy-m-d'.</li> <li>If the <code>date_compat</code> database configuration parameter is set to ON, then the same defaults as TIMESTAMP(0).</li> </ul>
TIME	<ul style="list-style-type: none"> <li>'hh.mm.ss'</li> <li>'hh:mm tt'</li> <li>'hh:mm:ss'</li> </ul>
TIMESTAMP	<p>'yyyy-mm-dd-hh.mm.ss.aaaaaaaaaaaa' or 'yyyy-mm-dd hh:mm:ss.aaaaaaaaaaaa'</p> <ul style="list-style-type: none"> <li>'yyyy-mm-dd-hh.mm.ss.aaaaaaaaaaaa'</li> <li>'yyyy-mm-dd hh:mm:ss.aaaaaaaaaaaa'</li> <li>'yyyy-mm-dd'</li> <li>'mm/dd/yyyy'</li> <li>'dd.mm.yyyy'</li> <li>'yyyymmdd' (delimited format only)</li> </ul> <p><b>Note:</b> In all but the first format, the month and day can also be specified as a single digit, for example, 'yyyy-m-d-hh.mm.ss.aaaaaaaaaaaa'.</p>

**OPTIONALLY ENCLOSED BY** *char*

This clause is valid only when the format is DELIMITED. The string can be enclosed by the specified character. The restrictions are the same as for the field delimiter. In addition, the character cannot be the same as the field delimiter or the decimal point. If this parameter is specified on any character, date, time, or timestamp field, it must have the same value on all other character, date, time, and timestamp fields.

The utility uses the following algorithm to determine whether the value starts with the delimiter:

- If the TRIM or LTRIM option is specified (or defaults), the utility tests the first character in the field after trimming of leading blanks.
- If the RTRIM or NOTRIM option is specified, the utility tests the first character in the field. (In other words, when RTRIM or

NOTRIM is specified, the utility does not skip over leading blanks to locate the string delimiter.)

Except for record delimiters, the utility considers all characters between the enclosing characters to be part of the string. If the string contains the enclosing character, it must be doubled inside the string. For example, if the string is don't know and the enclosing character is a single quote, specify 'don''t know'. If the string does not end with the enclosing character, the utility considers the opening delimiter to be part of the string.

The utility issues warning SQL3114W under the following conditions:

- TRIM or RTRIM is specified (or defaults) and there are extra non-blank characters between the closing delimiter and the next field delimiter (or end of record).
- NOTRIM or LTRIM is specified and there are extra characters (including blanks) between the closing delimiter and the next field delimiter (or end of record).

When this parameter is not specified, the default is OPTIONALLY ENCLOSED BY '"'. In that case, if the value does not start with a double quote, the utility considers any character (other than the field delimiter) around the string to be part of the string.

#### **DB2SECURITYLABEL** (*length*)

Specifies a DB2 security label. If neither NAME nor STRING is specified and the format is POSITIONAL, the default format is encoded numeric format; otherwise an error is returned.

For information about the default length, see “Rules and defaults for field lengths” on page 326.

#### **NAME**

The DB2 security label is specified by its name. If the format is DELIMITED, either NAME or STRING must be specified. If this parameter is specified on any DB2SECURITYLABEL field, it must have the same value on all other DB2SECURITYLABEL fields. If no security label exists with the indicated name for the security policy protecting the table, the row is not loaded and a warning is returned.

#### **STRING**

The DB2 security label is specified in string format. If the format is DELIMITED, either NAME or STRING must be specified. If this parameter is specified on any DB2SECURITYLABEL field, it must have the same value on all other DB2SECURITYLABEL fields. If a string is not in the proper format, the row is not loaded and a warning is returned. If the string does not represent a valid security label that is part of the security policy protecting the table, the row is not loaded and a warning is returned.

The default is encoded numeric format and is allowed only when the format is POSITIONAL.

#### **DEFAULTIF** (*position*) = *char*

The position is optional and can be specified only if the format is POSITIONAL. If the position is specified, the utility reads the character from the input record before applying any specified or default trim option. The position does not have to be in the field. If the position is omitted, the

## INGEST

utility first applies any specified or default trim option and then reads the first character of the field. In either case, if the character read from the input record matches the one specified, the utility considers the field value to be the default value of the table column to which the field is assigned.

The DEFAULTIF parameter has the following restrictions:

- When the position is omitted, there must be at least one space between DEFAULTIF and the equal sign. Otherwise a syntax error occurs.
- The DEFAULTIF character can be any single-byte character except the following values:
  - the null character (X'00')
  - carriage return or line feed (X'0D' or X'0A')

If the input data code page is a DBCS or EUC code page, the character must be in the range X'01' to X'3F', inclusive. If the input data code page is not a single-byte code page, the character must be in the range X'01' to X'7F', inclusive. If the input code page is a single-byte code page, the character can be in the range X'01' to X'FF'.

- The field must be assigned to only one column and cannot be part of an expression that is assigned to that column. For example, consider the following **INGEST** command:

```
INGEST FROM FILE ...
  UPDATE my_table
    SET (c1, c2, c3, c4, c5) =
      ($field1, $field2, $field2, $field3, $field4+$field5)
    WHERE $field3 = 1;
```

Only \$field1 can specify DEFAULTIF. \$field2 cannot because it is assigned to multiple columns. \$field3 cannot because it is also used in a predicate. \$field4 and \$field5 cannot because they are used in an expression.

- The default value of the table column must be a constant or NULL. It cannot be a special register.

**Note:** If a column is nullable (that is, NOT NULL is not specified on the column definition) and the WITH DEFAULT clause is not specified on the column definition, the column still has a default value of NULL.

- If the table column type differs from the field type, the ingest utility converts the default value to the field type, applying the same rules used to convert input data to the field type. For example, if the table column is defined as FLOAT WITH DEFAULT 3.7 and the field is defined as INTEGER DEFAULTIF=..., the ingest utility uses a default value of 3. Similarly, if the length of the table column's default value is greater than the field length, the utility truncates the default value to the field length.

If these restrictions are violated, the ingest utility issues an error and the **INGEST** command ends.

## Examples

### Basic ingest examples

The following example inserts data from a delimited text file:

```
INGEST FROM FILE my_file.txt
  FORMAT DELIMITED
  INSERT INTO my_table;
```



The following example inserts data from a delimited text file with fields separated by a comma (the default). The fields in the file correspond to the table columns.

```
INGEST FROM FILE my_file.txt
  FORMAT DELIMITED
  (
    $field1 INTEGER EXTERNAL,
    $field2 DATE 'mm/dd/yyyy',
    $field3 CHAR(32)
  )
  INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

### Delimiter override example

The following example inserts data like the previous example, but the fields are separated by a vertical bar.

```
INGEST FROM FILE my_file.txt
  FORMAT DELIMITED by '|'
  (
    $field1 INTEGER EXTERNAL,
    $field2 DATE 'mm/dd/yyyy',
    $field3 CHAR(32)
  )
  INSERT INTO my_table
  VALUES($field1, $field2, $field3);
```

### Omitting the field definition and VALUES list example

In the following example, the table is defined as follows:

```
CREATE TABLE my_table (
  c1 VARCHAR(32),
  c2 INTEGER GENERATED BY DEFAULT AS IDENTITY,
  c3 INTEGER GENERATED ALWAYS AS (c2 + 1),
);
```

The user issues the following **INGEST** command:

```
INGEST FROM FILE my_file.txt
  FORMAT DELIMITED
  INSERT INTO mytable;
```

- The default field definition list will be:

```
(
  $C1 CHARACTER(32),
  $C2 INTEGER EXTERNAL,
  $C3 INTEGER EXTERNAL
)
```

- The default VALUES list on the INSERT statement is:

```
VALUES($C1, $C2, DEFAULT)
```

Note that the third value is DEFAULT because the column that corresponds to field \$C3 is defined as GENERATED ALWAYS. The fourth value is omitted because it has no field.

### Extra fields used to compute column values example

The following example is the same as the delimiter override example, but only the first two fields correspond to the first two table columns (PROD\_ID and DESCRIPTION), whereas the value for the third table column (TOTAL\_PRICE) is computed from the remaining three fields

## INGEST

```
INGEST FROM FILE my_file.txt
FORMAT DELIMITED BY '|'
(
  $prod_ID    CHAR(8),
  $description CHAR(32),
  $price      DECIMAL(5,2) EXTERNAL,
  $sales_tax  DECIMAL(4,2) EXTERNAL,
  $shipping   DECIMAL(3,2) EXTERNAL
)
INSERT INTO my_table(prod_ID, description, total_price)
VALUES($prod_id, $description, $price + $sales_tax + $shipping);
```

### Filler fields example

The following example inserts data from a delimited text file with fields separated by a comma (the default). The fields in the file correspond to the table columns except that there are extra fields between the fields for columns 2 and 3 and columns 3 and 4.

```
INGEST FROM FILE my_file.txt
FORMAT DELIMITED
(
  $field1 INTEGER,
  $field2 CHAR(8),
  $filler1 CHAR,
  $field3 CHAR(32),
  $filler2 CHAR,
  $field4 DATE
)
INSERT INTO my_table VALUES($field1, $field2, $field3, $field4);
```

### Format modifiers example

The following example inserts data from a delimited text file in code page 850. Date fields are in American format and char fields are enclosed in equal signs.

```
INGEST FROM FILE my_file.txt
FORMAT DELIMITED
INPUT CODEPAGE 850
(
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32) ENCLOSED BY '='
)
INSERT INTO my_table
VALUES($field1, $field2, $field3);
```

### Positional example

The following example inserts data from a file with fields in the specified positions. The fields in the file correspond to the table columns.

```
INGEST FROM FILE my_file.txt
FORMAT POSITIONAL
(
  $field1 POSITION(1:8)    INTEGER EXTERNAL,
  $field2 POSITION(10:19) DATE 'yyyy-mm-dd',
  $field3 POSITION(25:34) CHAR(10)
)
INSERT INTO my_table
VALUES($field1, $field2, $field3);
```

## DEFAULTIF examples

This example is similar to the previous example, except if the second field starts with a blank, the ingest utility inserts the default value:

```
INGEST FROM FILE my_file.txt
FORMAT POSITIONAL
(
  $field1 POSITION(1:8) INTEGER EXTERNAL,
  $field2 POSITION(10:19) DATE 'yyyy-mm-dd' DEFAULTIF = ' ',
  $field3 POSITION(25:34) CHAR(10)
)
INSERT INTO my_table
VALUES($field1, $field2, $field3);
```

This example is the same as the previous example, except that the default indicator is in the column after the data columns:

```
INGEST FROM FILE my_file.txt
FORMAT POSITIONAL
(
  $field1 POSITION(1:8) INTEGER EXTERNAL,
  $field2 POSITION(10:19) DATE 'yyyy-mm-dd' DEFAULTIF(35) = ' ',
  $field3 POSITION(25:34) CHAR(10)
)
INSERT INTO my_table
VALUES($field1, $field2, $field3);
```

## Multiple input sources example

This example inserts data from three delimited text files:

```
INGEST FROM FILE my_file.txt, my_file2.txt, my_file3.txt
FORMAT DELIMITED
(
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
)
INSERT INTO my_table
VALUES($field1, $field2, $field3);
```

## Pipe example

This example inserts data from a pipe:

```
INGEST FROM PIPE my_pipe
FORMAT DELIMITED
(
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
)
INSERT INTO my_table
VALUES($field1, $field2, $field3);
```

## Options example

This example inserts data from a delimited text file with fields separated by a comma (the default). The fields in the file correspond to the table columns. The command specifies that write rows rejected by DB2 (for example, due to constraint violations) are to be written to table EXCP\_TABLE, rows rejected due to other errors are to be discarded, and messages are to be written to file messages.txt.

## INGEST

```
INGEST FROM FILE my_file.txt
FORMAT DELIMITED
(
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
)
EXCEPTION TABLE excp_table
  MESSAGES messages.txt
INSERT INTO my_table
VALUES($field1, $field2, $field3);
```

### Restart example

This example issues an **INGEST** command (which is restartable, by default) with a specified ingest job id:

```
INGEST FROM FILE my_file.txt
FORMAT DELIMITED
(
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
)
RESTART NEW 'ingestcommand001'
INSERT INTO my_table
VALUES($field1, $field2, $field3);
```

If the command terminates before completing, you can restart it with the following command:

```
INGEST FROM FILE my_file.txt
FORMAT DELIMITED
(
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
)
RESTART CONTINUE 'ingestcommand001'
INSERT INTO my_table
VALUES($field1, $field2, $field3);
```

### Restart terminate example

This example issues the same **INGEST** command as the previous "Restart example":

```
INGEST FROM FILE my_file.txt
FORMAT DELIMITED
(
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
)
RESTART NEW 'ingestcommand001'
INSERT INTO my_table
VALUES($field1, $field2, $field3);
```

If the command terminates before completing and you do not plan to restart it, you can clean up the restart records with the following command.

```
INGEST FROM FILE my_file.txt
FORMAT DELIMITED
(
  $field1 INTEGER EXTERNAL,
  $field2 DATE 'mm/dd/yyyy',
  $field3 CHAR(32)
```

```

)
RESTART TERMINATE 'ingestcommand001'
INSERT INTO my_table
VALUES($field1, $field2, $field3);

```

After issuing this command, you can no longer restart the **INGEST** command with the job id: "ingestcommand001", but you can reuse that string on the **RESTART NEW** parameter of a new **INGEST** command.

## Reordering columns example

This example inserts data from a delimited text file with fields separated by a comma. The table has three columns and the fields in the input data are in the reverse order of the table columns.

```

INGEST FROM FILE my_file.txt
FORMAT DELIMITED
(
$field1 INTEGER EXTERNAL,
$field2 DATE 'mm/dd/yyyy',
$field3 CHAR(32)
)
INSERT INTO my_table
VALUES($field3, $field2, $field1);

```

## Basic UPDATE, MERGE, and DELETE examples

The following examples update the table rows whose primary key matches the corresponding fields in the input file.

```

INGEST FROM FILE my_file.txt
FORMAT DELIMITED
(
$key1 INTEGER EXTERNAL,
$key2 INTEGER EXTERNAL,
$data1 CHAR(8),
$data2 CHAR(32),
$data3 DECIMAL(5,2) EXTERNAL
)
UPDATE my_table
SET (data1, data2, data3) = ($data1, $data2, $data3)
WHERE (key1 = $key1) AND (key2 = $key2);

```

or

```

INGEST FROM FILE my_file.txt
FORMAT DELIMITED
(
$key1 INTEGER EXTERNAL,
$key2 INTEGER EXTERNAL,
$data1 CHAR(8),
$data2 CHAR(32),
$data3 DECIMAL(5,2) EXTERNAL
)
UPDATE my_table
SET data1 = $data1, data2 = $data2, data3 = $data3
WHERE (key1 = $key1) AND (key2 = $key2);

```

This example merges data from the input file into the target table. For input rows whose primary key fields match a table row, it updates that table row with the input row. For other input rows, it adds the row to the table.

```

INGEST FROM FILE my_file.txt
FORMAT DELIMITED
(

```

## INGEST

```
$key1 INTEGER EXTERNAL,  
$key2 INTEGER EXTERNAL,  
$data1 CHAR(8),  
$data2 CHAR(32),  
$data3 DECIMAL(5,2) EXTERNAL  
)  
MERGE INTO my_table  
ON (key1 = $key1) AND (key2 = $key2)  
WHEN MATCHED THEN  
    UPDATE SET (data1, data2, data3) = ($data1, $data2, $data3)  
WHEN NOT MATCHED THEN  
    INSERT VALUES($key1, $key2, $data1, $data2, $data3);
```

This example deletes table rows whose primary key matches the corresponding primary key fields in the input file.

```
INGEST FROM FILE my_file.txt  
FORMAT DELIMITED  
(  
    $key1 INTEGER EXTERNAL,  
    $key2 INTEGER EXTERNAL  
)  
DELETE FROM my_table  
WHERE (key1 = $key1) AND (key2 = $key2);
```

### Complex SQL examples

Consider the following example in which there is a table with columns KEY, DATA, and ACTION. The following command updates the DATA column of table rows where the primary key column (KEY) matches the corresponding field in the input file and the ACTION column is 'U':

```
INGEST FROM FILE my_file.txt  
FORMAT DELIMITED  
(  
    $key_fld INTEGER EXTERNAL,  
    $data_fld INTEGER EXTERNAL  
)  
UPDATE my_table  
SET data = $data_fld  
WHERE (key = $key_fld) AND (action = 'U');
```

The following example is the same as the previous example except that if the keys match and the ACTION column is 'D', then it deletes the row from the table:

```
INGEST FROM FILE my_file.txt  
FORMAT DELIMITED  
(  
    $key_fld INTEGER EXTERNAL,  
    $data_fld INTEGER EXTERNAL  
)  
MERGE INTO my_table  
ON (key1 = $key_fld)  
WHEN MATCHED AND (action = 'U') THEN  
    UPDATE SET data = $data_fld  
WHEN MATCHED AND (action = 'D') THEN  
    DELETE;
```

### Usage notes

#### Using INGEST on the system command line (UNIX)

If you specify the **INGEST** command on the system command line in a UNIX command shell, you must specify field names in single quotation marks or put an escape character (backslash) before the \$ that starts the field name. Otherwise, the

command shell interprets the field names as environment variables and replace them with the value of the corresponding environment variable.

For example, suppose environment variable *\$field1* has value abc and environment variable *\$field2* has no value, and you enter the following command in the UNIX command shell:

```
db2 INGEST ... FORMAT DELIMITED ($field1 INTEGER, $field2 CHAR) ...
```

The DB2 CLP sees the following command, which is not valid:

```
INGEST ... FORMAT DELIMITED (abc INTEGER, CHAR) ...
```

You can avoid this by using single quotation marks, as follows:

```
db2 'INGEST ... FORMAT DELIMITED ($field1 INTEGER, $field2 CHAR) ...'
```

If the **INGEST** command contains single quotation marks, you can enclose the entire command in double quotation marks " and use escape characters \ before the \$ that starts field names. For example:

```
db2 "INGEST ... FORMAT DELIMITED BY X'7C' (\$field1 ...) ..."
```

The simplest way to avoid all the previously mentioned problems is to put the command in a file and specify the -f option on the db2 command.

## Categories of error

For purposes of determining how the ingest utility handles errors, we can group errors into the following categories:

### 1. Start-up errors

These errors include:

- syntax errors
- an input file is not found or not readable
- the target or exception table is not found
- the dump file or messages file cannot be opened
- not enough memory
- other errors detected at startup

When the utility detects any of these errors, it issues an error message and exits without ingesting any data.

### 2. Data errors

These errors include:

- Errors in the input data that the formatter detects, for example:
  - numbers that are invalid or out of range (based on the field type)
  - some dates, times, timestamps that do not fit the specified format
  - other errors detected by the formatters
- Errors from SQL statements. These are listed in the description of the EXCEPTION TABLE parameter on the **INGEST** command. The most common are
  - data too long, out of range, or incorrect format (not caught by formatter)
  - constraint violations on the target table, including not null, unique, referential, or check
  - range violations on the target table

When the utility detects any of these errors, it issues a warning or error message and continues on. In the case of formatter errors, the utility also writes the record to the dump file (if specified). In the case of SQL errors, the utility also inserts the row into the exception table (if specified). To have the utility end after a certain number of these errors, use the `WARNINGCOUNT` parameter.

### 3. Recoverable errors

These are errors that might go away if the utility reconnects (if needed), waits some small amount of time, and tries again. When the utility detects any of these errors and the `retry_count` or `reconnect_count` configuration parameter is greater than 0, the utility attempts to recover from the error. If `retry_count` and `reconnect_count` are 0, or the utility cannot recover, the utility considers these errors to be *terminating errors*.

### 4. Terminating errors

These are all other errors not listed in the previous three types. When the utility detects any of these errors, it issues an error message and exits. Any data that was committed remains in the target table. You can restart the command from the last commit point.

## Messages from the INGEST command

If the utility read at least one record from the input source, the utility issues a summary of the number of rows read, inserted, and rejected (similar to the import and load utilities) and a successful completion message. For example:

```
INGEST FROM FILE my_table.del FORMAT DELIMITED
($field1 INTEGER_EXTERNAL, $field2 CHAR(32))
INSERT INTO my_table VALUES($field1, $field2)
Number of rows read      = 6
Number of rows inserted = 6
Number of rows rejected = 0
SQL2980I The ingest utility completed successfully at timestamp "11/02/2011 12:34:56.123456".
```

The meaning of the “Number of rows...” messages is as follows:

#### Number of rows read

The number of records the utility read from the input source.

#### Number of rows inserted (updated, deleted, merged)

The number of rows affected by the execution of the SQL statement against the target table and committed to the database. The message says “inserted”, “updated”, “deleted”, or “merged”, depending on the SQL statement.

#### Number of rows rejected

The number of rows rejected (by the utility or DB2).

If there is only one warning or error and the error is such that no input records could be processed, the command issues only that error message and does not display the summary of rows read, rejected, and so on. For example:

```
INGEST FROM FILE bad_file.del FORMAT DELIMITED
($field1 INTEGER_EXTERNAL, $field2 CHAR)
INSERT INTO my_table VALUES($field1, $field2)

SQL2036N The path for the file or device "bad_file.del" is not valid.
```

After the **INGEST** command begins reading records, if there are one or more warning or error conditions detected, the command issues messages, along with



the summary of rows read, skipped, and so on, and a summary of the number of warnings and errors. Consider the following example which tries to insert two duplicate keys:

```
INGEST FROM FILE my_table1.del, my_table2.del FORMAT DELIMITED
( $field1 integer external, $field2 char(32) )
INSERT INTO my_table VALUES($field1, $field2)
```

```
SQL2905I The following error occurred issuing the SQL "INSERT"
statement on table "ALVEYWD.MY_TABLE" using data from line "6" of
input file "my_table1.del".
```

```
SQL0803N One or more values in the INSERT statement, UPDATE statement,
or foreign key update caused by a DELETE statement are not valid
because the primary key, unique constraint or unique index identified
by "1" constrains table "ALVEYWD.MY_TABLE" from having duplicate values
for the index key.  SQLSTATE=23505
```

```
SQL2905I The following error occurred issuing the SQL "INSERT" statement
on table "ALVEYWD.MY_TABLE" using data from line "9" of input file "my_table2.del".
```

```
SQL0803N One or more values in the INSERT statement, UPDATE statement,
or foreign key update caused by a DELETE statement are not valid because
the primary key, unique constraint or unique index identified by "1"
constrains table "ALVEYWD.MY_TABLE" from having duplicate values
for the index key.  SQLSTATE=23505
```

```
Number of rows read      = 6
Number of rows inserted = 4
Number of rows rejected  = 2
Number of rows committed = 4
```

```
SQL2902I The ingest utility completed at timestamp "11/02/2011 12:34:56.123456".
Number of errors: 2. Number of warnings: 0.
```

If the MESSAGES parameter is specified, most messages are written to that file, except as specified in the description of that parameter.

## File and pipe names

The metavariables *file-name* and *pipe-name* specify a file or pipe name that follows the operating system syntax. File or pipe names specified on the **INGEST** command that contain any characters other than the following must be enclosed in single quotation marks:

- alphanumeric
- underscore (\_)
- dash (-)
- forward-slash (/) or back-slash (\)
- dot (.)

The quotation marks need to be around the entire name. For example, if a filename contains the equal sign (=), specify:

```
'dir1/dir2/my=file'
```

instead of

```
dir1/dir2/'my=file'
```

## Strings and characters

The metavariable *string* specifies any character string constant, including:

- A sequence of characters that starts and ends with a string delimiter which is an apostrophe (').

- A hexadecimal constant: X followed by a sequence of hex characters that starts and ends with an apostrophe.
- A Unicode string constant: U& followed by a sequence of characters that starts and ends with an apostrophe and that is optionally followed by the UESCAPE clause.
- A graphic string constant: G or N followed by a sequence of double-byte characters that starts and ends with an apostrophe.
- A hexadecimal graphic string constant: GX followed by a sequence of hex characters that starts and ends with an apostrophe.
- A UCS-2 graphic string constant: UX followed by a sequence of hex characters that starts and ends with an apostrophe.

The metavariable `char` specifies a character string constant that is 1 byte in length. Examples of this are `'A'` and `X'30'`.

If the string contains the enclosing character, it must be doubled inside the string. For example, if the string is "don't know" and the enclosing character is a single quote, specify `'don''t know'`. Examples of this are `'This is a string'`, `X'303132'`, and `U&'Unicode chars'`.

### Default field definition list

When all of the following are true, you can omit the field definition list:

- The format is delimited.
- The SQL statement is INSERT.
- The VALUES clause is omitted from the INSERT statement.

The field definition list defaults as follows:

- If a column list follows the table name on the INSERT statement, there is one field for each column in the list.
- If the INSERT statement omits the column list and there are no implicitly hidden columns, then there is one field for each column in the table.
- If the INSERT statement omits the column list and there are implicitly hidden columns, then you must explicitly specify whether or not the implicitly hidden columns are included. Use the `DB2_DMU_DEFAULT` registry variable, or the `IMPLICITLYHIDDENINCLUDE` or `IMPLICITLYHIDDENMISSING` keywords to specify if implicitly hidden columns are included.
- Each field has the same name as the corresponding table column, prefixed with a dollar sign \$.
- Each field has the same data type and length (or precision and scale) as the corresponding table column. Numeric fields (integer, decimal, and so on) default to EXTERNAL format. `DB2SECURITYLABEL` fields default to STRING format.

Table 21 on page 325 shows how the field, column, and values lists default for every possible combination of omitting or including those lists.

Table 21. Possible combinations when field definition list is specified or omitted

Field definition list specified?	Column list specified?	VALUES clause specified?	Defaults
no	no	no	Field list and column list default to all columns. Values list defaults to field list. <b>Note:</b> If the table contains implicitly hidden columns, then you must explicitly specify whether or not the implicitly hidden columns are included using the DB2_DMU_DEFAULT registry variable, or the IMPLICITLYHIDDENINCLUDE or IMPLICITLYHIDDENMISSING keywords.
no	no	yes	Not allowed
no	yes	no	Field list and values list default to the specified column list
no	yes	yes	Not allowed
yes	no	no	Column list defaults to all columns. Values list defaults to user-specified field list. <b>Notes:</b> 1. The number of fields must be same as number of columns. Otherwise an error occurs. 2. If the table contains implicitly hidden columns, then you must explicitly specify whether or not the implicitly hidden columns are included using the DB2_DMU_DEFAULT registry variable, or the IMPLICITLYHIDDENINCLUDE or IMPLICITLYHIDDENMISSING keywords.
yes	no	yes	Column list defaults to all columns. <b>Note:</b> If the table contains implicitly hidden columns, then you must explicitly specify whether or not the implicitly hidden columns are included using the DB2_DMU_DEFAULT registry variable, or the IMPLICITLYHIDDENINCLUDE or IMPLICITLYHIDDENMISSING keywords.

Table 21. Possible combinations when field definition list is specified or omitted (continued)

Field definition list specified?	Column list specified?	VALUES clause specified?	Defaults
yes	yes	no	Values list defaults to user-specified field list. <b>Note:</b> Number of fields must be same as number of columns. Otherwise an error occurs.
yes	yes	yes	No defaults

## Rules and defaults for field lengths

Field lengths specified on field types are in bytes and must be 1 - 32 767, inclusive.

The following tables show the length of a field under all combinations of input data format, field type, and whether the length and ending position is specified. The cell value *the specified length* means that the field length is explicitly specified; for example, the length that can follow "INTEGER EXTERNAL". It is not the precision that is on some numeric types. The cell value *n/a* means not applicable or not allowed.

Table 22. Field lengths for DELIMITED format

Field type	Field type omits length (or datetime format) <sup>1</sup>	Field type specifies length (or datetime format) <sup>1</sup>
CHARACTER	255 <sup>2</sup>	the specified length
SMALLINT , INTEGER, BIGINT EXTERNAL	255 <sup>2</sup>	the specified length
DECIMAL(p,s) EXTERNAL (default for (p,s) is (5,0))	255 <sup>2</sup>	the specified length
DECFLOAT(16 or 34) EXTERNAL	255 <sup>2</sup>	the specified length
REAL EXTERNAL	255 <sup>2</sup>	the specified length
FLOAT or DOUBLE EXTERNAL	255 <sup>2</sup>	the specified length
DATE	<ul style="list-style-type: none"> <li>• <b>date_compat</b> off: length of maximum length default format for a date (10)</li> <li>• <b>date_compat</b> on: length of maximum length default format for a timestamp (32)</li> </ul>	<ul style="list-style-type: none"> <li>• <b>date_compat</b> off: length of maximum length default format for a date (10)</li> <li>• <b>date_compat</b> on: length of maximum length default format for a timestamp (32)</li> </ul>
TIME	length of maximum length default format (8)	length of max length default format (8)
TIMESTAMP(p) (default for p is 6)	length of maximum length default format (32)	length of max length default format (32)
DB2SECURITYLABEL with NAME modifier	255 <sup>2</sup>	the specified length
DB2SECURITYLABEL with STRING modifier	255 <sup>2</sup>	the specified length

Table 22. Field lengths for DELIMITED format (continued)

Field type	Field type omits length (or datetime format) <sup>1</sup>	Field type specifies length (or datetime format) <sup>1</sup>
<p><sup>1</sup> In delimited format, the field length means the length to which the utility truncates the field data if it exceeds that length. The truncation occurs after applying the trim option. If the truncation results in loss of non-blank characters, the utility issues a warning.</p> <p><sup>2</sup> In delimited format, character, and numeric fields that are specified in ASCII and omit the length have a default length of 255 bytes. If you need a longer length, specify an explicit length on the field type.</p>		

Table 23. Field lengths for POSITIONAL format

Field type	Field type omits length (or date/time format), POSITION omits end position	Field type specifies length (or date/time format), POSITION omits end position	Field type omits length (or date/time format), POSITION specifies end position	Field type specifies length (or date/time format), and POSITION specifies end position
CHARACTER	$n/a^3$	the specified length	$(end - start + 1)$	the lesser of the specified length and $(end - start + 1)^7$
SMALLINT (binary)	2	$n/a^4$	If $(end - start + 1)$ is 2, 4, or 8, use that. Otherwise, use length of binary type. If conflict, issue warning. <sup>5</sup>	$n/a^4$
INTEGER (binary)	4	$n/a^4$	If $(end - start + 1)$ is 2, 4, or 8, use that. Otherwise, use length of binary type. If conflict, issue warning. <sup>5</sup>	$n/a^4$
BIGINT (binary)	8	$n/a^4$	If $(end - start + 1)$ is 2, 4, or 8, use that. Otherwise, use length of binary type. If conflict, issue warning. <sup>5</sup>	$n/a^4$
SMALLINT, INTEGER, BIGINT EXTERNAL	$n/a^3$	the specified length, which must be $\leq 50$	$(end - start + 1)$ , which must be $\leq 50$	the lesser of the specified length and $(end - start + 1)^7$
DECIMAL(p,s) PACKED (default for (p,s) is (5,0))	$(p+2)/2$	$n/a^4$	Use $(p+2)/2$ . If conflict with end position, issue warning. <sup>5</sup>	$n/a^4$
DECIMAL(p,s) ZONED (default for (p,s) is (5,0))	p	$n/a^4$	Use precision. If conflict with end position, issue warning. <sup>5</sup>	$n/a^4$
DECIMAL(p,s) EXTERNAL (default for (p,s) is (5,0))	$n/a^3$	the specified length	$(end - start + 1)$	the lesser of the specified length and $(end - start + 1)^7$

## INGEST

Table 23. Field lengths for POSITIONAL format (continued)

Field type	Field type omits length (or date/time format), POSITION omits end position	Field type specifies length (or date/time format), POSITION omits end position	Field type omits length (or date/time format), POSITION specifies end position	Field type specifies length (or date/time format), and POSITION specifies end position
DECFLOAT(16) (binary)	8	n/a <sup>4</sup>	Use 8. If conflict with end position, issue warning. <sup>5</sup>	n/a <sup>4</sup>
DECFLOAT(34) (binary)	16	n/a <sup>4</sup>	Use 16. If conflict with end position, issue warning. <sup>5</sup>	n/a <sup>4</sup>
DECFLOAT(16 or 34) EXTERNAL	n/a <sup>3</sup>	the specified length	( <i>end - start + 1</i> )	the lesser of the specified length and ( <i>end - start + 1</i> ) <sup>7</sup>
REAL (binary)	4	n/a <sup>4</sup>	Use 4. If conflict with end position, issue warning. <sup>5</sup>	n/a <sup>4</sup>
REAL EXTERNAL	n/a <sup>3</sup>	the specified length	( <i>end - start + 1</i> )	the lesser of the specified length and ( <i>end - start + 1</i> ) <sup>7</sup>
FLOAT or DOUBLE (binary)	8	n/a <sup>4</sup>	Use 8. If conflict with end position, issue warning. <sup>5</sup>	n/a <sup>4</sup>
FLOAT or DOUBLE EXTERNAL	n/a <sup>3</sup>	the specified length	( <i>end - start + 1</i> )	the lesser of the specified length and ( <i>end - start + 1</i> ) <sup>7</sup>
DATE	<ul style="list-style-type: none"> <li>• <b>date_compat</b> off: length of maximum length default format for a date (10)</li> <li>• <b>date_compat</b> on: length of maximum length default format for a TIMESTAMP(0) (19)</li> </ul>	length of format string, which must be $\geq 1$	<ul style="list-style-type: none"> <li>• <b>date_compat</b> off: (<i>end - start + 1</i>), which must be <math>\geq 8</math> and <math>\leq 10</math></li> <li>• <b>date_compat</b> on: (<i>end - start + 1</i>), which must be <math>\geq 19</math> and <math>\leq 32</math></li> </ul>	If ( <i>end - start + 1</i> ) is greater than or equal to length of the format string, use that. Otherwise, ( <i>end - start + 1</i> ) must be greater than or equal to the length of the shortest value that matches the format string. <sup>6</sup>
TIME	length of max length default format (8)	length of format string, which must be $\geq 1$	( <i>end - start + 1</i> ), which must be $\geq 4$ and $\leq 8$	If ( <i>end - start + 1</i> ) is greater than or equal to length of the format string, use that. Otherwise, ( <i>end - start + 1</i> ) must be greater than or equal to the length of the shortest value that matches the format string. <sup>6</sup>

Table 23. Field lengths for POSITIONAL format (continued)

Field type	Field type omits length (or date/time format), POSITION omits end position	Field type specifies length (or date/time format), POSITION omits end position	Field type omits length (or date/time format), POSITION specifies end position	Field type specifies length (or date/time format), and POSITION specifies end position
TIMESTAMP(p) (default for p is 6)	length of default format (If p = 0, then 19. Otherwise, 20 + p.)	length of format string, which must be $\geq 1$	$(end - start + 1)$ , which must be $\geq 19$ and $\leq 32$	If $(end - start + 1)$ is greater than or equal to length of the format string, use that. Otherwise, $(end - start + 1)$ must be greater than or equal to the length of the shortest value that matches the format string. <sup>6</sup>
DB2SECURITYLABEL (binary)	n/a <sup>3</sup>	the specified length	$(end - start + 1)$	the lesser of the specified length and $(end - start + 1)$ <sup>7</sup>
DB2SECURITYLABEL with NAME modifier	n/a <sup>3</sup>	the specified length	$(end - start + 1)$	the lesser of the specified length and $(end - start + 1)$ <sup>7</sup>
DB2SECURITYLABEL with STRING modifier	n/a <sup>3</sup>	the specified length	$(end - start + 1)$	the lesser of the specified length and $(end - start + 1)$ <sup>7</sup>
<p><sup>3</sup> These cases (positional format, length and end position omitted) are not allowed because the utility cannot determine the intended length of the field and there is no suitable default.</p> <p><sup>4</sup> These cases (positional format, numeric binary field, length specified, end position omitted) are not allowed because with binary numeric types, the length is implied by the binary type and the syntax does not allow specifying an explicit field length in bytes.</p> <p><sup>5</sup> The various utilities handle conflicts between the numeric binary type specified and the length specified on the POSITION parameter as shown in Table 24.</p> <p><sup>6</sup> The various utilities handle conflicts between the date/time format string length and the length specified on the POSITION parameter as shown in Table 25 on page 330.</p> <p><sup>7</sup> In these cases (length and end position are specified but conflict), the ingest utility issues a warning and uses the lesser of the specified length and the value of <math>(end - start + 1)</math>.</p>				

Table 24. Conflicts between numeric binary type and specified length

Field type	Load <sup>1</sup>	Ingest
SMALLINT, INTEGER, BIGINT	If $(end - start + 1)$ is 2, 4, or 8, use that length with no warning or error. Otherwise issue warning, and use NULL or reject the row.	If $(end - start + 1)$ is 2, 4, or 8, use that length and issue a warning. Otherwise, use the length of the binary type and issue a warning.
DECIMAL	Issue error SQL3123W and reject the row.	Use the length of the binary type and issue a warning.
REAL, FLOAT, DECFLOAT	Use the length of the binary type. Do not issue a warning or error.	Use the length of the binary type and issue a warning.

Table 24. Conflicts between numeric binary type and specified length (continued)

Field type	Load <sup>1</sup>	Ingest
<sup>1</sup> Import does not support binary numeric types (MODIFIED BY binarynumerics).		

Table 25. Conflicts between date/time format string length and specified length

Conflict	Import/Load	Ingest
$(end - start + 1)$ is greater than or equal to the length of the format string.	Use $end - start + 1$ . No warning or error.	Use $end - start + 1$ . No warning or error.
$(end - start + 1)$ is less than the length of the format string.	Issue error.	Issue error.

## How the ingest utility determines field lengths

The algorithm for determining field length is as follows:

- Determine the length of the field as it appears in the input data:
  - If the format is DELIMITED:
    - Apply the specified or default trim option. (For CHAR fields, the default is TRIM. For all other non-binary field types, the ingest utility always trims leading and trailing blanks.)
    - If ENCLOSED BY is not specified, the data length is the number of bytes left in the field.
    - If ENCLOSED BY is specified, the data length is the number of bytes between the string delimiters (after converting doubled delimiters inside the string to single delimiters).
  - If the format is POSITIONAL:
    - Apply the specified or default trim option. (For CHAR fields, the default is NOTRIM. For all other non-binary field types, the ingest utility always trims leading and trailing blanks.)
- If the length determined in step 1 is greater than the field length as described in “Rules and defaults for field lengths” on page 326:
  - For fields other than DATE, TIME, or TIMESTAMP, truncate the data to the field length. If non-blank characters were truncated, issue a warning.
  - For DATE, TIME, and TIMESTAMP fields, issue an error.
- If the SQL statement sets a column to the field value and the value is too long or out of range, DB2 issues an error message and rejects the row.

**Note:** This can happen only on an INSERT, UPDATE, or MERGE statement.

The following tables show some examples. In these tables, the tilde (~) means a blank.

Table 26. FORMAT DELIMITED BY '|'

Field definition	Input field	Final field value
CHAR TRIM	~ABC~	ABC
CHAR NOTRIM	~ABC~	~ABC~
CHAR OPTIONALLY ENCLOSED BY ''''	~"ABC"~	ABC



Table 26. *FORMAT DELIMITED BY 'I'* (continued)

Field definition	Input field	Final field value
CHAR OPTIONALLY ENCLOSED BY '''	~"~ABC~"~	~ABC~
CHAR NOTRIM OPTIONALLY ENCLOSED BY '''	~"~ABC~"	~ABC~
CHAR NOTRIM OPTIONALLY ENCLOSED BY '''	~"ABC"~	~"ABC"~
CHAR(1) TRIM	~ABC~	A (and warning because non-blanks were trimmed)
CHAR(1) NOTRIM	~ABC~	~ (and warning because non-blanks were trimmed)
CHAR(1) OPTIONALLY ENCLOSED BY '''	~ABC~	A (and warning because non-blanks were trimmed)
CHAR(6) TRIM	~ABC~	ABC
CHAR(6) NOTRIM	~ABC~	~ABC~
CHAR(6) OPTIONALLY ENCLOSED BY '''	~"ABC"~	ABC
CHAR(6) OPTIONALLY ENCLOSED BY '''	~"~ABC~"~	~ABC~
INTEGER EXTERNAL	~12345~	12345
INTEGER EXTERNAL(3)	~12345~	123 (and warning because non-blanks were trimmed)
INTEGER EXTERNAL(9)	~12345~	12345

Table 27. *FORMAT POSITIONAL*

Field definition	Input field	Final field value
POSITION(1:5) CHAR TRIM	~ABC~	ABC
POSITION(1:5) CHAR NOTRIM	~ABC~	~ABC~
POSITION(1:5) CHAR(3) TRIM	ABC	A (and warning because end position conflicts with length)
POSITION(1:5) CHAR(5) NOTRIM	~ABC~	~ABC~
POSITION(1:5) CHAR(5) TRIM	~ABC~	ABC
POSITION(1:5) CHAR(7) NOTRIM	~ABC~	~ABC~ (and warning because end position conflicts with length)
POSITION(1:7) INTEGER EXTERNAL	~12345~	12345
POSITION(1:7) INTEGER EXTERNAL(3)	123	123 (and warning because end pos conflicts with length)

Table 27. *FORMAT POSITIONAL* (continued)

Field definition	Input field	Final field value
POSITION(1:7) INTEGER EXTERNAL(9)	~12345~	12345 (and warning because end position conflicts with length)

## Handling of invalid numeric data

In general, when numeric data is specified as a character string and the data cannot be converted to a number or is out of range for the field type, the ingest utility rejects the record and issues an error. This is different than the import and load utilities, which handle these errors as follows:

- For all numeric types, if the data cannot be converted to the column's type (for example, "ABC" in an integer field):
  - If the column is nullable, the import and load utilities replace the value with NULL and issue a warning.
  - If the column is not nullable, the import and load utilities reject the record and issue an error.
- For all numeric types except DECIMAL, if the data is out of range:
  - If the column is nullable, the import and load utilities replace the value with NULL and issue a warning.
  - If the column is not nullable, the import and load utilities reject the record and issue an error.
- For DECIMAL fields, if the data is out of range, the import and load utilities reject the record and issue an error.

## Notes regarding the FORMAT clause

- The priority for delimiters is the same as for the load utility: record, character, field. The import and load utilities also support the `delprioritychar` file type modifier, which changes the priority to character, record, field, but the ingest utility does not.
- When modifiers are specified on a field, they must be specified on all fields that have a similar type. For example, if a decimal field specifies RADIX POINT, the same RADIX POINT character must be specified on all other fields of type INTEGER, DECIMAL, DECLFOAT, and FLOAT.
- For character fields, the various TRIM options specify whether the ingest utility strips leading and/or trailing blanks. For all other non-binary field types (for example, INTEGER EXTERNAL), the ingest utility always trims leading and trailing blanks.
- When the format is DELIMITED or the format is POSITIONAL and the RECORDLEN parameter is omitted:
  - If a record contains fewer fields than defined in the format, the ingest utility considers the missing fields to be NULL.
  - If a record contains extra fields, the ingest utility ignores them.
- When the format is POSITIONAL:
  - If the RECORDLEN parameter is specified and the last record is too short, the ingest utility issues an error and rejects the record.
  - If there is data outside the specified field positions, the ingest utility ignores it. For example, if the field specifies two fields at POSITION(1:10) and

POSITION(21:30), the ingest utility ignores data in positions 11:20 and any data past position 30 to the end of the record.

### SQL statements on the INGEST command

- The ingest utility supports the following values in the PERIOD clause:
  - constants
  - special registers

**Note:** Only the following special registers affect the execution of the SQL statement on the **INGEST** command:

- CURRENT SCHEMA
- CURRENT TEMPORAL SYSTEM\_TIME
- CURRENT TEMPORAL BUSINESS\_TIME
- scalar functions whose arguments are supported operands (although nested function invocations and user-defined functions cannot be used)
- CAST specifications where the cast operand is a supported operand expression using arithmetic operators and operands
- expressions using arithmetic operators and operands

You can also specify a field name, which the utility converts to a parameter marker.

- There are several other considerations for ingesting into a temporal table:
  - When special register CURRENT TEMPORAL SYSTEM\_TIME is set, you cannot ingest into a system-period temporal table or a bi-temporal table.
  - When special register CURRENT TEMPORAL BUSINESS\_TIME is set, you can still ingest into a business-time temporal table or a bi-temporal table (subject to the other restrictions on system-time temporal tables).
  - You cannot use the REPLACE or DELETE operation on a system-period temporal table or bi-temporal table.
  - The ingest utility cannot insert values for the SYSTEM\_TIME period in a system-period temporal table.
- Any strings in the SQL statements are assumed to be in the application code page.
- Any column names that start with a dollar sign (\$) must be specified as SQL delimited identifiers (enclosed in double quotation marks), in order to distinguish them from field names.
- In most cases, after converting field names to parameter markers and binding in values for the fields, the ingest utility passes these statements *as is* to DB2. They are therefore subject to the same restrictions. For example:
  - Fields are bound as their field type. Therefore they can be used only where a value of the specified type is allowed. For example, fields of type DB2SECURITYLABEL can be used only where a security label is allowed.
  - If the INSERT statement omits the column list but specifies the VALUES list, the VALUES list must contain an item for each column. If the table contains implicitly hidden columns, then you must explicitly specify whether or not the implicitly hidden columns are included. Use the DB2\_DMU\_DEFAULT registry variable, or the IMPLICITLYHIDDENINCLUDE or IMPLICITLYHIDDENMISSING keywords to specify if implicitly hidden columns are included.
- The ingest utility issues an error and ends the **INGEST** command if any of the following conditions is true:

- If the SQL statement specified on the command does not reference any fields
- The SQL statement is DELETE or UPDATE and it has no WHERE clause or the WHERE clause does not reference any fields.
- The SQL statement is MERGE and the ON clause does not reference any fields.
- The default isolation level is cursor stability (CS). This can be overridden by specifying the WITH clause on the SQL statement (except INSERT), or by setting the CLI IsolationLevel keyword in the file specified by the **DB2CLIINIPATH** environment variable (the default is sql11ib/cfg/db2cli.ini). Setting the IsolationLevel keyword, however, affects all CLI applications.

## The MERGE statement

The **INGEST** command issues the MERGE statement once for each input data record, treating the record as a one-row table (equivalent to the USING *table-reference* parameter on the SQL MERGE statement). This can occasionally produce different results than the equivalent SQL MERGE statement. Consider the following example:

- Table SOURCE\_TABLE contains two rows:
  - (1, 'source data 1')
  - (2, 'source data 2')
- Input file source\_table.del contains the same two records as in SOURCE\_TABLE.
- Table TARGET\_TABLE contains one row:
  - (NULL, 'target data 1')

The SQL MERGE statement is:

```
MERGE INTO target_table
  USING (SELECT * FROM source_table) source_table
  ON target_table.c1 <> source_table.c1
  WHEN NOT MATCHED THEN
    INSERT VALUES(source_table.c1, source_table.c2);
```

The **INGEST** command (using the equivalent MERGE statement) is:

```
INGEST FROM FILE source_table.del
  FORMAT DELIMITED
  (
    $c1 INTEGER EXTERNAL,
    $c2 CHAR(32)
  )
  MERGE INTO target_table
  ON c1 <> $c1
  WHEN NOT MATCHED THEN
    INSERT VALUES($c1, $c2);
```

Following the SQL MERGE statement, TARGET\_TABLE contains the following rows:

```
(NULL, 'target data 1')
(1, 'source data 1')
(2, 'source data 2')
```

Following the **INGEST** command, TARGET\_TABLE contains the following rows:

```
(NULL, 'target data 1')
(1, 'source data 1')
```

The reason the **INGEST** command results in fewer rows is that after the **INGEST** command processes the first input data record, the target table

contains the two rows shown previously. From that point onwards, the ON clause has a matching row, which means the INSERT statement on the WHEN NOT MATCHED clause is not processed again.

Note that this example is not a common use of the MERGE statement. When the MERGE statement has the more common format of updating or inserting rows based on matching keys, the SQL MERGE statement and the **INGEST** command produce the same results.

### The REPLACE statement

REPLACE is basically the same as INSERT except that the ingest utility issues the DELETE statement to delete all the table rows before beginning the inserts. If the table is large, the DELETE could take some time and use significant log space.

### Data type conversion

For each field the utility ingests, there are two places that data conversion can occur:

- When the input data is in character format (including numbers specified in ASCII), the utility converts the data from character to the field type.
- If a field's type differs from its corresponding column type, DB2 converts the field value from the field type to the column type.

In both cases, the ingest utility and DB2 use the rules described in the topic "Assignments and comparisons". This can occasionally produce different results than converting directly from character to the column type. Consider the following example:

- The input file specifies string '9.99E6143', which is a valid DECFLOAT(34) number but outside the range of DECFLOAT(16).
- The field type is DECFLOAT(34).
- The column type is DECFLOAT(16).

In this example, the ingest utility converts the field value from character to DECFLOAT(34). Then DB2 converts from DECFLOAT(34) to DECFLOAT(16). That results in a warning and a value of DECFLOAT infinity in the DECFLOAT(16) column.

### Casting a field to an SQL data type

Before issuing the SQL statement that you specify on the **INGEST** command, the ingest utility converts field names to parameter markers and then issues the SQL statement dynamically. In order to determine the data types of the parameter markers, DB2 uses the rules described in the topic "Determining data types of untyped expressions". Just as with dynamic SQL, if the data type of a field is not obvious from the context where it is used, you might get an error, in which case you need to do perform one of the following actions:

- Set the **DB2\_DEFERRED\_PREPARE\_SEMANTICS** registry variable to YES.
- Use a CAST specification to cast the field name to an SQL data type.

For example, consider the following **INGEST** command:

```
INGEST FROM FILE my_file.txt
  FORMAT DELIMITED ( $date_fld DATE )
  INSERT INTO my_table(int_col) VALUES(day($date_fld));
```

If you have the **DB2\_DEFERRED\_PREPARE\_SEMANTICS** registry variable set to NO (the default), when the ingest utility tries to issue the statement, it might return an error because DB2 cannot determine which version of the DAY

## INGEST

function to use. (This is true even though you defined the field as a DATE and the ingest utility binds it as a DATE.)

You can fix the problem by casting field names to their corresponding field types, for example:

```
INSERT INTO my_table(int_col) VALUES( day(CAST($date_fld AS DATE)) );
```

In some cases, you can also fix the problem by setting registry variable DB2\_DEFERRED\_PREPARE\_SEMANTICS=YES. However, this does not always work.

As an additional example, consider the following **INGEST** command:

```
INGEST FROM FILE my_file.txt
  FORMAT DELIMITED ( $field1 DECIMAL(5,2) )
  INSERT INTO my_table(dec_col) VALUES($field1 + 1);
```

Because \$field1 is added to an integer, DB2 assigns type INTEGER to the field. In order to have DB2 assign type DECIMAL(5,2), you need to change the SQL statement to:

```
INSERT INTO my_table(dec_col) VALUES(CAST($field1 AS DECIMAL(5,2)) + 1);
```

or

```
INSERT INTO my_table(dec_col) VALUES($field1 + 001.00);
```

## 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 authorities:

- SYSADM
- SYSCTRL
- SYSMANT

### 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.

## INSPECT

Inspects database for architectural integrity, checking the pages of the database for page consistency. The **INSPECT** command checks that the structures of table objects and structures of table spaces are valid. Cross object validation conducts an online index to data consistency check.

### Scope

In a single partition database environment, the scope is that single partition only. In a partitioned database environment, 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

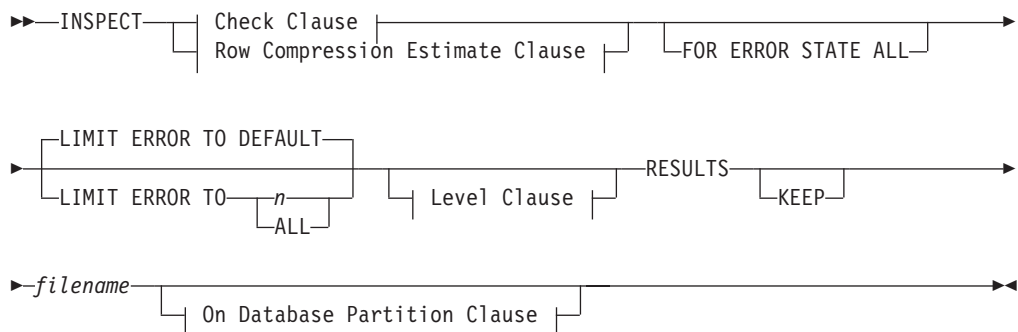
For **INSPECT CHECK**, one of the following authorities:

- SYSADM
- DBADM
- SYSCTRL
- SYSMANT
- CONTROL privilege if single table.

### Required Connection

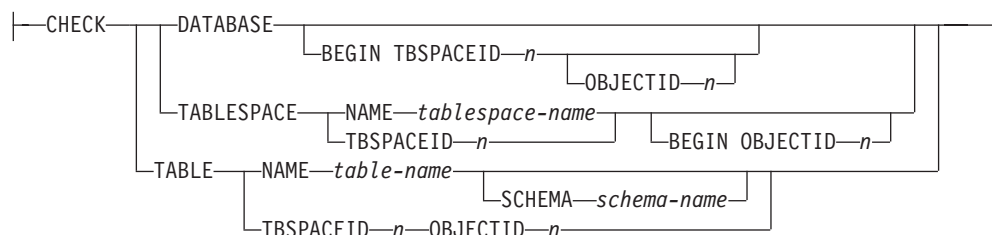
Database

### Command Syntax

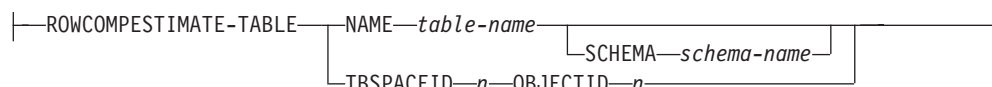




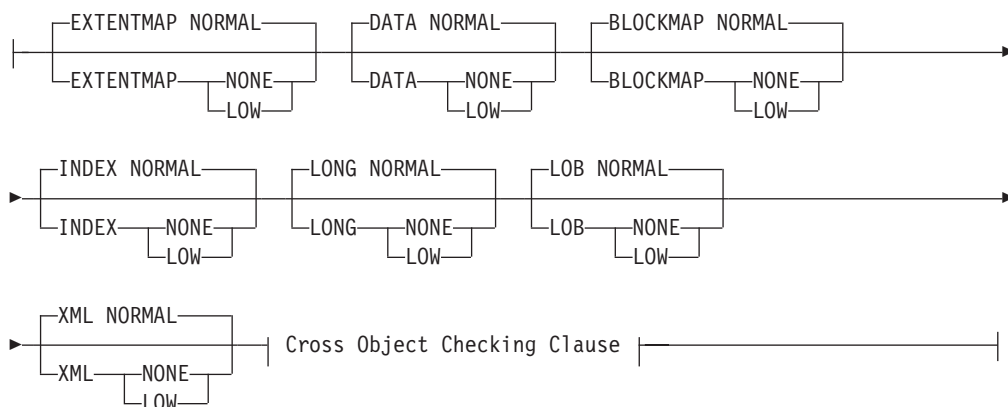
**Check Clause:**



**Row Compression Estimate Clause:**



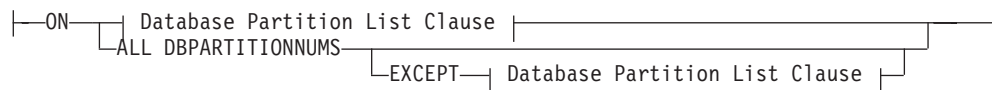
**Level Clause:**



**Cross Object Checking Clause:**

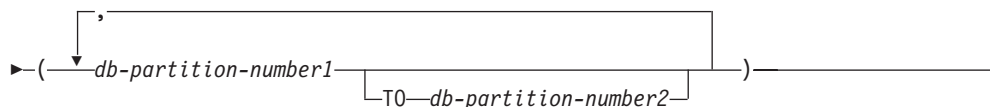


**On Database Partition Clause:**



**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.

### OBJECTID *n*

Specifies processing to begin from table with given table space ID number and object ID number.

### TABLESPACE

#### 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 partitions this operation is to be done on.

This operation will keep the RESULTS output file regardless if the **KEEP** option is specified.

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
- 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 data row compression. For partitioned tables, a separate dictionary is built and inserted on each partition.

When sampling table row data and building a compression dictionary for a table, the **INSPECT** command supports only the table row data in the table object. If the table contains XML columns, data is not sampled and a compression dictionary is not built for the XML data in the XML storage object of the table. Use the table function instead.

The **ROWCOMPESTIMATE** option does not provide an index compression estimate. Use the table function instead.

## 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. The file has to be created in the diagnostic data directory path.

## 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 database partition 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 database partition 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.

When used with the **INDEXDATA** option, as long as the index or data object is in an error state, the online index to data consistency checking will not be performed.

**LIMIT ERROR TO *n***

Number of pages in error for an object to which reporting is limited. 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.

When used with the **INDEXDATA** option, *n* represents the number of errors to which reporting is limited during the online index to data consistency checking.

**LIMIT ERROR TO DEFAULT**

Default number of pages to limit error reporting for an object. This value is the extent size of the object. This parameter is the default.

When used with the **INDEXDATA** option, **DEFAULT** represents the default number of errors to which reporting is limited during the online index to data consistency checking.

**LIMIT ERROR TO ALL**

No limit on number of pages in error reported.

When used with the **INDEXDATA** option, **ALL** represents no limit on the number of errors reported during the online index to data consistency checking.

**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****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.

**INDEXDATA**

Specified in order to perform an index to data consistency check.

**INDEXDATA** checking is not performed by default.

**Examples**

- To perform an index to data consistency check that allows read/write access to all objects, even the object inspected at the moment, issue the following command:

```
inspect check table name fea3 indexdata results keep fea3high.out
```

- To perform an index to data consistency check that allows read or write access to all objects, including the object that is being currently inspected, issue:

```
INSPECT CHECK TABLE NAME car SCHEMA vps INDEXDATA RESULTS KEEP table1.out
```

- To estimate how much storage space will be saved if the data in a table named EMPLOYEE is compressed, issue:

```
INSPECT ROWCOMPESTIMATE TABLE NAME car SCHEMA vps RESULTS table2.out
```

**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** option 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 TABLESPACE** option 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. The exception is when the **INDEXDATA** option is used. **INDEXDATA** will check index to data consistency as long as the index object resides in the table space. This means:
  - If the data object resides in a different table space than the specified table space to be inspected where the index object resides, it can still benefit from the **INDEXDATA** checking.
  - For a partitioned table, each index can reside in a different table space. Only those indexes that reside in the specified table space will benefit from the index to data checking. If you want to inspect all the indexes against one table, use the **CHECK TABLE** option or the **CHECK DATABASE** option.
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.
9. The **INDEXDATA** option only examines the logical inconsistency between index and data. Therefore, it is recommended that you first run **INDEX** and **DATA** checking separately, to rule out any physical corruption, before running **INDEXDATA** checking.
10. The **INSPECT** command, specified with the **INDEXDATA** parameter, performs an index to data consistency check while allowing read/write access to all objects/tables, even the one being inspected at the moment. The **INSPECT INDEXDATA** option includes the following inspections:
  - the existence of the data row for a given index entry.
  - a key to data value verification.

When the **INDEXDATA** option is specified:

- By default, only the values of explicitly specified level clause options will be used. For any level clause options which are not explicitly specified, the default levels will be overwritten from NORMAL to NONE. For instance, when **INDEXDATA** is the only level clause option specified, by default, only index to data checking will be performed.

11. The **BLOCKMAP** option returns information that includes whether a block has been reclaimed for use by the table space following a reorganization to reclaim multidimensional clustering (MDC) or insert time clustering (ITC) table blocks that were empty.

---

### 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 information:

- Database name
- Number of applications currently connected to the database
- Database path.

#### Scope

This command is invoked on the currently attached member and only returns information for that member by default. The currently attached member defaults to the host on which the command is run. If there are multiple members per host, the currently attached member is the first member listed in the `db2nodes.cfg` file on that host.

To run the command on a member that is not the current attach member, specify the **AT MEMBER** option to run this command on a specific member, or specify the **GLOBAL** option to run this command on all members to receive an aggregated result.

To change the current attach member from which to run any subsequent commands, use the **ATTACH\_MEMBER** parameter of the **SET CLIENT** command. You must **DETACH** and then again **ATTACH** your application for this client setting to take effect.

#### Authorization

One of the following authorities:

- SYSADM
- SYSCtrl
- SYSMAINT
- SYSMON

#### Command syntax

```
▶▶—LIST ACTIVE DATABASES—▶▶  
    —AT MEMBER—member-number—  
    —GLOBAL—
```

#### Command parameters

**AT MEMBER** *member-number*

Specifies the member for which the list of active databases is to be displayed.

**GLOBAL**

Returns a list of active databases for all members in a partitioned database environment or a DB2 pureScale environment.



## Examples

The following example 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 previous versions:

- **DBPARTITIONNUM** or **NODE** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.

## 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 is invoked on the currently attached member and only returns information for that member by default. The currently attached member defaults to the host on which the command is run. If there are multiple members per host, the currently attached member is the first member listed in the `db2nodes.cfg` file on that host.

To run the command on a member that is not the current attach member, specify the **AT MEMBER** option to run this command on a specific member, or specify the **GLOBAL** option to run this command on all members to receive an aggregated result.

To change the current attach member from which to run any subsequent commands, use the **ATTACH\_MEMBER** parameter of the **SET CLIENT** command. You must **DETACH** and then again **ATTACH** your application for this client setting to take effect.

### Authorization

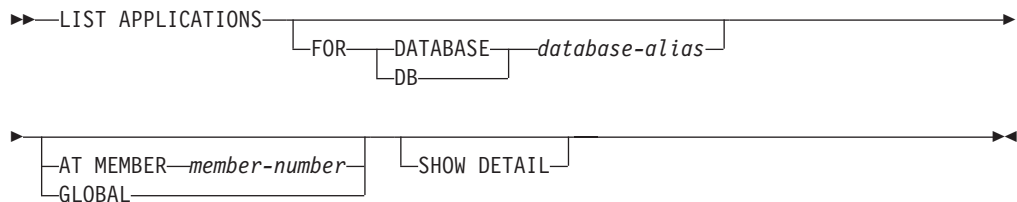
one of the following authorities:

- 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 member to which the user is currently attached.

The default application information consists of the following:

- Authorization ID
- Application name
- Application handle
- Application ID
- Database name
- Number of agents

**AT MEMBER** *member-number*

Specifies the member for which the active applications are to be displayed.

**GLOBAL** Returns a list of active applications for all members in a partitioned database environment or a DB2 pureScale environment.

**SHOW DETAIL**

Some of the additional output information will include:

- CONNECT Auth ID
- Sequence number
- Coordinating member number
- Coordinator pid or thread
- Status
- Status change time
- Node
- 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.

**LIST APPLICATIONS** only shows user applications while **LIST APPLICATIONS SHOW DETAIL** shows all applications including the system applications. Event monitors are an example of system applications. System applications usually appear in snapshot output with application names beginning "db2" (for example, db2stmm, db2taskd).

## LIST APPLICATIONS

### Compatibilities

For compatibility with previous versions:

- **DBPARTITIONNUM** or **NODE** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to **ON**.

## 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

## 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.

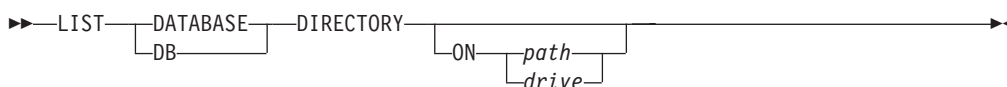
### Authorization

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.

## LIST DATABASE DIRECTORY

- 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

Regardless of the **DB2LDAPCACHE** miscellaneous variable setting, if using the **LIST DATABASE DIRECTORY** or the **LIST NODE DIRECTORY** commands, the list of local database and node entries are read from the LDAP server.

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.



## 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/sqllib/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 authorities is required:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON
- DBADM
- 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

The following example is sample output from the **LIST DATABASE PARTITION GROUPS** command:

```

DATABASE PARTITION GROUP NAME
-----
IBMCGROUP
IBMDEFAULTGROUP

```

2 record(s) selected.

The following example is sample output from the **LIST DATABASE PARTITION GROUPS SHOW DETAIL** command:

## LIST DATABASE PARTITION GROUPS

DATABASE PARTITION GROUP NAME	PMAP_ID	DATABASE PARTITION NUMBER	IN_USE
IBMCATGROUP	0		0 Y
IBMDEFAULTGROUP	1		0 Y

2 record(s) selected.

The fields are identified as follows:

### 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.

---

## LIST DBPARTITIONNUMS

Lists all database partitions or members associated with the current database.

### Scope

This command can be issued from any database partition or member that is listed in `$HOME/sqllib/db2nodes.cfg`. It returns the same information from any of these database partitions or members.

### Authorization

None

### Required connection

Database

### Command syntax

▶—LIST DBPARTITIONNUMS—▶

### Command parameters

None

### Examples

**Example 1 - Partitioned database instance :** Following is sample output from the **LIST DBPARTITIONNUMS** command:

```

DATABASE PARTITION NUMBER
-----
                                0
                                2
                                5
                                7
                                9

```

5 record(s) selected.

**Example 2 - DB2 pureScale instance :** In a DB2 pureScale environment, **LIST DBPARTITIONNUMS** will always report a single partition.

```

DATABASE PARTITION NUMBER
-----
                                0

```

1 record(s) selected.

---

## 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 authorities:

- SYSADM
- SYSCTRL
- SYSMOINT
- 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
- Client protocol
- Client code page
- Process ID of the client application
- Host coded character set ID (CCSID).

**Note:**

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.

### 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 example 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.

**DCS directory release level**

Specifies the version number of the Distributed Database Connection Services<sup>®</sup> 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 or 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 System i hosts
- DB2 databases on VSE & VM hosts

---

## 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

None

### 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.

## LIST HISTORY

Lists entries in the database history records. The database history records contain 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.

The **LIST HISTORY** command only returns history information for the database partition it is issued on. To list the history on multiple partitions, you can either issue the **LIST HISTORY** command from each individual database partition, or use the **db2\_a11** prefix to run the **LIST HISTORY** command on all database partitions

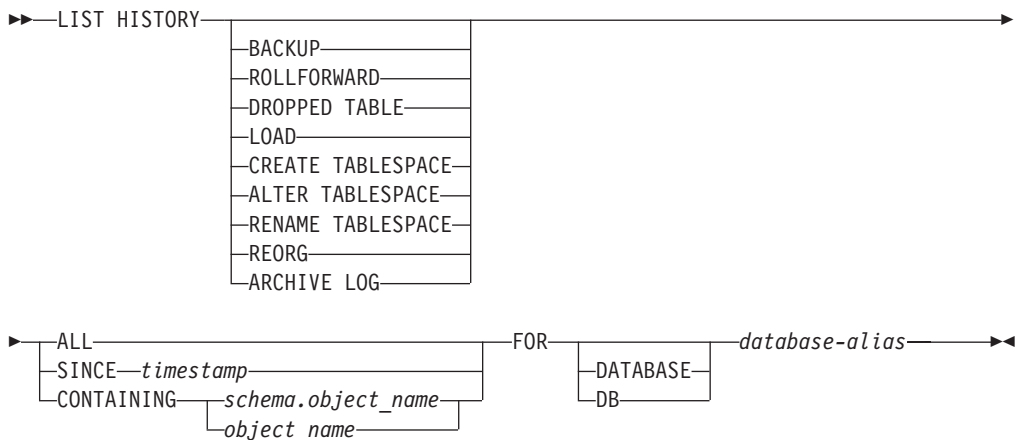
### Authorization

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 database history records.

#### 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.

**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 database history records.

**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 database history records are to be listed.

**Examples**

The following examples show different uses of the **LIST HISTORY** command:

```
db2 list history since 19980201 for sample
db2 list history backup containing userspace1 for sample
db2 list history dropped table all for db sample
```

**Example 1**

The following sample output shows two entries, one for a Load (L) operation and another one for a backup (B) operation:

```
db2 list history all for SAMPLE
```

```

                                List History File for sample

Number of matching file entries = 2

Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-----
L T 20100106133005001 R S S0000000.LOG S0000000.LOG
-----
"USERNAME"."T1" resides in 1 tablespace(s):

00001 USERSPACE1
-----
Comment: DB2
Start Time: 20100106133005
End Time: 20100106133006
Status: A
-----
EID: 3 Location: /home/hotel19/username/mydatafile.de1
-----
```

## LIST HISTORY

### SQLCA Information

sqlcaid : SQLCA      sqlcabc: 136      sqlcode: 3107      sqlerrml: 0

sqlerrmc:

sqlerrp : SQLUVLD

sqlerrd : (1) -2146107283      (2) 0      (3) 0  
(4) 0      (5) 0      (6) 0

sqlwarn : (1) W      (2)      (3)      (4)      (5)      (6)  
(7)      (8)      (9)      (10)      (11)

sqlstate:

```
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-----
B D 20100106135509001 F D S0000000.LOG S0000000.LOG
```

Contains 2 tablespace(s):

00001 SYSCATSPACE  
00002 USERSPACE1

Comment: DB2 BACKUP SAMPLE OFFLINE  
Start Time: 20100106135509  
End Time: 20100106135512  
Status: A

EID: 4 Location: /home/hotel19/username

### Example 2

The following sample output shows one entry for the reorganization reclaim operation:

db2 -v "list history reorg all for wsdb"

```
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-----
G T 20080924101408 N S0000000.LOG S0000000.LOG
```

Table: "ZHMFENG"."T1"

Comment: REORG RECLAIM  
Start Time: 20080924101408  
End Time: 20080924101409  
Status: A

### Example 3

Use the **db2\_all** prefix to run the LIST HISTORY command on all database partitions:

db2\_all "db2 list history since 20010601 for sample"

### Example 4

The following is an example of DB history records in a DB2 pureScale environment.

db2 list history since 20091020163200 for database sample

```
Op Obj Timestamp+Sequence Type Dev Backup ID
-----
X D 20091020163218 1 D
```

```
Log Stream ID Earliest Log Current Log
-----
3 S0000023.LOG C0000000
```

Comment:  
Start Time: 20091020163218  
End Time: 20091020163245  
Status: A

EID: 28 Location: /notnfs/billings/arch\_logs/billings/SAMPLE/NODE0000/LOGSTREAM0002/C0000000/S0000023.LOG

```
Op Obj Timestamp+Sequence Type Dev Backup ID
-----
X D 20091020163219 1 D
```

```

-----
Log Stream ID Earliest Log Current Log
-----
0 S0000001.LOG C0000000
-----
Comment:
Start Time: 20091020163219
End Time: 20091020163257
Status: A
-----
EID: 29 Location: /notnfs/billings/arch_logs/billings/SAMPLE/NODE0000/LOGSTREAM0000/C0000000/S0000001.LOG

```

### Example 5

The following is an example of DB history records outside of a DB2 pureScale environment.

db2 list history since 20091020155300 for database sample

```

Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-----
X D 20091020155341 1 D S0000004.LOG C0000000
-----
Comment:
Start Time: 20091020155341
End Time: 20091020155345
Status: A
-----
EID: 9 Location: /notnfs/billings/arch_logs/billings/SAMPLE/NODE0000/LOGSTREAM0000/C0000000/S0000004.LOG

```

### Usage notes

The SYSIBMADM.DB\_HISTORY administrative view can be used to retrieve data from all database partitions.

In a DB2 pureScale instance, all DB history records for the database are global. The DB history records can be retrieved using the list history or administrative view interface connected to any member.

The report generated by this command contains the following symbols:

#### Operation

- A - Create table space
- B - Backup
- C - Load copy
- D - Drop table
- F - Rollforward
- G - Reorganize
- L - Load
- N - Rename table space
- O - Drop table space
- Q - Quiesce
- R - Restore
- T - Alter table space
- U - Unload
- X - Archive log

#### Object

- D - Database
- I - Index
- P - Table space
- T - Table
- R - Partitioned table

#### Type

Alter table space operation types:

- C - Add container
- R - Rebalance

## LIST HISTORY

### Archive log operation types:

- F - Failover archive path
- M - Secondary (mirror) log path
- N - Archive log command
- P - Primary log path
- 1 - Primary log archive method
- 2 - Secondary log archive method

### Backup and restore operation types:

- D - Delta offline
- E - Delta online
- F - Offline
- I - Incremental offline
- M - Merged
- N - Online
- O - Incremental online
- R - Rebuild

### Load operation types:

- I - Insert
- R - Replace

### Rollforward operation types:

- E - End of logs
- P - Point-in-time

### Quiesce operation types:

- S - Quiesce share
- U - Quiesce update
- X - Quiesce exclusive
- Z - Quiesce reset

### History entry status flag:

- A - Active
- D - Deleted
- E - Expired
- I - Inactive
- N - Not yet committed
- P - Pending delete
- X - Do not delete
- a - Incomplete active
- i - Incomplete inactive

---

## 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 1)
- List indoubt transaction number *x* (enter 1, 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 about the 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.



## 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.toro1ab.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

Regardless of the **DB2LDAPCACHE** miscellaneous variable setting, if using the **LIST DATABASE DIRECTORY** or the **LIST NODE DIRECTORY** commands, the list of local database and node entries are read from the LDAP server.

A node directory is created and maintained on each IBM Data Server Runtime 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

---

## 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.

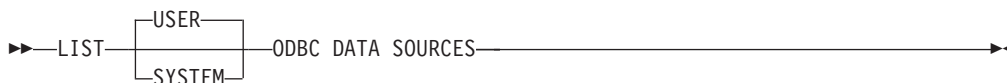
### Authorization

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 example is sample output from the **LIST ODBC DATA SOURCES** command:

```
                User ODBC Data Sources  
  
Data source name      Description  
-----  
SAMPLE                IBM DB2 ODBC DRIVER
```

## LIST PACKAGES/TABLES

Lists packages or tables associated with the current database.

### Authorization

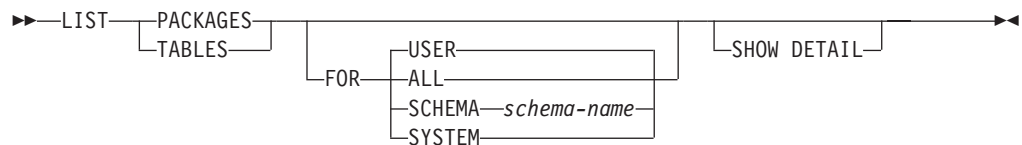
For the system catalog SYSCAT.PACKAGES (**LIST PACKAGES**) and SYSCAT.TABLES (**LIST TABLES**), one of the following is required:

- SYSADM
- SYSCtrl
- SYSMAINT
- SYSMON
- DBADM
- 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** *schema-name*

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 bound by 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

## LIST PACKAGES/TABLES

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.

### 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;
```

## LIST PACKAGES/TABLES

```
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;
```

---

# LIST TABLESPACE CONTAINERS

Lists containers for the specified table space.

**Important:** This command or API has been deprecated in Version 9.7 and might be removed in a future release. You can use the `MON_GET_TABLESPACE` and the `MON_GET_CONTAINER` table functions instead which return more information. For more information, see “LIST TABLESPACES and LIST TABLESPACE CONTAINERS commands have been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055001.html>.

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 authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON
- 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 usable pages
- Accessible (yes or no).

### Examples

The following is sample output from **LIST TABLESPACE CONTAINERS FOR 0**:

Tablespace Containers for Tablespace 0

Container ID	= 0
Name	= /home/smith/smith/NODE0000/SQL00001/SQLT0000.0
Type	= Path

The following is sample output from **LIST TABLESPACE CONTAINERS FOR 0 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

---

# LIST TABLESPACES

Lists table spaces and information about table spaces for the current database.

**Important:** This command or API has been deprecated in Version 9.7 and might be removed in a future release. You can use the `MON_GET_TABLESPACE` and the `MON_GET_CONTAINER` table functions instead which return more information. For more information, see “LIST TABLESPACES and LIST TABLESPACE CONTAINERS commands have been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055001.html>.

Information displayed by this command is also available in the table space snapshot.

### Scope

This command returns information only for the database partition on which it is executed.

### Authorization

one of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON
- DBADM
- LOAD authority

### 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
0x10000	Suspend Write
0x20000	Load in progress
0x2000000	Storage may be defined
0x4000000	StorDef is in 'final' state
0x8000000	StorDef was change before roll forward
0x10000000	DMS rebalance in progress
0x20000000	Table space deletion in progress
0x40000000	Table space creation in progress

**Note:** DB2 LOAD does not set the table space state to Load pending or Delete pending.

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 (earliest point in time to which a table space can be rolled forward; timestamp expressed in UTC 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).

## LIST TABLESPACES

### 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

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
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

```

## LIST TABLESPACES

```
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.

When the **LIST TABLESPACES SHOW DETAIL** command is issued, it will attempt to free all pending free extents in the table space. If the pending free extents are freed successfully, a record will be logged.

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.

## 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 returns information for all database partitions.

### Authorization

One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON

### 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

```

## LIST UTILITIES

```
Description          =  
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.

The **LIST UTILITIES** command can be used to monitor the progress of deferred cleanup of indexes by asynchronous index cleanup.

Starting with DB2 Version 9.7 Fix Pack 1, the **LIST UTILITIES** command can be used to monitor the progress of the completion of a detach of a data partition from a partitioned table by the asynchronous partition detach task. Detaching a data partition from a data partitioned table is initiated by issuing a **ALTER TABLE** statement with the **DETACH PARTITION** clause.



---

## LOAD

Loads data into a DB2 table.

Data stored on the server can be in the form of a file, tape, or named pipe. Data stored 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 is subject to compression on every data and database partition for which a dictionary exists in the table, including data in the XML storage object of the table.

Quick link to “File type modifiers for the load utility” on page 416.

### Restrictions

The load utility does not support loading data at the hierarchy level. The load utility is not compatible with range-clustered tables. The load utility does not support the NOT LOGGED INITIALLY parameter for the CREATE TABLE or ALTER TABLE statements.

### Scope

This command can be issued against multiple database partitions in a single request.

### Authorization

One of the following authorities:

- DATAACCESS
- LOAD authority on the database and the following privileges:
  - 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 directly or indirectly through a group or a role 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:
  - The security label is part of the security policy protecting the table.
  - The security label was granted to the session authorization ID directly or indirectly through a group or a role 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. The security label protects a loaded row if the session authorization ID 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

# LOAD

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.

When you load data into a table with protected rows, the target table has one column with a data type of DB2SECURITYLABEL. If the input row of data does not contain a value for that column, that row is rejected unless the usedefaults file type modifier is specified in the load command, in which case the security label you hold for write access from the security policy protecting the table is used. If you do not hold a security label for write access, the row is rejected and processing continues on to the next row

- If the REPLACE option is specified, the session authorization ID must have the authority to drop the table.
- If the LOCK WITH FORCE option is specified, SYSADM authority is required.
- If the table has row access control activated, then **LOAD REPLACE** on that table would require the ability to drop the table. Specifically, you must have either CONTROL or DBADM on the table.

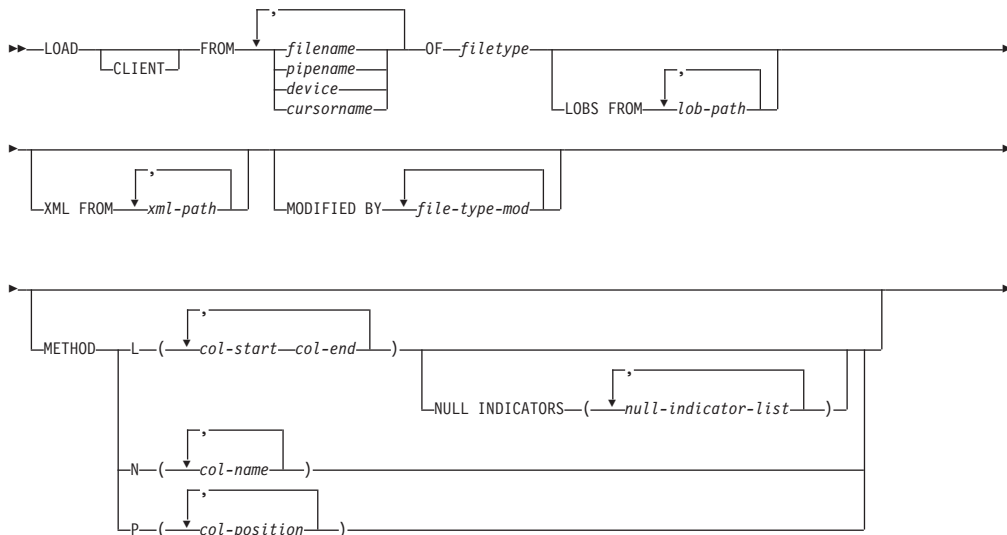
Since all load processes (and all DB2 server processes, in general) are owned by the instance owner, and all of these processes use the identification of the instance owner to access needed files, the instance owner must have read access to input data files. These input data files must be readable by the instance owner, regardless of who invokes the command.

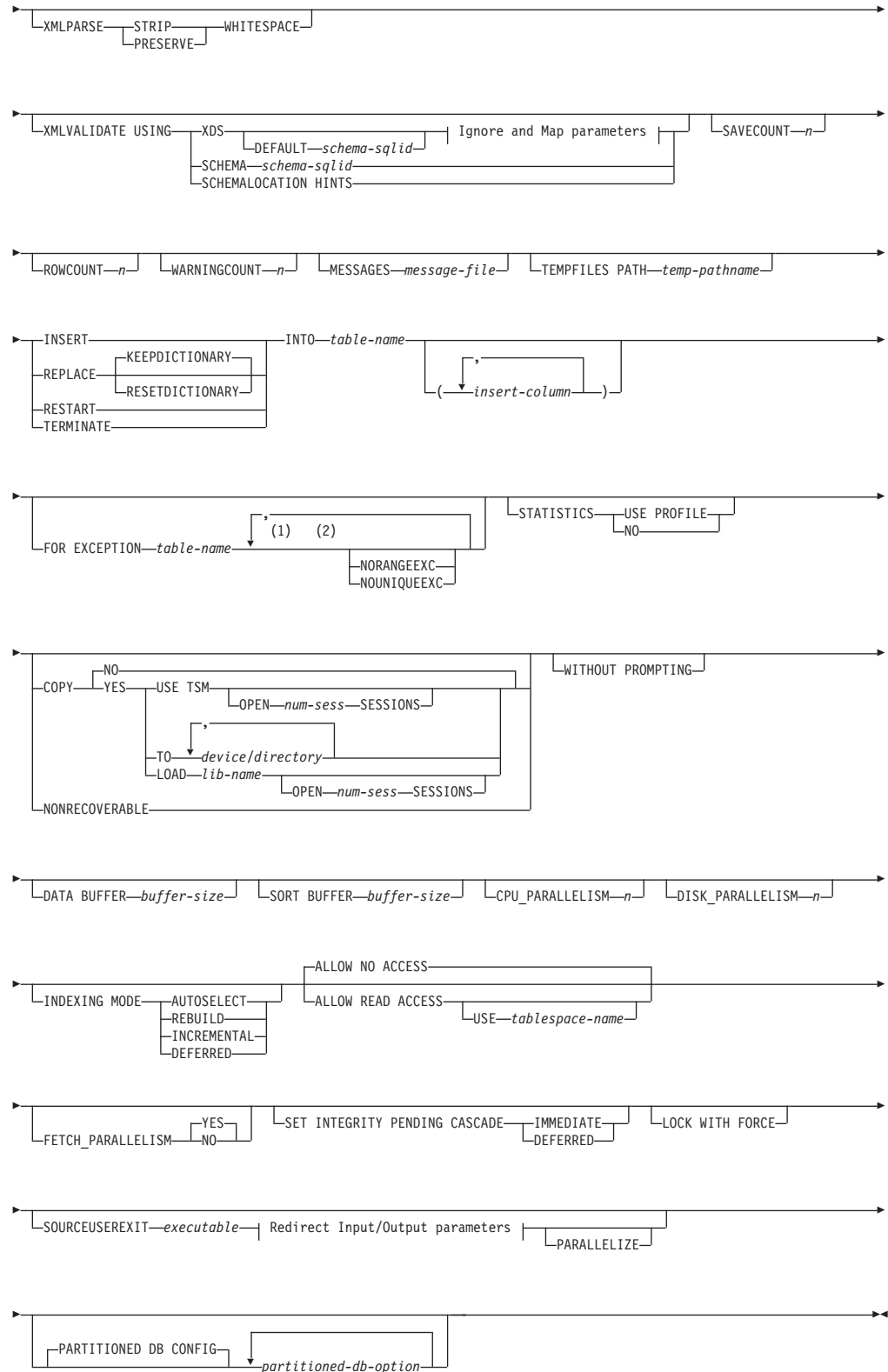
## Required connection

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.

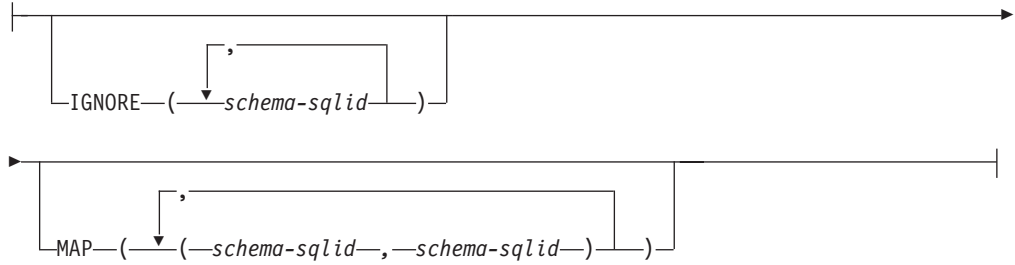
## Command syntax



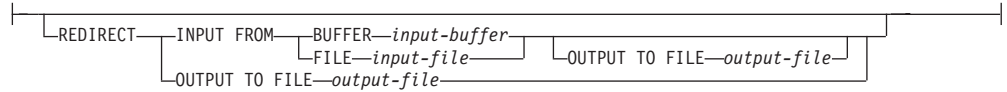


Ignore and Map parameters:

## LOAD



### Redirect Input/Output parameters:



### 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` file type.

#### Note:

1. The `dumpfile` and `lobsinfile` modifiers and the `XML FROM` option 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* | *pipename* | *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.

**Note:**

- Use the fully qualified file name. 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.
- 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** 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.
- 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.)
- 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.
- When loading XML data from files into tables in a partitioned database environment, the XML data files must be read-accessible to all the database partitions where loading is taking place.

**OF** *filetype*

Specifies the format of the data:

- ASC (non-delimited ASCII format)
- DEL (delimited ASCII format)
- IXF (Integration Exchange Format, PC version) is a binary format that is used exclusively by DB2 databases.
- CURSOR (a cursor declared against a SELECT or VALUES statement).

**Note:** When using a CURSOR file type to load XML data into a table in a distributed database environment, the PARTITION\_ONLY and LOAD\_ONLY modes are not supported.

**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** behavior.

This option is ignored when specified in conjunction with the CURSOR file type.

**MODIFIED BY** *file-type-mod*

Specifies file type modifier options. See “File type modifiers for the load utility” on page 416.

**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.

**XML FROM** *xml-path*

Specifies one or more paths that contain the XML files. XDSs are contained in the main data file (ASC, DEL, or IXF), in the column that will be loaded into the XML column.

**XMLPARSE**

Specifies how XML documents are parsed. If this option is not specified, the parsing behavior 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 in the following section.

**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 a loaded 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 loaded XML document.

**Note:**

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.

**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 a loaded 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.

## **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 loaded 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.

## **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 not allowed when specified in conjunction with the **CURSOR** file type or when loading a table containing an XML column.

## **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 to standard output unless the instance is configured as a partitioned database environment. 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. The following list 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
- 15 KB 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.

This option cannot be used to load data into system-period temporal tables.

**KEEPDICTIONARY**

An existing compression dictionary is preserved across the **LOAD REPLACE** operation. Provided the table COMPRESS attribute is YES, the newly replaced data is subject to being compressed using the dictionary that existed before the invocation of the load. If no dictionary previously existed in the table, a new dictionary is built using the data that is being replaced into the table as long as the table COMPRESS attribute is YES. The amount of data that is required to build the compression dictionary in this case is subject to the policies of ADC. This data is populated into the table as uncompressed. Once the dictionary is inserted into the table, the remaining data to be loaded is subject to being compressed with this dictionary. This is the default parameter. For summary, see Table 1.

Table 28. LOAD REPLACE KEEPDICTIONARY

Compress	Table row data dictionary exists	XML storage object dictionary exists <sup>1</sup>	Compression dictionary	Data compression
YES	YES	YES	Preserve table row data and XML dictionaries.	Data to be loaded is subject to compression.
YES	YES	NO	Preserve table row data dictionary and build a new XML dictionary.	Table row data to be loaded is subject to compression. After XML dictionary is built, remaining XML data to be loaded is subject to compression.

## LOAD

Table 28. LOAD REPLACE KEEPDICTIONARY (continued)

Compress	Table row data dictionary exists	XML storage object dictionary exists <sup>1</sup>	Compression dictionary	Data compression
YES	NO	YES	Build table row data dictionary and preserve XML dictionary.	After table row data dictionary is built, remaining table row data to be loaded is subject to compression. XML data to be loaded is subject to compression.
YES	NO	NO	Build new table row data and XML dictionaries.	After dictionaries are built, remaining data to be loaded is subject to compression.
NO	YES	YES	Preserve table row data and XML dictionaries.	Data to be loaded is not compressed.
NO	YES	NO	Preserve table row data dictionary.	Data to be loaded is not compressed.
NO	NO	YES	No effect on table row dictionary. Preserve XML dictionary.	Data to be loaded is not compressed.
NO	NO	NO	No effect.	Data to be loaded is not compressed.

### Note:

1. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in DB2 Version 9.7 or later, or if the table is migrated using an online table move.
2. If **LOAD REPLACE KEEPDICTIONARY** operation is interrupted, load utility can recover after either **LOAD RESTART** or **LOAD TERMINATE** is issued. Existing XML storage object dictionary may not be preserved after recovery from interrupted **LOAD REPLACE KEEPDICTIONARY** operation. A new XML storage object dictionary will be created if **LOAD RESTART** is used

### RESETDICTIONARY

This directive instructs **LOAD REPLACE** processing to build a new dictionary for the table data object provided that the table COMPRESS attribute is YES. If the COMPRESS attribute is NO and a dictionary was already present in the table it will be removed and no new dictionary will be inserted into the table. A compression dictionary can be built with just one user record. If the loaded data set size is zero and if there is a preexisting dictionary, the dictionary will not be preserved. The amount of data required to build a dictionary with this directive is not subject to the policies of ADC. For summary, see Table 2.

Table 29. LOAD REPLACE RESETDICTIONARY

Compress	Table row data dictionary exists	XML storage object dictionary exists <sup>1</sup>	Compression dictionary	Data compression
YES	YES	YES	Build new dictionaries <sup>2</sup> . If the DATA CAPTURE CHANGES option is enabled on the CREATE TABLE or ALTER TABLE statements, the current table row data dictionary is kept (and referred to as the <i>historical compression dictionary</i> ).	After dictionaries are built, remaining data to be loaded is subject to compression.
YES	YES	NO	Build new dictionaries <sup>2</sup> . If the DATA CAPTURE CHANGES option is enabled on the CREATE TABLE or ALTER TABLE statements, the current table row data dictionary is kept (and referred to as the <i>historical compression dictionary</i> ).	After dictionaries are built, remaining data to be loaded is subject to compression.
YES	NO	YES	Build new dictionaries.	After dictionaries are built, remaining data to be loaded is subject to compression.
YES	NO	NO	Build new dictionaries.	After dictionaries are built, remaining data to be loaded is subject to compression.
NO	YES	YES	Remove dictionaries.	Data to be loaded is not compressed.
NO	YES	NO	Remove table row data dictionary.	Data to be loaded is not compressed.
NO	NO	YES	Remove XML storage object dictionary.	Data to be loaded is not compressed.
NO	NO	NO	No effect.	All table data is not compressed.

**Notes:**

1. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in DB2 Version 9.7 or later, or if the table is migrated using an online table move.
2. If a dictionary exists and the compression attribute is enabled, but there are no records to load into the table partition, a new dictionary cannot be built and the **RESETDICTIONARY** operation will not keep the existing dictionary.

**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

## LOAD

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. For summary of dictionary management, see Table 3.

The **LOAD TERMINATE** option will not remove a backup pending state from table spaces.

### 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. For summary of dictionary management, see Table 4.

### INTO *table-name*

Specifies the database table into which the data is to be loaded. This table cannot be a system table, a declared temporary table, or a created 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.

If the database table contains implicitly hidden columns, you must specify whether data for the hidden columns is included in the load operation. Use one of the following methods to indicate if data for hidden columns is included:

- Use *insert-column* to explicitly specify the columns into which data is to be inserted.  

```
db2 load from delfile1 of del
      insert into table1 (c1, c2, c3,...)
```
- Use one of the hidden column file type modifiers: specify **implicitlyhiddeninclude** when the input file contains data for the hidden columns, or **implicitlyhiddenmissing** when the input file does not.  

```
db2 load from delfile1 of del modified by implicitlyhiddeninclude
      insert into table1
```
- Use the DB2\_DMU\_DEFAULT registry variable on the server-side to set the default behavior when data movement utilities encounter tables with implicitly hidden columns. Specify **IMPLICITLYHIDDENINCLUDE** when utilities assume that the implicitly hidden columns are included, or **IMPLICITLYHIDDENMISSING** when utilities assume that the implicitly hidden columns are not included.  

```
db2set DB2_DMU_DEFAULT=IMPLICITLYHIDDENINCLUDE
db2 load from delfile1 of del insert into table1
```

### *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, otherwise, contains rows that cannot be loaded because they are invalid or have syntax errors.

When loading XML data, using the **FOR EXCEPTION** clause to specify a load exception table is not supported in the following cases:

- When using label-based access control (LBAC).
- When loading data into a partitioned table.

**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.

During load, distribution statistics are not collected for columns of type XML.

**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, if either **logarchmeth1** or **logarchmeth2** is set to a value other than OFF). 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** 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

## LOAD

automatically by the database when the index is accessed by a query. If one of **COPY NO**, **COPY YES**, or **NONRECOVERABLE** is not specified, and the database is recoverable (**logarchmeth1** or **logarchmeth2** is set to value other than OFF), then **COPY NO** is the default.

### **COPY YES**

Specifies that a copy of the loaded data will be saved. This option is invalid if forward recovery is disabled.

### **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 unrecoverable 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. If one of **COPY NO**, **COPY YES**, or **NONRECOVERABLE** is not specified, and the database is not recoverable (**logarchmeth1** and **logarchmeth2** are both set to OFF), then **NONRECOVERABLE** is the default.

### **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 4 KB 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. Beginning in version 9.5, the value of the DATA BUFFER option of the **LOAD** command can temporarily exceed **util\_heap\_sz** if more memory is available in the system. In this situation, the utility heap is dynamically increased as needed until the **database\_memory** limit is reached. This memory will be released once the load operation completes.

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 create for parsing, converting, and formatting records when building table objects. This parameter is designed to exploit the number of processes running per database partition. 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.

#### **Note:**

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 create 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.

#### **INDEXING MODE**

Specifies whether the load utility is to rebuild indexes or to extend them incrementally. Valid values are:

##### **AUTOSELECT**

The load utility will automatically decide between REBUILD or INCREMENTAL mode. The decision is based on the amount of

## LOAD

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.

### 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.

**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”.

**SET INTEGRITY PENDING CASCADE**

If **LOAD** puts the table into Set Integrity Pending state, the **SET 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 descendents (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

## LOAD

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 query of a table that is in the Set Integrity Pending state might succeed if an eligible materialized query table that is not in the Set Integrity Pending state is accessed by the query instead of the specified table. A warning (SQLSTATE 01586) will be issued to indicate that descendent 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** *partitioned-db-option*

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 options:

```

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
NEWLINE
DISTFILE x
OMIT_HEADER
RUN_STAT_DBPARTNUM x

```

Detailed descriptions of these options are provided in “Load configuration options for partitioned database environments”.

## LOAD

### RESTARTCOUNT

Deprecated.

### USING *directory*

Deprecated.

## 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 7 elements:

- ELE1 positions 01 to 20
- ELE2 positions 21 to 22
- ELE3 positions 23 to 23
- ELE4 positions 24 to 27
- ELE5 positions 28 to 31
- ELE6 positions 32 to 32
- ELE7 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)
```

### Note:

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).
2. The specification of **reclen=40** in the **MODIFIED BY** parameter indicates that there is no newline character at the end of each input record, and that each record is 40 bytes long. The last 8 bytes are not used to load the table.
3. Since COL4 is not provided in the input file, it will be inserted into TABLE1 with its default value (it is defined NOT NULL WITH DEFAULT).
4. Positions 23 and 32 are used to indicate whether COL2 and COL3 of TABLE1 will be loaded NULL for a given row. If there is a Y in the column's null indicator position for a given record, the column will be NULL. If there is an N, the data values in the column's data positions of the input record (as defined in

L(.....)) are used as the source of column data for the row. In this example, neither column in row 1 is NULL; COL2 in row 2 is NULL; and COL3 in row 3 is NULL.

5. In this example, the NULL INDICATORS for COL1 and COL5 are specified as 0 (zero), indicating that the data is not nullable.
6. The NULL INDICATOR for a given column can be anywhere in the input record, but the position must be specified, and the Y or N values must be supplied.

### Example 2 (Loading LOBs from Files)

TABLE1 has 3 columns:

- COL1 CHAR 4 NOT NULL WITH DEFAULT
- LOB1 LOB
- LOB2 LOB

ASCFILE1 has 3 elements:

- ELE1 positions 01 to 04
- ELE2 positions 06 to 13
- ELE3 positions 15 to 22

The following files reside in either /u/user1 or /u/user1/bin:

- ASCFILE2 has LOB data
- ASCFILE3 has LOB data
- ASCFILE4 has LOB data
- ASCFILE5 has LOB data
- ASCFILE6 has LOB data
- ASCFILE7 has LOB data

Data Records in ASCFILE1:

```
1...5...10...15...20...25...30.
REC1 ASCFILE2 ASCFILE3
REC2 ASCFILE4 ASCFILE5
REC3 ASCFILE6 ASCFILE7
```

The following command loads the table from the file:

```
db2 load from ascfile1 of asc
      lobs from /u/user1, /u/user1/bin
      modified by lobsinfile reclen=22
      method L (1 4, 6 13, 15 22)
      insert into table1
```

### Note:

1. The specification of **lobsinfile** in the **MODIFIED BY** parameter tells the loader that all LOB data is to be loaded from files.
2. The specification of **reclen=22** in the **MODIFIED BY** parameter indicates that there is no newline character at the end of each input record, and that each record is 22 bytes long.
3. LOB data is contained in 6 files, ASCFILE2 through ASCFILE7. Each file contains the data that will be used to load a LOB column for a specific row. The relationship between LOBs and other data is specified in ASCFILE1. The first record of this file tells the loader to place REC1 in COL1 of row 1. The contents of ASCFILE2 will be used to load LOB1 of row 1, and the contents of

## LOAD

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 nondelimited ASCII file), without the **lobsinfile** modifier, the following rules must be observed:
  - The total length of any record, including LOBs, cannot exceed 32 KB.
  - 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
```

**Note:**

1. The following command generates identity values for rows 1 and 2, since no identity values are supplied in DATAFILE1 for those rows. However, row 3 is assigned the user-supplied identity value of 100 and row 4 is assigned the the user-supplied identity value of 101.

```
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)
```

```
db2 load 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 a Table with a Row Change Timestamp Column)**

TABLE1 has 4 columns:

- C1 VARCHAR(30)
- C2 ROW CHANGE TIMESTAMP GENERATED BY DEFAULT
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 is the same as TABLE1, except that C2 is a GENERATED ALWAYS column.

Data records in DATAFILE1 (DEL format):

## LOAD

```
"Liszt"  
"Hummel",,187.43, H  
"Grieg", 2006-05-23-15.55.53.209971, 66.34, G  
"Satie", 2006-05-22-19.34.14.947681, 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
```

### Note:

1. The following command generates ROW CHANGE TIMESTAMP values for rows 1 and 2, since no ROW CHANGE TIMESTAMP values are supplied in DATAFILE1 for those rows. Rows 3 and 4, however, are assigned the user-supplied ROW CHANGE TIMESTAMP values of 2006-05-23-15.55.53.209971 and 2006-05-22-19.34.14.947681.

```
db2 load from datafile1.del of del replace  
into table1
```

2. To load DATAFILE1 into TABLE1 so that ROW CHANGE TIMESTAMP 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)
```

```
db2 load from datafile1.del of del modified by rowchangetimestampignore  
replace into table1
```

3. To load DATAFILE2 into TABLE1 so that ROW CHANGE TIMESTAMP 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 rowchangetimestampmissing  
replace into table1
```

4. To load DATAFILE1 into TABLE2 so that the ROW CHANGE TIMESTAMP values of 2006-05-23-15.55.53.209971 and 2006-05-22-19.34.14.947681 are assigned to rows 3 and 4, issue the following command:

```
db2 load from datafile1.del of del modified by rowchangetimestampoverride  
replace into table2
```

In this case, rows 1 and 2 will be rejected, because the utility has been instructed to override system-generated ROW CHANGE TIMESTAMP values in favor of user-supplied values. If user-supplied values are not present, however, the row must be rejected, because row change timestamp columns are implicitly not NULL.

5. If DATAFILE1 is loaded into TABLE2 without using any of the ROW CHANGE 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 row change timestamp column is GENERATED ALWAYS.

### Example 6 (Loading using the CURSOR file type)

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
```

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 statement 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, 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
```

### Example : Loading XML data

The user has constructed a data file with XDS fields to describe the documents that are to be inserted into the table. It might appear like this :

```
1, "<XDS FIL=""file1.xml"" />"
2, "<XDS FIL='file2.xml' OFF='23' LEN='45' />"
```

For the first row, the XML document is identified by the file named file1.xml. Note that since the character delimiter is the double quote character, and double quotation marks exist inside the XDS, the double quotation marks contained within the XDS are doubled. For the second row, the XML document is identified by the file named file2.xml, and starts at byte offset 23, and is 45 bytes in length.

The user issues a load command without any parsing or validation options for the XML column, and the data is loaded successfully:

```
LOAD
FROM data.del of DEL INSERT INTO mytable
```

### Example : Loading XML data from CURSOR

Loading data from cursor is the same as with a regular relational column type. The user has two tables, T1 and T2, each of which consist of a single XML column named C1. To LOAD from T1 into T2, the user will first declare a cursor:

```
DECLARE
X1 CURSOR FOR SELECT C1 FROM T1;
```

Next, the user may issue a **LOAD** using the cursor type:

```
LOAD FROM X1 of
CURSOR INSERT INTO T2
```

Applying the XML specific **LOAD** options to the cursor type is the same as loading from a file.

### Examples of using the XMLVALIDATE clause

#### XMLVALIDATE USING XDS

##### Example 1

## LOAD

### USING XDS DEFAULT *schema-sqlid*

The user would like to validate according to the schema indicated in the XDS. If there are any XDS values without SCH attributes, these documents will be validated against the schema that is used in this clause.

```
XMLVALIDATE  
USING XDS DEFAULT S1.SCHEMA_B
```

### Example 2

The user would like to validate but IGNORE certain schemas that are mentioned in the XDS.

```
XMLVALIDATE USING XDS IGNORE S1.SCHEMA_C
```

### Example 3

The user would like to validate but remap some of the schemas in the XDS.

```
XMLVALIDATE USING XDS MAP( (S1.SCHEMA_A,  
S2.SCHEMA_B ), (S3.SCHEMA_C, S5.SCHEMA_E) )
```

Given the previously mentioned XDS, any document with an SCH attribute of S1.SCHEMA\_A will be validated against S2.SCHEMA\_B. Also, any document with an SCH attribute of S3.SCHEMA\_C will be validated against S5.SCHEMA\_E.

### Example 4

The user would like to use a combination of the **DEFAULT**, **IGNORE**, and **MAP** options:

```
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.

### XMLVALIDATE USING SCHEMA

The user wants to validate all XML documents according to a single 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.

### XMLVALIDATE USING SCHEMALOCATION HINTS

The user would like to validate against schema information located within the document itself. For the following **XMLVALIDATE** clause:

```
XMLVALIDATE  
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

- 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. If preservation of the source data order is not required, consider using the **ANYORDER** file type modifier, described in the following "File type modifiers for the load utility" section.

## LOAD

- 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 before loading. Data does not need to be sorted before 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 file type where the DATABASE keyword was specified during the DECLARE CURSOR statement, 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 statement). 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 statement.
- Loading a multiple-part PC/IXF file whose individual parts are copied from a Windows system to an AIX system is supported. The names of all the files must be specified in the **LOAD** command. For example, LOAD FROM DATA.IXF, DATA.002 OF IXF INSERT INTO TABLE1. Loading to the Windows operating system from logically split PC/IXF files is not supported.
- When restarting a failed **LOAD**, the behavior will follow the existing behavior in that the BUILD phase will be forced to use the REBUILD mode for indexes.
- Loading XML documents between databases is not supported and returns error message SQL1407N.
- The **LOAD** utility does not support loading into tables that contain columns that reference fenced procedures. If you issue the **LOAD** command on such table, you will receive error message SQL1376N. To work around this restriction, you can redefine the routine to be unfenced, or use the import utility.
- If the database table contains implicitly hidden columns, you must specify whether data for the hidden columns is included in the load operation.

### Summary of LOAD TERMINATE and LOAD RESTART dictionary management

The following chart summarizes the compression dictionary management behavior for **LOAD** processing under the **TERMINATE** directive.

Table 30. LOAD TERMINATE dictionary management

Table COMPRESS attribute	Does table row data dictionary exist before LOAD?	XML storage object dictionary exists before LOAD <sup>1</sup>	TERMINATE: LOAD REPLACE KEEPDICTIONARY or LOAD INSERT	TERMINATE: LOAD REPLACE RESETDICTIONARY
YES	YES	YES	Keep existing dictionaries.	Neither dictionary is kept. <sup>2</sup>
YES	YES	NO	Keep existing dictionary.	Nothing is kept. <sup>2</sup>
YES	NO	YES	Keep existing dictionary.	Nothing is kept.
YES	NO	NO	Nothing is kept.	Nothing is kept.
NO	YES	YES	Keep existing dictionaries.	Nothing is kept.
NO	YES	NO	Keep existing dictionary.	Nothing is kept.
NO	NO	YES	Keep existing dictionary.	Nothing is kept.
NO	NO	NO	Do nothing.	Do nothing.

**Note:**

1. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in DB2 Version 9.7 or later, or if the table is migrated using an online table move.
2. In the special case that the table has data capture enabled, the table row data dictionary is kept.

**LOAD RESTART** truncates a table up to the last consistency point reached. As part of **LOAD RESTART** processing, a compression dictionary will exist in the table if it was present in the table at the time the last **LOAD** consistency point was taken. In that case, **LOAD RESTART** will not create a new dictionary. For a summary of the possible conditions, see Table 4.

Table 31. LOAD RESTART dictionary management

Table COMPRESS Attribute	Table row data dictionary exist before LOAD consistency point? <sup>1</sup>	XML Storage object dictionary existed before last LOAD? <sup>2</sup>	RESTART: LOAD REPLACE KEEPDICTIONARY or LOAD INSERT	RESTART: LOAD REPLACE RESETDICTIONARY
YES	YES	YES	Keep existing dictionaries.	Keep existing dictionaries.
YES	YES	NO	Keep existing table row data dictionary and build XML dictionary subject to ADC.	Keep existing table row data dictionary and build XML dictionary.
YES	NO	YES	Build table row data dictionary subject to ADC. Keep existing XML dictionary.	Build table row data dictionary. Keep existing XML dictionary.
YES	NO	NO	Build table row data and XML dictionaries subject to ADC.	Build table row data and XML dictionaries.
NO	YES	YES	Keep existing dictionaries.	Remove existing dictionaries.
NO	YES	NO	Keep existing table row data dictionary.	Remove existing table row data dictionary.

## LOAD

Table 31. LOAD RESTART dictionary management (continued)

Table COMPRESS Attribute	Table row data dictionary exist before LOAD consistency point? <sup>1</sup>	XML Storage object dictionary existed before last LOAD? <sup>2</sup>	RESTART: LOAD REPLACE KEEPDICTIONARY or LOAD INSERT	RESTART: LOAD REPLACE RESETDICTIONARY
NO	NO	YES	Keep existing XML dictionary.	Remove existing XML dictionary.
NO	NO	NO	Do nothing.	Do nothing.

### Notes:

1. The **SAVECOUNT** option is not allowed when loading XML data, load operations that fail during the load phase restart from the beginning of the operation.
2. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in DB2 Version 9.7 or later, or if the table is migrated using an online table move.

## File type modifiers for the load utility

Table 32. Valid file type modifiers for the load utility: All file formats

Modifier	Description
<b>anyorder</b>	This modifier is used in conjunction with the <b>cpu_parallelism</b> parameter. Specifies that the preservation of source data order is not required, yielding significant additional performance benefit on SMP systems. If the value of <b>cpu_parallelism</b> is 1, this option is ignored. This option is not supported if <b>SAVECOUNT</b> > 0, since crash recovery after a consistency point requires that data be loaded in sequence.
<b>generatedignore</b>	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 <b>generatedmissing</b> or the <b>generatedoverride</b> modifier.
<b>generatedmissing</b>	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 <b>generatedignore</b> or the <b>generatedoverride</b> modifier.

Table 32. Valid file type modifiers for the load utility: All file formats (continued)

Modifier	Description
<b>generatedoverride</b>	<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 <b>RECOVER DROPPED TABLE</b> option on the <b>ROLLFORWARD DATABASE</b> 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 <i>table-name</i> 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 <i>table-name</i> 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 <b>LOAD</b> command will automatically convert the modifier to <b>generatedignore</b> 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 <b>generatedmissing</b> or the <b>generatedignore</b> modifier.</p>
<b>identityignore</b>	<p>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 <b>GENERATED ALWAYS</b> and <b>GENERATED BY DEFAULT</b> identity columns. This means that for <b>GENERATED ALWAYS</b> columns, no rows will be rejected. This modifier cannot be used with either the <b>identitymissing</b> or the <b>identityoverride</b> modifier.</p>
<b>identitymissing</b>	<p>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 <b>GENERATED ALWAYS</b> and <b>GENERATED BY DEFAULT</b> identity columns. This modifier cannot be used with either the <b>identityignore</b> or the <b>identityoverride</b> modifier.</p>
<b>identityoverride</b>	<p>This modifier should be used only when an identity column defined as <b>GENERATED ALWAYS</b> 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 <b>GENERATED ALWAYS</b>, or when loading a table from data that was recovered using the <b>DROPPED TABLE RECOVERY</b> option on the <b>ROLLFORWARD DATABASE</b> 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 <b>identitymissing</b> or the <b>identityignore</b> 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.</p>
<b>implicitlyhiddeninclude</b>	<p>If this modifier is specified, the utility assumes that the input data file contains data for the implicitly hidden columns and this data will also be loaded. This modifier cannot be used with the <b>implicitlyhiddenmissing</b> modifier. See the Note: section for information about the precedence when multiple modifiers are specified.</p>

## LOAD

Table 32. Valid file type modifiers for the load utility: All file formats (continued)

Modifier	Description
<b>implicitlyhiddenmissing</b>	If this modifier is specified, the utility assumes that the input data file does not contain data for the implicitly hidden columns and the utility will generate values for those hidden columns. This modifier cannot be used with the <b>implicitlyhiddeninclude</b> modifier. See the Note: section for information about the precedence when multiple modifiers are specified.
<b>indexfreespace=<i>x</i></b>	<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 <b>INDEXING MODE INCREMENTAL</b> ignores this option. The first entry in a page is added without restriction; subsequent entries are added to maintain the percent free space threshold. 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 <b>indexfreespace</b> option affects index leaf pages only.</p>
<b>lobsinfile</b>	<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 <b>LOBS FROM</b> clause specifies where the LOB files are located when the <b>lobsinfile</b> modifier is used. The <b>LOBS FROM</b> clause will implicitly activate the <b>lobsinfile</b> behavior. The <b>LOBS FROM</b> clause conveys to the <b>LOAD</b> 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>
<b>noheader</b>	<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 <b>noheader</b> 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 <b>noheader</b> modifier is if you wanted to perform LOAD_ONLY operation using a file that does not have a header.</p>
<b>norowwarnings</b>	Suppresses all warnings about rejected rows.



Table 32. Valid file type modifiers for the load utility: All file formats (continued)

Modifier	Description
<b>pagefreespace=x</b>	$x$ is an integer between 0 and 100 inclusive. The value is interpreted as the percentage of each data page that is to be left as free space. If the specified value is invalid because of the minimum row size, (for example, a row that is at least 3 000 bytes long, and an $x$ value of 50), the row will be placed on a new page. If a value of 100 is specified, each row will reside on a new page. The PCTFREE value of a table determines the amount of free space designated per page. If a <b>pagefreespace</b> 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 <b>pagefreespace</b> overrides the PCTFREE value specified for the table.
<b>periodignore</b>	This modifier informs the load utility that data for the period columns is present in the data file but should be ignored. When this modifier is specified, all period column values are generated by the utility. This modifier cannot be used with the <b>periodmissing</b> modifier or the <b>periodoverride</b> modifier.
<b>periodmissing</b>	If this modifier is specified, the utility assumes that the input data file contains no data for the period columns. When this modifier is specified, all period column values are generated by the utility. This modifier cannot be used with the <b>periodignore</b> modifier or the <b>periodoverride</b> modifier.
<b>periodoverride</b>	This modifier instructs the load utility to accept user-supplied data for GENERATED ALWAYS AS ROW BEGIN and GENERATED ALWAYS AS ROW END columns in a system-period temporal table. This behavior is contrary to the normal rules for these types of columns. The modifier can be useful when you want to maintain history data and load data that includes time stamps into a system-period temporal table. When this modifier is used, any rows with no data or NULL data in a ROW BEGIN or ROW END column are rejected.
<b>rowchangetimestampignore</b>	This modifier informs the load utility that data for the row change timestamp column is present in the data file but should be ignored. This results in all ROW CHANGE TIMESTAMPS being generated by the utility. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT columns. This means that for GENERATED ALWAYS columns, no rows will be rejected. This modifier cannot be used with either the <b>rowchangetimestampmissing</b> or the <b>rowchangetimestampoverride</b> modifier.
<b>rowchangetimestampmissing</b>	If this modifier is specified, the utility assumes that the input data file contains no data for the row change timestamp 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 columns. This modifier cannot be used with either the <b>rowchangetimestampignore</b> or the <b>rowchangetimestampoverride</b> modifier.
<b>rowchangetimestampoverride</b>	This modifier should be used only when a row change timestamp 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 row change timestamp 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 <b>DROPPED TABLE RECOVERY</b> option on the <b>ROLLFORWARD DATABASE</b> command. When this modifier is used, any rows with no data or NULL data for the ROW CHANGE TIMESTAMP column will be rejected (SQL3116W). This modifier cannot be used with either the <b>rowchangetimestampmissing</b> or the <b>rowchangetimestampignore</b> modifier. The load utility will not attempt to maintain or verify the uniqueness of values in the table's row change timestamp column when this option is used.

## LOAD

Table 32. Valid file type modifiers for the load utility: All file formats (continued)

Modifier	Description
<b>seclabelchar</b>	<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. <b>LOAD</b> 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 <b>seclabelname</b> 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 <b>seclabelchar</b> file type modifier must be used: LOAD FROM input.del OF DEL MODIFIED BY SECLABELCHAR INSERT INTO t1</p>
<b>seclabelname</b>	<p>Indicates that security labels in the input source file are indicated by their name rather than the default encoded numeric format. <b>LOAD</b> 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 <b>seclabelchar</b> 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 <b>seclabelname</b> file type modifier must be used: LOAD FROM input.del OF DEL MODIFIED BY SECLABELNAME INSERT INTO t1</p> <p><b>Note:</b> 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 <b>striptblanks</b> file type modifier to make sure the spaces are removed.</p>
<b>total freespace=x</b>	<p><math>x</math> 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 <math>x</math> 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>
<b>transactionidignore</b>	<p>This modifier informs the load utility that data for the TRANSACTION START ID column is present in the data file but should be ignored. When this modifier is specified, the value for the TRANSACTION START ID column is generated by the utility. This modifier cannot be used with the <b>transactionidmissing</b> modifier or the <b>transactionidoverride</b> modifier.</p>

Table 32. Valid file type modifiers for the load utility: All file formats (continued)

Modifier	Description
<b>transactionidmissing</b>	If this modifier is specified, the utility assumes that the input data file contains no data for the TRANSACTION START ID columns. When this modifier is specified, the value for the TRANSACTION START ID column is generated by the utility. This modifier cannot be used with the <b>transactionidignore</b> modifier or the <b>transactionidoverride</b> modifier.
<b>transactionidoverride</b>	This modifier instructs the load utility to accept user-supplied data for the GENERATED ALWAYS AS TRANSACTION START ID column in a system-period temporal table. This behavior is contrary to the normal rules for this type of column. When this modifier is used, any rows with no data or NULL data in a TRANSACTION START ID column are rejected.
<b>usedefaults</b>	<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"> <li>• 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.</li> <li>• For DEL/ASC 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.</li> </ul> <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"> <li>• For DEL/ASC files: If the column is nullable, a NULL is loaded. If the column is not nullable, the utility rejects the row.</li> </ul>

Table 33. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL)

Modifier	Description
<b>codepage=<i>x</i></b>	<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"> <li>• For pure DBCS (graphic), mixed DBCS, and EUC, delimiters are restricted to the range of x00 to x3F, inclusive.</li> <li>• For DEL data specified in an EBCDIC code page, the delimiters might not coincide with the shift-in and shift-out DBCS characters.</li> <li>• <b>nullindchar</b> 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.</li> </ul> <p>This option is not supported in conjunction with the CURSOR filetype.</p>

## LOAD

Table 33. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
<code>dateformat="x"</code>	<p><i>x</i> is the format of the date in the source file.<sup>1</sup> 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 01 - 12; mutually exclusive with M)  D - Day (one or two digits ranging from 1 - 31)  DD - Day (two digits ranging from 01 - 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>
<code>dumpfile = x</code>	<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. The following section 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 <code>dumpfileaccessall</code> file type modifier.</p> <p><b>Note:</b></p> <ol style="list-style-type: none"> <li>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.</li> <li>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 <b>LOAD RESTART</b>. The file can only be assumed to be complete for a load operation that starts and completes in a single pass.</li> <li>3. If the specified file already exists, it will not be re-created, but it will be truncated.</li> </ol>
<code>dumpfileaccessall</code>	<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"> <li>1. it is used in conjunction with <code>dumpfile</code> file type modifier</li> <li>2. the user has SELECT privilege on the load target table</li> <li>3. it is issued on a DB2 server database partition that resides on a UNIX operating system</li> </ol> <p>If the specified file already exists, its permissions will not be changed.</p>
<code>fastparse</code>	<p>Use with caution. Reduces syntax checking on user-supplied column values, and enhances performance. Tables are guaranteed to be architecturally correct (the utility performs sufficient data checking to prevent a segmentation violation or trap), however, the coherence of the data is not validated. Only use this option if you are certain that your data is coherent and correct. For example, if the user-supplied data contains an invalid timestamp column value of <code>:1&gt;0-00-20-07.11.12.000000</code>, this value is inserted into the table if <code>fastparse</code> is specified, and rejected if <code>fastparse</code> is not specified.</p>

Table 33. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
<b>implieddecimal</b>	<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 <b>packeddecimal</b> modifier.</p>
<b>timeformat="x"</b>	<p><i>x</i> is the format of the time in the source file.<sup>1</sup> Valid time elements are:</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 00 - 12 for a 12 hour system, and 00 - 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 00 - 59; mutually exclusive with M)</p> <p>S - Second (one or two digits ranging from 0 - 59)</p> <p>SS - Second (two digits ranging from 00 - 59; mutually exclusive with S)</p> <p>SSSSS - Second of the day after midnight (5 digits ranging from 00000 - 86400; mutually exclusive with other time elements)</p> <p>TT - Meridian indicator (AM or PM)</p> <p>A default value of 0 is assigned for each element that is not specified. Some examples of time formats are:</p> <p>"HH:MM:SS"</p> <p>"HH.MM TT"</p> <p>"SSSSS"</p>

## LOAD

Table 33. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
<b>timestampformat="x"</b>	<p>x is the format of the time stamp in the source file.<sup>1</sup> 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 01 - 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 00 - 12 for a 12 hour system, and 00 - 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 00 - 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 00 - 59; mutually exclusive with S)</p> <p>SSSSS - Second of the day after midnight (5 digits ranging from 00000 - 86400; mutually exclusive with other time elements)</p> <p>U (1 to 12 times) - Fractional seconds (number of occurrences of U represent the number of digits with each digit ranging from 0 to 9)</p> <p>TT - Meridian indicator (AM or PM)</p>
<b>timestampformat="x"</b> (Continued)	<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. The following section is an example of a time stamp format:</p> <p style="text-align: center;">"YYYY/MM/DD HH:MM:SS.UUUUUU"</p> <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>If the <b>timestampformat</b> modifier is not specified, the load utility formats the timestamp field using one of two possible formats:</p> <p>YYYY-MM-DD-HH.MM.SS YYYY-MM-DD HH:MM:SS</p> <p>The load utility chooses the format by looking at the separator between the DD and HH. If it is a dash '-', the load utility uses the regular dashes and dots format (YYYY-MM-DD-HH.MM.SS). If it is a blank space, then the load utility expects a colon ':' to separate the HH, MM and SS.</p> <p>In either format, if you include the microseconds field (UUUUUU), the load utility expects the dot '.' as the separator. Either YYYY-MM-DD-HH.MM.SS.UUUUUU or YYYY-MM-DD HH:MM:SS.UUUUUU are acceptable.</p> <p>The following example illustrates how to load data containing user defined date and time formats into a table called schedule:</p> <pre>db2 load from delfile2 of del modified by timestampformat="yyyy.mm.dd hh:mm tt" insert into schedule</pre>

Table 33. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
<b>usegraphiccodepage</b>	<p>If <b>usegraphiccodepage</b> 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. <b>LOAD</b> determines the character code page through either the <b>codepage</b> modifier, if it is specified, or through the code page of the database if the <b>codepage</b> 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><b>Restrictions</b></p> <p>The <b>usegraphiccodepage</b> modifier <b>MUST NOT</b> be specified with DEL files created by the <b>EXPORT</b> utility, as these files contain data encoded in only one code page. The <b>usegraphiccodepage</b> modifier is also ignored by the double-byte character large objects (DBCLOBs) in files.</p>
<b>xmlchar</b>	<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 <b>codepage</b> 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>
<b>xmlgraphic</b>	<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 <b>codepage</b> 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>

## LOAD

Table 34. Valid file type modifiers for the load utility: ASC file formats (Non-delimited ASCII)

Modifier	Description
<b>binarynumerics</b>	<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 <b>reclen</b> option.</p> <p>The following rules apply:</p> <ul style="list-style-type: none"> <li>• No conversion between data types is performed, with the exception of BIGINT, INTEGER, and SMALLINT.</li> <li>• Data lengths must match their target column definitions.</li> <li>• FLOATs must be in IEEE Floating Point format.</li> <li>• Binary data in the load source file is assumed to be big-endian, regardless of the platform on which the load operation is running.</li> </ul> <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>
<b>nochecklengths</b>	<p>If <b>nochecklengths</b> 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>
<b>nullindchar=<i>x</i></b>	<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.<sup>2</sup></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>
<b>packeddecimal</b>	<p>Loads packed-decimal data directly, since the <b>binarynumerics</b> modifier does not include the DECIMAL field type.</p> <p>This option is supported only with positional ASC, using fixed length records specified by the <b>reclen</b> option.</p> <p>Supported values for the sign nibble are:</p> <pre>+ = 0xC 0xA 0xE 0xF - = 0xD 0xB</pre> <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 <b>implieddecimal</b> modifier.</p>
<b>reclen=<i>x</i></b>	<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 newline character is not used to indicate the end of the row.</p>



Table 34. Valid file type modifiers for the load utility: ASC file formats (Non-delimited ASCII) (continued)

Modifier	Description
<b>striptblanks</b>	<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 <b>striptnulls</b>. These are mutually exclusive options. This option replaces the obsolete <b>t</b> option, which is supported for earlier compatibility only.</p>
<b>striptnulls</b>	<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 <b>striptblanks</b>. These are mutually exclusive options. This option replaces the obsolete <b>padwithzero</b> option, which is supported for earlier compatibility only.</p>
<b>zoneddecimal</b>	<p>Loads zoned decimal data, since the <b>binarynumerics</b> modifier does not include the DECIMAL field type. This option is supported only with positional ASC, using fixed length records specified by the <b>reclen</b> option.</p> <p>Half-byte sign values can be one of the following value:</p> <p>+ = 0xC 0xA 0xE 0xF 0x3 - = 0xD 0xB 0x7</p> <p>Supported values for digits are 0x0 to 0x9.</p> <p>Supported values for zones are 0x3 and 0xF.</p>

Table 35. Valid file type modifiers for the load utility: DEL file formats (Delimited ASCII)

Modifier	Description
<b>charde1x</b>	<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.<sup>23</sup> If you want to explicitly specify the double quotation mark (") as the character string delimiter, you should specify it as follows:</p> <p>modified by charde1""</p> <p>The single quotation mark (') can also be specified as a character string delimiter as follows:</p> <p>modified by charde1''</p>
<b>colde1x</b>	<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.<sup>23</sup></p>
<b>decplusblank</b>	<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>
<b>decptx</b>	<p><i>x</i> 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.<sup>23</sup></p>

## LOAD

Table 35. Valid file type modifiers for the load utility: DEL file formats (Delimited ASCII) (continued)

Modifier	Description
<b>delprioritychar</b>	<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&lt;row delimiter&gt; "Vincent,&lt;row delimiter&gt;, is a manager", ... ... 4005,44.37&lt;row delimiter&gt;</pre> <p>With the <b>delprioritychar</b> modifier specified, there will be only two rows in this data file. The second &lt;row delimiter&gt; will be interpreted as part of the first data column of the second row, while the first and the third &lt;row delimiter&gt; 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 &lt;row delimiter&gt;.</p>
<b>keepblanks</b>	<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>
<b>nocharde1</b>	<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 a DB2 database system (unless <b>nocharde1</b> 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 <b>charde1x</b>, <b>delprioritychar</b> or <b>nodoublede1</b>. These are mutually exclusive options.</p>
<b>nodoublede1</b>	<p>Suppresses recognition of double character delimiters.</p>

Table 36. Valid file type modifiers for the load utility: IXF file format

Modifier	Description
<b>forcein</b>	<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 <b>nochecklengths</b> is specified, no checking is done, and an attempt is made to load each row.</p>
<b>nochecklengths</b>	<p>If <b>nochecklengths</b> 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>

**Note:**

1. Double quotation marks around the date format string are mandatory. Field separators cannot contain any of the following characters: 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. Character values provided for the **charde1**, **colde1**, or **decpt** file type modifiers 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 statements:

```
... modified by colde1# ...
... modified by colde10x23 ...
... modified by colde1X23 ...
```

3. "Delimiter considerations for moving data" lists restrictions that apply to the characters that can be used as delimiter overrides.
4. 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.
5. When multiple modifiers suffixed with **ignore**, **include**, **missing**, and **override** are specified, they are applied in the order that they are listed. In the following statement, data for implicitly hidden columns that are not identity columns is included in the input data. While data for all identity columns, regardless of their implicitly hidden status, is not.

```
db2 load from delfile1 of del modified by
    implicitlyhiddeninclude identitymissing insert into table1
```

However, changing the order of the file type modifiers in the following statement means that data for all implicitly hidden columns (including hidden identity columns) is included in the input data. While data for identity columns that are not implicitly hidden is not.

```
db2 load from delfile1 of del modified by
    identitymissing implicitlyhiddeninclude insert into table1
```

## LOAD

Table 37. 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 <b>CLIENT</b> option is specified.
Present	Absent	All data in the file is assumed to be in code page N. <b>Warning:</b> 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 <b>CLIENT</b> option is specified. Graphic data is assumed to be in the code page of the database graphic data, even if the <b>CLIENT</b> option is specified.  If the database code page is single-byte, then all data is assumed to be in the database code page. <b>Warning:</b> 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. <b>Warning:</b> Graphic data will be corrupted when loaded into the database if N is a single-byte code page.

### Related information:

## LOAD QUERY

Checks the status of a load operation during processing and returns the table state. If a load is not processing, then the table state alone is returned.

A connection to the same database, and a separate CLP session are also required to successfully invoke this command. It can be used either by local or remote users.

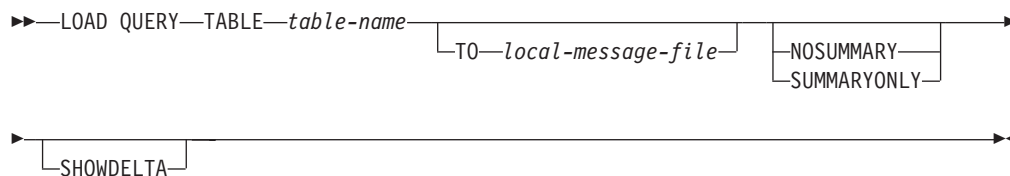
### Authorization

None

### Required connection

Database

### Command syntax



### Command parameters

#### NOSUMMARY

Specifies that no load summary information (rows read, rows skipped, rows loaded, rows rejected, rows deleted, rows committed, and number of warnings) is to be reported.

#### SHOWDELTA

Specifies that only new information (pertaining to load events that have occurred since the last invocation of the **LOAD QUERY** command) is to be reported.

#### SUMMARYONLY

Specifies that only load summary information is to be reported.

#### TABLE *table-name*

Specifies the name of the table into which data is currently being loaded. If an unqualified table name is specified, the table will be qualified with the CURRENT SCHEMA.

#### TO *local-message-file*

Specifies the destination for warning and error messages that occur during the load operation. This file cannot be the *message-file* specified for the **LOAD** command. If the file already exists, all messages that the load utility has generated are appended to it.

### Examples

A user loading a large amount of data into the STAFF table in the BILLYBOB database, wants to check the status of the load operation. The user can specify:

```

db2 connect to billybob
db2 load query table staff to /u/mydir/staff.tempsmsg

```

## LOAD QUERY

The output file /u/mydir/staff.tempsmsg might look like the following output:

```
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.de1"
```

```
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 Only state and all other data partitions are in Normal state, the load query operation returns the Read Access Only 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

A table is in Normal state if it is not in any of the other (abnormal) table states. Normal state is the initial state of a table after it is created.

#### 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 in Progress**

This is a transient state that is only in effect during a load operation.

#### **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**

A table is in this state during a load operation if the **ALLOW READ ACCESS** option was specified. Read Access Only is a transient state that allows other applications and utilities to have read access to data that existed before the load operation.

#### **Reorg Pending**

A **REORG** command 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 to determine the table state.

There are currently at least 25 table or table space states supported by the IBM DB2 database product. These states are used to control access to data under certain circumstances, or to elicit specific user actions, when required, to protect the integrity of the database. Most of them result from events related to the operation of one of the DB2 database utilities, such as the load utility, or the backup and restore utilities.

Although dependent table spaces are no longer quiesced (a quiesce is a persistent lock) before a load operation, the Load in Progress table space state prevents the backup of dependent tables during a load operation. The Load in Progress table space state is different from the Load in Progress table state: All load operations use the Load in Progress table state, but load operations (against a recoverable database) with the **COPY NO** option specified also use the Load in Progress table space state.

The following table describes each of the supported table states. The table also provides you with working examples that show you exactly how to interpret and

## LOAD QUERY

respond to states that you might encounter while administering your database. The examples are taken from command scripts that were run on AIX; you can copy, paste and run them yourself. If you are running the DB2 database product on a system that is not UNIX, ensure that any path names are in the correct format for your system. Most of the examples are based on tables in the SAMPLE database that comes with the DB2 database product. A few examples require scenarios that are not part of the SAMPLE database, but you can use a connection to the SAMPLE database as a starting point.

Table 38. Supported table states

State	Examples
Load Pending	<p>Given load input file staffdata.del with a substantial amount of data (for example, 20000 or more records), create a small table space that contains the target table of the load operation, a new table called NEWSTAFF:</p> <pre>connect to sample; create tablespace ts1 managed by database using (file '/home/melnik/melnik/NODE0000 /SQL00001/ts1c1' 256); create table newstaff like staff in ts1; load from staffdata.del of del insert into newstaff; load query table newstaff; load from staffdata.del of del terminate into newstaff; load query table newstaff; connect reset;</pre> <p>Information returned by the <b>LOAD QUERY</b> command shows that the NEWSTAFF table is in Load Pending state; after a load terminate operation, the table is in Normal state.</p>
Load in Progress	<p>Given load input file staffdata.del with a substantial amount of data (for example, 20000 or more records):</p> <pre>connect to sample; create table newstaff like staff; load from staffdata.del of del insert into newstaff;</pre> <p>While the load operation is running, execute the following script from another session:</p> <pre>connect to sample; load query table newstaff; connect reset;</pre> <p>Information returned by the <b>LOAD QUERY</b> command shows that the NEWSTAFF table is in Load in Progress state.</p>
Normal	<pre>connect to sample; create table newstaff like staff; load query table newstaff;</pre> <p>Information returned by the <b>LOAD QUERY</b> command shows that the NEWSTAFF table is in Normal state.</p>



Table 38. Supported table states (continued)

State	Examples
Not Load Restartable	<p>Given load input file staffdata.del with a substantial amount of data (for example, 20000 or more records):</p> <pre>update db cfg for sample using logarchmeth1 logretain; backup db sample; connect to sample; create tablespace ts1 managed by database using (file '/home/melnyk/melnyk/NODE0000 /SQL00001/ts1c1' 256); create table newstaff like staff in ts1; connect reset; backup db sample;</pre> <p>The timestamp for this backup image is: 20040629205935</p> <pre>connect to sample; load from staffdata.del of del insert into newstaff copy yes to /home/melnyk/backups; connect reset; restore db sample taken at 20040629205935; rollforward db sample to end of logs and stop; connect to sample; load query table newstaff; connect reset;</pre> <p>Information returned by the <b>LOAD QUERY</b> command shows that the NEWSTAFF table is in Not Load Restartable and Load Pending state.</p> <pre>connect to sample; load from staffdata.del of del terminate into newstaff copy yes to /home/melnyk/backups; load query table newstaff; connect reset;</pre> <p>Information returned by the <b>LOAD QUERY</b> command shows that the NEWSTAFF table is now in Normal state.</p>
Read Access Only	<p>Given load input file staffdata.del with a substantial amount of data (for example, 20000 or more records):</p> <pre>connect to sample; export to st_data.del of del select * from staff; create table newstaff like staff; import from st_data.del of del insert into newstaff; load from staffdata.del of del insert into newstaff allow read access;</pre> <p>While the load operation is running, execute the following script from another session:</p> <pre>connect to sample; load query table newstaff; select * from newstaff; connect reset;</pre> <p>Information returned by the <b>LOAD QUERY</b> command shows that the NEWSTAFF table is in Read Access Only and Load in Progress state. The query returns only the exported contents of the STAFF table, data that existed in the NEWSTAFF table before the load operation.</p>
Set Integrity Pending	<p>Given load input file staff_data.del with content:</p> <pre>11,"Melnyk",20,"Sales",10,70000,15000:</pre> <pre>connect to sample; alter table staff add constraint max_salary check (100000 - salary &gt; 0); load from staff_data.del of del insert into staff; load query table staff;</pre> <p>Information returned by the <b>LOAD QUERY</b> command shows that the STAFF table is in Set Integrity Pending state.</p>

## LOAD QUERY

Table 38. Supported table states (continued)

State	Examples
Unavailable	<p>Given load input file staff_data.del with content:</p> <pre>11,"Melnyk",20,"Sales",10,70000,15000: update db cfg for sample using logarchmeth1 logretain; backup db sample;</pre> <p>The timestamp for this backup image is: 20040629182012</p> <pre>connect to sample; load from staff_data.del of del insert into staff nonrecoverable; connect reset; restore db sample taken at 20040629182012; rollforward db sample to end of logs and stop; connect to sample; load query table staff; connect reset;</pre> <p>Information returned by the <b>LOAD QUERY</b> command shows that the STAFF table is in Unavailable state.</p>

The progress of a load operation can also be monitored with the **LIST UTILITIES** command.

## MIGRATE DATABASE

Converts a previous version of a DB2 database to the formats corresponding to the release run by the instance.

This command is deprecated and will be discontinued in a future release. You should use the **UPGRADE DATABASE** command instead.

### Authorization

SYSADM

### Required connection

This command establishes a database connection.

### Command syntax

```

▶▶ MIGRATE DATABASE database-alias
      DB
▶
  USER username
    USING password
  
```

### Command parameters

**DATABASE** *database-alias*

Specifies the alias of the database to be migrated to the currently installed version of the database manager.

**USER** *username*

Identifies the user name under which the database is to be migrated.

**USING** *password*

The password used to authenticate the user name. If the password is omitted, but a user name was specified, the user is prompted to enter it.

### Usage notes

Refer to the **UPGRADE DATABASE** command documentation.

## PING

Tests the network response time of the underlying connectivity between a client and a connected database server.

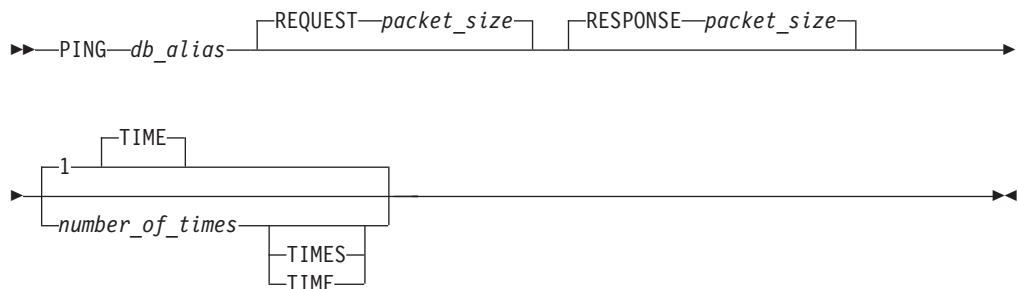
### Authorization

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 Universal Database 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:

```
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 IBM data server 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.

---

## 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 authorizations:

- DBADM authority
- If EXPLAIN ONLY is specified, EXPLAIN authority or an authority that implicitly includes EXPLAIN is sufficient.
- If a package does not exist, BINDADD authority and:
  - If the schema name of the package does not exist, IMPLICIT\_SCHEMA authority on the database.
  - If the schema name of the package does exist, CREATEIN privilege on the schema.
- If the package exists, one of the following privileges:
  - ALTERIN privilege on the schema
  - BIND privilege on the package

In addition, if capturing explain information using the EXPLAIN or the EXPLSNAP clause, one of the following authorizations is required:

- INSERT privilege on the explain tables
- DATAACCESS authority

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.

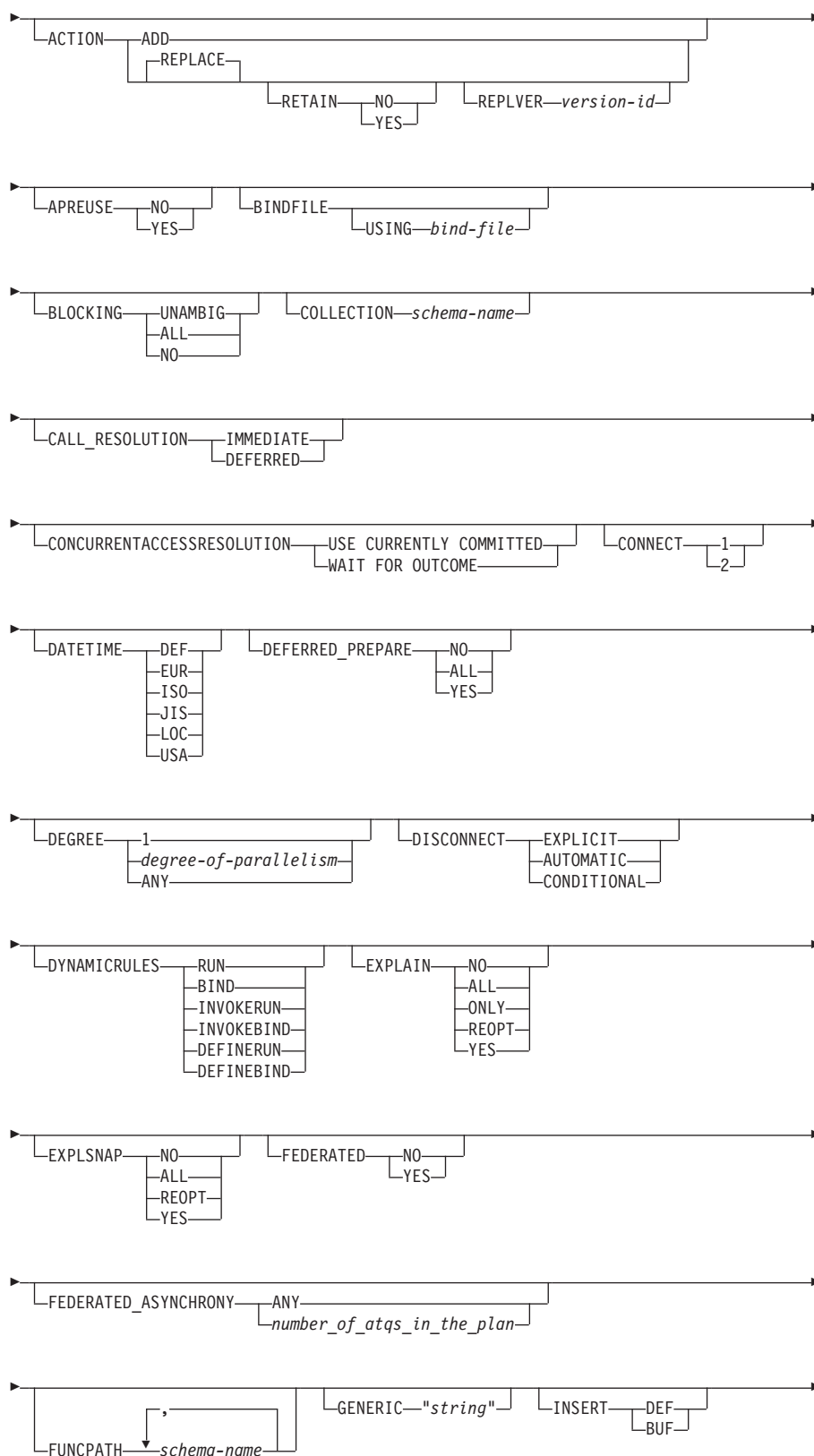
### Required connection

Database. If implicit connect is enabled, a connection to the default database is established.

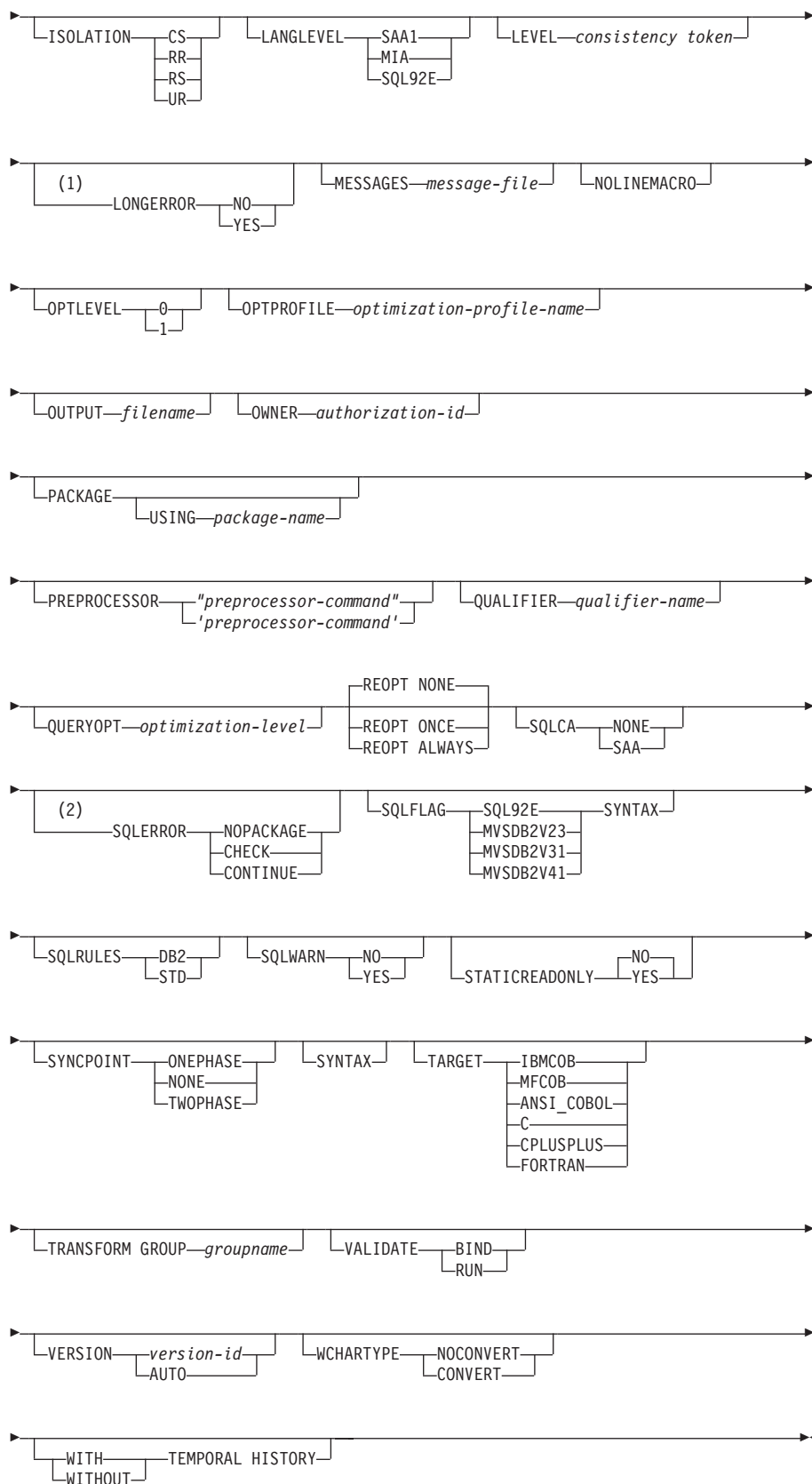
### Command syntax

For DB2 Database for Linux, UNIX, and Windows

```
➤ PRECOMPILE filename ➤  
  └── PREP ───┘
```



# PRECOMPILE

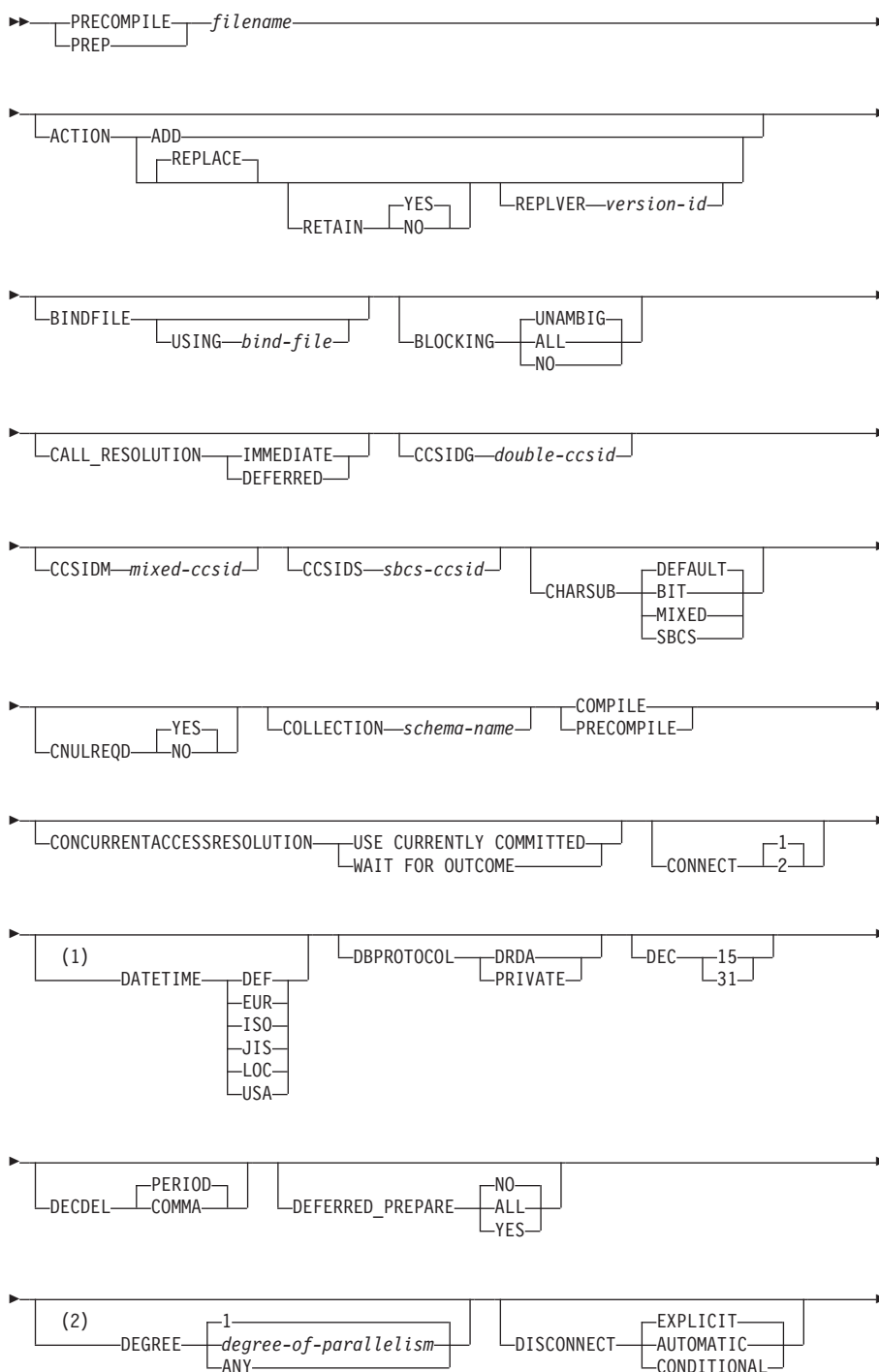




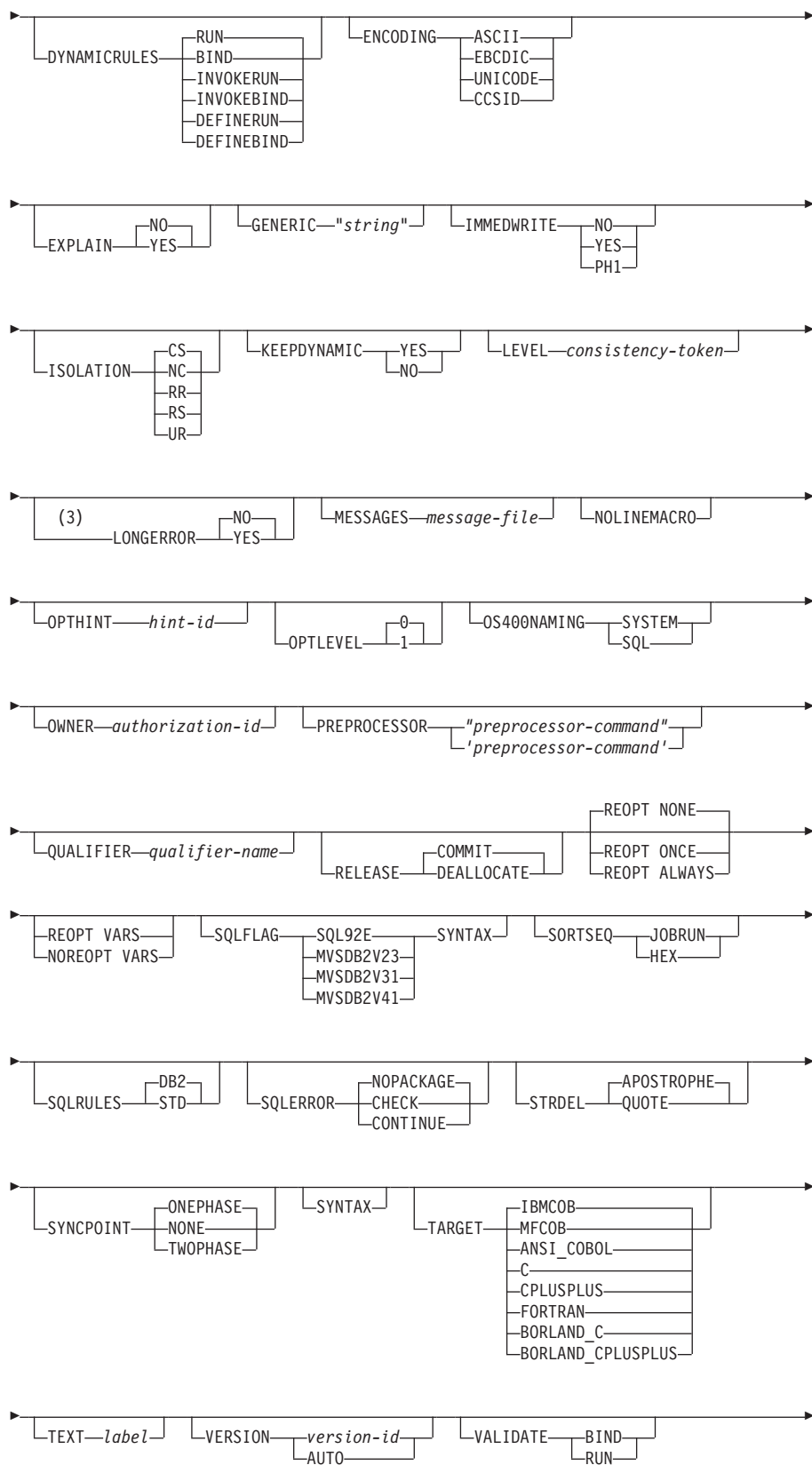
Notes:

- 1 NO is the default for 32 bit systems and for 64 bit Windows 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 Database on servers other than Linux, 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.
- 3 NO is the default for 32 bit systems and for 64 bit Windows 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 and Linux operating systems) must be specified for C++ applications (generates a .cxx file on Windows operating systems, or a .C file on UNIX and Linux operating systems)
- .sqb must be specified for COBOL applications (generates a .cbl file)
- .sqf must be specified for FORTRAN applications (generates a .for file on Windows operating systems, or a .f file on UNIX and Linux operating systems).

The preferred extension for C++ applications containing embedded SQL on UNIX and Linux operating systems is sqC; however, the sqx convention, which was invented for systems that are not case sensitive, is tolerated by UNIX and Linux operating 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.

## PRECOMPILE

### **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.

### **APREUSE**

Specifies whether static SQL access plans are to be reused. When this option is enabled, the query compiler will attempt to reuse the access plans for the statement in any existing packages during the bind and during future implicit and explicit rebinds.

**YES** The query compiler will attempt to reuse the access plans for the statements in the package. If there is an existing package, the query compiler will attempt to reuse the access plan for every statement that can be matched with a statement in the new bind file. For a statement to match, the statement text must be identical and the section number for the statement in the existing package must match what the section number will be for the statement in the new package.

**NO** The query compiler will not attempt to reuse access plans for the statements in the package. This is the default setting.

### **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 authorization 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. The blocking of row data that contains references to LOB column data types is also supported in partitioned database environments.

**ALL** For cursors that are specified with the FOR READ ONLY clause or cursors not specified as FOR UPDATE, blocking occurs.

Ambiguous cursors are treated as read-only.

**NO** Blocking does not occur for any cursor.

For the definition of a read-only cursor and an ambiguous cursor, refer to *DECLARE CURSOR statement*.

Ambiguous cursors are treated as updatable.

#### UNAMBIG

For cursors that are specified with the FOR READ ONLY clause, blocking occurs.

Cursors that are not declared with the FOR READ ONLY or FOR UPDATE clause which are not ambiguous and are read-only will be blocked. Ambiguous cursors will not be blocked.

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 sqleproc() 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 the server for DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a built-in 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 the server for DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a built-in default value if this option is not specified.

#### CCSIDS *sbcs-ccsid*

An integer specifying the coded character set identifier (CCSID) to be used for single byte characters in character column definitions (without a specific CCSID clause) in CREATE and ALTER TABLE SQL statements. This option is not supported by the server for DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a built-in 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 precompile/bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows.

**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 built-in default in all new character columns for which an explicit sub-type is not specified.

## PRECOMPILE

### 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. It is valid only if the bind file is created from a C or a C++ application. This bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows.

**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 128-byte collection identifier for the package. If not specified, the authorization identifier for the user processing the package is used.

### CONCURRENTACCESSRESOLUTION

Specifies the concurrent access resolution to use for statements in the package.

### USE CURRENTLY COMMITTED

Specifies that the database manager can use the currently committed version of the data for applicable scans when it is in the process of being updated or deleted. Rows in the process of being inserted can be skipped. This clause applies when the isolation level in effect is Cursor Stability or Read Stability (for Read Stability it skips uncommitted inserts only) and is ignored otherwise. Applicable scans include read-only scans that can be part of a read-only statement as well as a non read-only statement. The settings for the registry variables **DB2\_EVALUNCOMMITTED**, **DB2\_SKIPDELETED**, and **DB2\_SKIPINSERTED** no longer apply.

### WAIT FOR OUTCOME

Specifies Cursor Stability and higher scans wait for the commit or rollback when encountering data in the process of being updated or deleted. Rows in the process of being inserted are not skipped. The settings for the registry variables **DB2\_EVALUNCOMMITTED**, **DB2\_SKIPDELETED**, and **DB2\_SKIPINSERTED** no longer apply.

### 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 precompile or bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a built-in 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 precompile/bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a built-in 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.

## PRECOMPILE

### DEGREE

Specifies the degree of parallelism for the execution of static SQL statements in an SMP system. This option does not affect CREATE INDEX parallelism.

**1** The execution of the statement will not use parallelism.

#### *degree-of-parallelism*

Specifies the degree of parallelism with which the statement can be executed, a value between 2 and 32 767 (inclusive).

**ANY** Specifies that the execution of the statement can involve parallelism using a degree determined by the database manager.

### DISCONNECT

#### **AUTOMATIC**

Specifies that all database connections are to be disconnected at commit.

#### **CONDITIONAL**

Specifies that the database connections that have been marked RELEASE or have no open WITH HOLD cursors are to be disconnected at commit.

#### **EXPLICIT**

Specifies that only database connections that have been explicitly marked for release by the RELEASE statement are to be disconnected at commit.

### DYNAMICRULES

Defines which rules apply to dynamic SQL at run time for the initial setting of the values used for authorization ID and for the implicit qualification of unqualified object references.

**RUN** Specifies that the authorization ID of the user executing the package is to be used 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 stand-alone 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 stand-alone 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 stand-alone 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 stand-alone 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.

**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

## PRECOMPILE

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).

**ONLY** The **ONLY** option allows you to explain statements without having the privilege to execute them. The explain tables are populated but no persistent package is created. If an existing package with the same name and version is encountered during the bind process, the existing package is neither dropped nor replaced even if you specified **ACTION REPLACE**. If an error occurs during population of the explain tables, explain information is not added for the statement that returned the error and for any statements that follow it.

**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.

**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 or 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.

**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.

### FEDERATED\_ASYNCHRONY

Specifies the maximum number of asynchrony table queues (ATQs) that the federated server supports in the access plan for programs that use embedded SQL.

**ANY** The optimizer determines the number of ATQs for the access plan. The optimizer assigns an ATQ to all eligible SHIP or remote pushdown operators in the plan. The value that is specified for **DB2\_MAX\_ASYNC\_REQUESTS\_PER\_QUERY** server option limits the number of asynchronous requests.

*number\_of\_atqs\_in\_the\_plan*

The number of ATQs in the plan. You specify a number in the range 0 to 32767.

### 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

## PRECOMPILE

schema cannot appear more than once in the function path. The schema name SYSPUBLIC cannot be specified for the function path. The number of schemas that can be specified is limited by the length of the resulting function path, which cannot exceed 2048 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 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. Do not use this option to pass bind options that *are* defined in BIND or PRECOMPILE. 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, one could use the following to bind each of the OPT1, OPT2, and OPT3 options:

```
generic "opt1 value1 opt2 value2 opt3 value3"
```

The maximum length of the string is 4096 bytes. The maximum length of each option name in the string is 255 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.

### KEEPDYNAMIC

This parameter specifies whether dynamic SQL statements are to be kept across transactions.

For details about the DB2 for z/OS supported option, see documentation in the *DB2 for z/OS Information Center*.

Starting with DB2 for Linux, UNIX, and Windows Version 9.8 Fix Pack 2, dynamic SQL statements prepared in a package bound with the

**KEEPDYNAMIC** YES option are kept in the SQL context after a COMMIT or ROLLBACK operation. This is the default behavior.

#### YES

Instructs the SQL context to keep the statement text and section associated with prepared statements indefinitely. Dynamic SQL statements are kept across transactions. All packages bound with **KEEPDYNAMIC** YES are by default compatible with the existing package cache behavior.

#### No

Instructs the SQL context to remove the statement text and section associated with prepared statements at the end of each unit of work. Inactive dynamic SQL statements prepared in a package bound with **KEEPDYNAMIC** NO are removed from the SQL context during a COMMIT or ROLLBACK operation. The statements must be prepared again in a new transaction. The client, driver, or application needs to prepare any dynamic SQL statement it wants to reuse in a new unit of work again.

Dynamic SQL statements can remain active beyond the end of a transaction under the following circumstances:

- Cursors declared using the WITH HOLD option are open at a commit point.
- A dynamic SQL statement is executing a COMMIT or ROLLBACK operation.
- A dynamic SQL statement invokes a stored procedure or a user defined function that is executing COMMIT or ROLLBACK operation.

#### LANGLEVEL

Specifies the SQL rules that apply for both the syntax and the semantics for both static and dynamic SQL in the application.

**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 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:

## PRECOMPILE

- 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 side of the assignment clause.
- A column function that can be resolved using an index (for example MIN or MAX) will also check for nulls and return warning SQLSTATE 01003 if there were any nulls.
- An error is returned when a duplicate unique constraint is included in a CREATE or ALTER TABLE statement.
- An error is returned when no privilege is granted and the grantor has no privileges on the object (otherwise a warning is returned).

### LEVEL *consistency-token*

Defines the level of a module using the consistency token. The consistency token is any alphanumeric value up to 8 characters in length. The RDB package consistency token verifies that the requester's application and the relational database package are synchronized. This option is not recommended for general use.

### LONGERROR

Indicates whether long host variable declarations will be treated as an 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 Windows 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.

#### **OPTPROFILE** *optimization-profile-name*

Specifies the name of an existing optimization profile to be used for all static statements in the package. The default value of the option is an empty string. The value also applies as the default for dynamic preparation of DML statements for which the CURRENT OPTIMIZATION PROFILE special register is null. If the specified name is unqualified, it is an SQL identifier, which is implicitly qualified by the QUALIFIER bind option.

The **BIND** command does not process the optimization file, but only validates that the name is syntactically valid. Therefore if the optimization profile does not exist or is invalid, an SQL0437W warning with reason code 13 will not occur until a DML statement is optimized using that optimization profile.

#### **OUTPUT** *filename*

Overrides the default name of the modified source file produced by the compiler. It can include a path.

## PRECOMPILE

### OS400NAMING

Specifies which naming option is to be used when accessing DB2 for System i data. Supported by DB2 for System i only. For a list of supported option values, refer to the documentation for DB2 for System i.

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 System i 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 System i system naming convention is used, whether or not it has been precompiled or bound using the **OS400NAMING** SYSTEM option.

### OWNER *authorization-id*

Designates a 128-byte 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 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. The *authorization-id* can only be a user (cannot be a role or a group).

### 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 128 bytes.

### PREPROCESSOR "*preprocessor-command*"

Specifies the preprocessor command that can be executed by the precompiler before it processes embedded SQL statements. The preprocessor command string (maximum length 1024 bytes) must be enclosed either by double or by single quotation marks.

This option enables the use of macros within the declare section. A valid preprocessor command is one that can be issued from the command line to invoke the preprocessor without specifying a source file. For example,

```
xlc -P -DMYMACRO=0
```

### QUALIFIER *qualifier-name*

Provides an 128-byte 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 required 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.

### RELEASE

Indicates whether resources are released at each COMMIT point, or when the application terminates. This precompile/bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows.



**COMMIT**

Release resources at each COMMIT point. Used for dynamic SQL statements.

**DEALLOCATE**

Release resources only when the application terminates.

**REOPT**

Specifies whether to have DB2 optimize an access path using values for host variables, parameter markers, global variables, and special registers. Valid values are:

**NONE**

The access path for a given SQL statement containing host variables, parameter markers, global variables, 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, global variables, 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, global variables, 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, global 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.

## PRECOMPILE

### 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.

### MVSDB2V23 SYNTAX

The SQL statements will be checked against MVS DB2 Version 2.3 SQL language syntax. Any deviation from the syntax is reported in the precompiler listing.

### MVSDB2V31 SYNTAX

The SQL statements will be checked against MVS DB2 Version 3.1 SQL language syntax. Any deviation from the syntax is reported in the precompiler listing.

### MVSDB2V41 SYNTAX

The SQL statements will be checked against MVS DB2 Version 4.1 SQL language syntax. Any deviation from the syntax is reported in the precompiler listing.

### SORTSEQ

Specifies which sort sequence table to use on the System i system. Supported by DB2 for System i only. For a list of supported option values, refer to the documentation for DB2 for System i.

### 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 an application specifies the format of LOB columns in the result set.

**DB2**

- Permits the SQL CONNECT statement to switch the current connection to another established (*dormant*) connection.
- This default setting allows an application to specify whether LOB values or LOB locators are retrieved only during the first fetch request. Subsequent fetch requests must use the same format for the LOB columns.

**STD**

- Permits the SQL CONNECT statement to establish a *new* connection only. The SQL SET CONNECTION statement must be used to switch to a dormant connection.
- The application can change between retrieving LOB values and LOB locators with each fetch request. This means that cursors with one or more LOB columns cannot be blocked, regardless of the BLOCKING bind option setting.

**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.

**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 precompile/bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a built-in 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.

**SYNCPPOINT**

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,

## PRECOMPILE

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 Linux, UNIX and Windows operating systems.

### 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 precompile/bind option is not supported by the server for DB2 Database for Linux, UNIX, and Windows.

### 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.

#### *groupname*

An SQL identifier of up to 128 bytes 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 previously listed 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 or 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 or 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*).

### WITH / WITHOUT TEMPORAL HISTORY

Specifies whether changes to data in system-period temporal tables made by static or dynamic SQL statements causes changes to the corresponding history table.

**WITH** Specifies that changes to data in system-period temporal tables causes changes to the corresponding history table.

This is the default option.

### WITHOUT

Specifies that changes to data in system-period temporal tables do not cause changes to the corresponding history table. The database manager can provide values to override the row-begin, row-end, and transaction-start-ID columns even though they are defined as **GENERATED ALWAYS**.

DBADM authority is required for this option.

## 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. Although the maximum length of a package name is 128 bytes, unless the **PACKAGE USING** option is specified, only the first 8 characters of the file name are used to maintain compatibility with previous versions of DB2.

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.

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.

For an embedded SQL program, if the **FEDERATED\_ASYNC** precompile option is not explicitly specified the static statements in the package are bound using the **federated\_async** configuration parameter. If the **FEDERATED\_ASYNC** option is specified explicitly, that value is used for binding the packages and is also the initial value of the special register. Otherwise, the value of the database manager configuration parameter is used as the initial value of the special register. The **FEDERATED\_ASYNC** precompile option influences dynamic SQL only when it is explicitly set.

## 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.

In a partitioned environment, the **PRUNE HISTORY** command only performs on the database partition it is issued on. To prune the history on multiple partitions, you can either issue the **PRUNE HISTORY** command from each individual database partition, or use the `db2_all` prefix to run the **PRUNE HISTORY** command on all database partitions.

**Important:** The **PRUNE LOGFILE** command is deprecated and might be removed in a future release. Use the **PRUNE HISTORY** command instead.

### Authorization

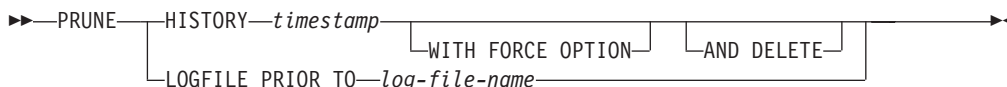
One of the following authorities:

- 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. When an initial prefix is specified, the unspecified components of the time stamp are interpreted as *yyyy0101000000*.

#### **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.

If you set the **auto\_del\_rec\_obj** database configuration parameter to ON, calling **PRUNE HISTORY** with the **AND DELETE** parameter will also physically delete backup images and load copy images if their history file entry is pruned.

**LOGFILE PRIOR TO** *log-file-name*

Specifies a string for a log file name, for example S0000100.LOG. All log files before (but not including) the specified log file will be deleted. The **Logarchmeth1** database configuration parameter must be set to a value other than **OFF**.

**Note:** This value is not supported in DB2 pureScale environments.

## Examples

To remove the entries for all restores, loads, table space backups, and full database backups taken before and including December 1, 1994 from the recovery history file, enter:

```
db2 prune history 199412
```

199412 is interpreted as 19941201000000.

## Usage notes

If the **WITH 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 **db2ckrst** utility from being able to correctly analyze the complete chain of required backup images. Using the **PRUNE HISTORY** command without the **WITH FORCE OPTION** prevents required entries from being deleted.

Those entries with status DB2HISTORY\_STATUS\_DO\_NOT\_DELETE will not be pruned. If the **WITH FORCE OPTION** is used, then objects marked as DB2HISTORY\_STATUS\_DO\_NOT\_DELETE will still be pruned or deleted. You can set the status of recovery history file entries to DB2HISTORY\_STATUS\_DO\_NOT\_DELETE using the **UPDATE HISTORY** command, the ADMIN\_CMD with **UPDATE\_HISTORY**, or the db2HistoryUpdate API. You can use the DB2HISTORY\_STATUS\_DO\_NOT\_DELETE status to prevent key recovery history file entries from being pruned and to prevent associated recovery objects from being deleted.

You can prune snapshot backup database history file entries using the **PRUNE HISTORY** command, but you cannot delete the related physical recovery objects using the **AND DELETE** parameter. The only way to delete snapshot backup object is to use the **db2acsutil** command.



---

## 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 example 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_MEMBER = 0
ATTACH_MEMBER = -1
```

If **CONNECT\_MEMBER** and **ATTACH\_MEMBER** 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\_MEMBER** or the **ATTACH\_MEMBER** 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.

---

## QUIESCE

Forces all users off either the specified instance or database across all members and puts them into a quiesced mode.

While the 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 or database and allow other users to connect to the database.

In this mode, only users with authority in this restricted mode are allowed to attach or connect to the instance or database. Users with SYSADM, SYSMANT, 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 or group and SYSADM, SYSMANT, 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, SYSMANT, and SYSCTRL and allowed user or group.

If an instance is in quiesced mode, a database in the instance cannot be put in quiesced mode.

If a database is in the SUSPEND\_WRITE state, it cannot be put in quiesced mode.

### Authorization

One of the following authorities:

For database level quiesce:

- SYSADM
- DBADM

For instance level quiesce:

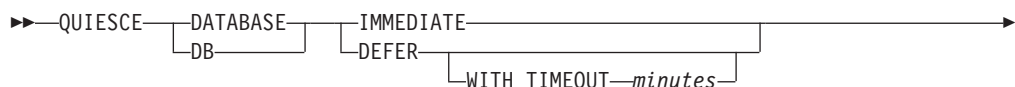
- SYSADM
- SYSCTRL

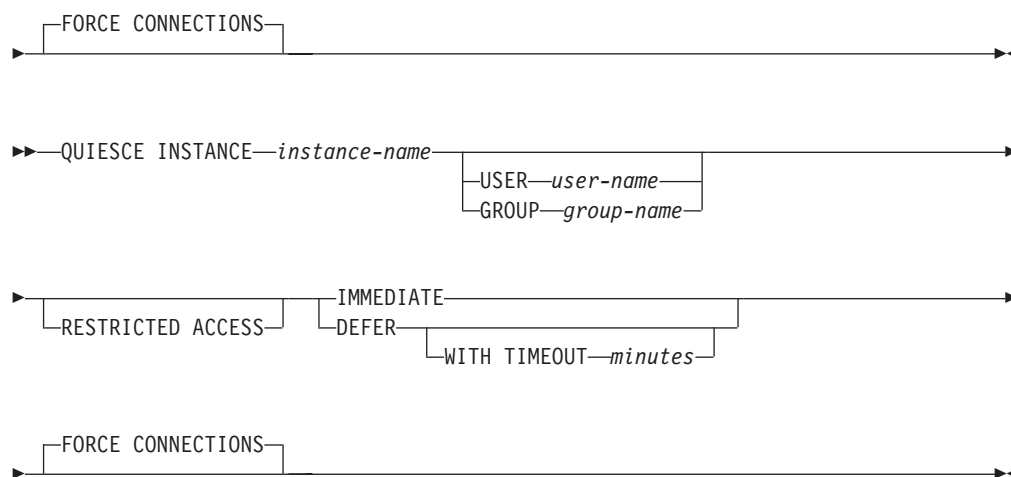
### Required connection

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** *minutes*

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\_time** database manager configuration parameter will be used.

**IMMEDIATE**

Do not wait for the transactions to be committed, immediately roll back 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.

The databases within the quiesced instance will be activated to do authorization checking for all connect attempts to the database. This is necessary to determine if the connecting user ID has DBADM authority; this authority is stored in the database catalog and the database must be activated to determine if the user ID has it. To prevent this authorization checking from happening, specify the **RESTRICTED ACCESS** option.

**USER** *user-name*

Specifies the name of a user who will be allowed access to the instance while it is quiesced.

## QUIESCE

### **GROUP** *group-name*

Specifies the name of a group that will be allowed access to the instance while the instance is quiesced.

### **RESTRICTED ACCESS**

Specify this option to prevent authorization checking for all connect attempts to the databases of a quiesced instance to determine if the user ID has DBADM authority. Instance-level authorization checking can still occur; checking a user ID for SYSADM, SYSCTRL, or SYSMAINT authority does not require a database to be activated.

**Note:** This command option is available in DB2 Version 9.7 Fix Pack 2 and later fix packs.

## 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
```

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 CONNECTIONS** option is supplied. **FORCE CONNECTIONS** is the default behavior; the parameter is allowed in the command for compatibility reasons.
- The command will be synchronized with the **FORCE CONNECTIONS** and will only complete once the **FORCE CONNECTIONS** 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.
- The **RESTRICTED ACCESS** option will prevent the databases within the quiesced instance from being activated to do authorization checking. Any user ID trying to connect to a database, which has DBADM authority or QUIESCE\_CONNECT privilege on the database, will not be allowed to connect. Only user IDs which have SYSADM, SYSCTRL, or SYSMAINT authority and the user or group specified with the command will be allowed to connect to the database.
- The **RESTRICTED ACCESS** option should be used when there is a need to have exclusive connections to a database within the quiesced instance. Such cases can include making an offline backup or performing other maintenance activities.
- After **QUIESCE DATABASE**, users with SYSADM, SYSMAINT, SYSCTRL, or DBADM authority, and GRANT or 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
```

- In a DB2 pureScale environment, after quiescing a database and restarting the instance, the database will remain quiesced across all members. An explicit **UNQUIESCE DATABASE** command is required to remove the quiesce state.

**Related information:**

---

## 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 '', 'A' or 'D') are quiesced.

### Authorization

One of the following authorities:

- SYSADM
- SYCTRL
- SYSMANT
- DBADM
- LOAD

### Required connection

Database

### Command syntax

```
►► QUIESCE TABLESPACES FOR TABLE tablename | schema.tablename | SHARE | INTENT TO UPDATE | EXCLUSIVE | RESET
```

### Command parameters

#### TABLE

*tablename*

Specifies the unqualified table name. The table cannot be a system catalog table.

*schema.tablename*

Specifies the qualified table name. If *schema* is not provided, the CURRENT SCHEMA will be used. The table cannot be a system catalog table.



**SHARE** Specifies that the quiesce is to be in share mode.

When a "quiesce share" request is made, the transaction requests intent share locks for the table spaces and a share lock for the table. When the transaction obtains the locks, the state of the table spaces is changed to QUIESCED SHARE. The state is granted to the quiescer only if there is no conflicting state held by other users. The state of the table spaces, along with the authorization ID and the database agent ID of the quiescer, are recorded in the table space table, so that the state is persistent. The table cannot be changed while the table spaces for the table are in QUIESCED SHARE state. Other share mode requests to the table and table spaces are allowed. When the transaction commits or rolls back, the locks are released, but the table spaces for the table remain in QUIESCED SHARE state until the state is explicitly reset.

**INTENT TO UPDATE**

Specifies that the quiesce is to be in intent to update mode.

When a "quiesce intent to update" request is made, the table spaces are locked in intent exclusive (IX) mode, and the table is locked in update (U) mode. The state of the table spaces is recorded in the table space table.

**EXCLUSIVE**

Specifies that the quiesce is to be in exclusive mode.

When a "quiesce exclusive" request is made, the transaction requests super exclusive locks on the table spaces, and a super exclusive lock on the table. When the transaction obtains the locks, the state of the table spaces changes to QUIESCED EXCLUSIVE. The state of the table spaces, along with the authorization ID and the database agent ID of the quiescer, are recorded in the table space table. Since the table spaces are held in super exclusive mode, no other access to the table spaces is allowed. The user who invokes the quiesce function (the quiescer) has exclusive access to the table and the table spaces.

**RESET** Specifies that the state of the table spaces is to be reset to normal. A quiesce state cannot be reset if the connection that issued the quiesce request is still active.

When a quiescer issues a reset, only the quiesce mode for that quiescer is reset. If there are multiple quiescers, then the state of the table space will appear unchanged.

When working with a system-period temporal table and its associated history table, the reset operation must be performed on the same table that was used to originally set the quiesce mode.

### 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 in DB2 pureScale environments.

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).

## QUIESCE TABLESPACES FOR TABLE

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 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 alter 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.

When quiescing against a system-period temporal table, all the table spaces associated with the system-period temporal table and the history table are quiesced. When quiescing against a history table, all the table spaces associated with the history table, and the associated system-period temporal table are quiesced.

---

## 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 backend process or break a database connection. **CONNECT RESET** breaks a connection, but does not terminate the backend process. The **TERMINATE** command does both.

## REBIND

Allows the user to re-create a package stored in the database without the need for a bind file.

### Authorization

One of the following authorities:

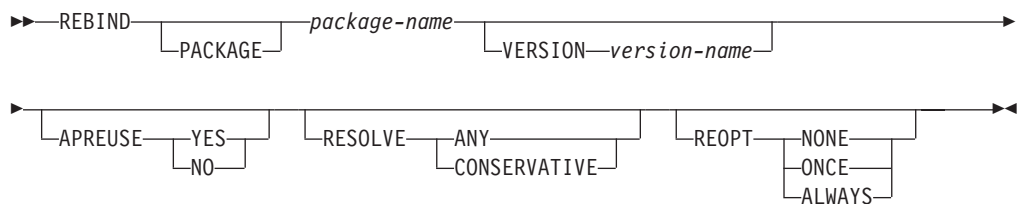
- DBADM authority
- ALTERIN privilege on the schema
- BIND privilege on the package.

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 objects that use the SQL path for resolution are considered during resolution on static DML statements in the package. This option is not supported by DRDA. Valid values are:

**ANY** All possible matches in the SQL path are considered for resolving references to any objects that use the SQL path for object resolution. Conservative binding semantics are not used. This is the default.

**CONSERVATIVE**

Only those objects in the SQL path that were defined before the last explicit bind time stamp are considered for resolving references to any objects that use the SQL path for object resolution. Conservative binding semantics are used. This option is not supported for an inoperative package.

**APREUSE**

Specifies whether static SQL access plans are to be reused. When this option is enabled, the query compiler will attempt to reuse the access plans for static SQL statements in the existing package during the rebind and during future implicit and explicit rebinds. The default is the value used during the previous invocation of the **BIND** or **REBIND** command or the **ALTER PACKAGE** statement. To determine the value, query the **APREUSE** column for the package in **SYSCAT.PACKAGES**.

**YES** The query compiler will attempt to reuse the access plans for the statements in the package.

**NO** The query compiler will not attempt to reuse access plans for the statements in the package.

**REOPT**

Specifies whether to have DB2 optimize an access path using values for host variables, parameter markers, global variables, and special registers.

**NONE**

The access path for a given SQL statement containing host variables, parameter markers, global variables, 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, global variables, 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 re-optimized using the values of the host variables, parameter markers, global variables, 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 re-create 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 re-create the

## REBIND

package. **REBIND** can also be used to re-create packages after **RUNSTATS** has been executed, thereby taking advantage of the new statistics.

- Provides a method to re-create 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 database upgrade, all packages stored in the database will be invalidated by the **UPGRADE DATABASE** command. 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 wants to modify any of the bind options as part of the rebind. **REBIND** does not support any bind options. For example, if the user wants 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 required. **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 re-creates 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** or **ALWAYS** might change static and dynamic statement compilation and performance.

If **REOPT** is not specified, **REBIND** will preserve the existing **REOPT** value used at **PRECOMPILE** or **BIND** time.

**Related information:**

## 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.

In a DB2 pureScale environment, the **RECOVER DATABASE** command can be issued from any member.

### Authorization

To recover an existing database, one of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT

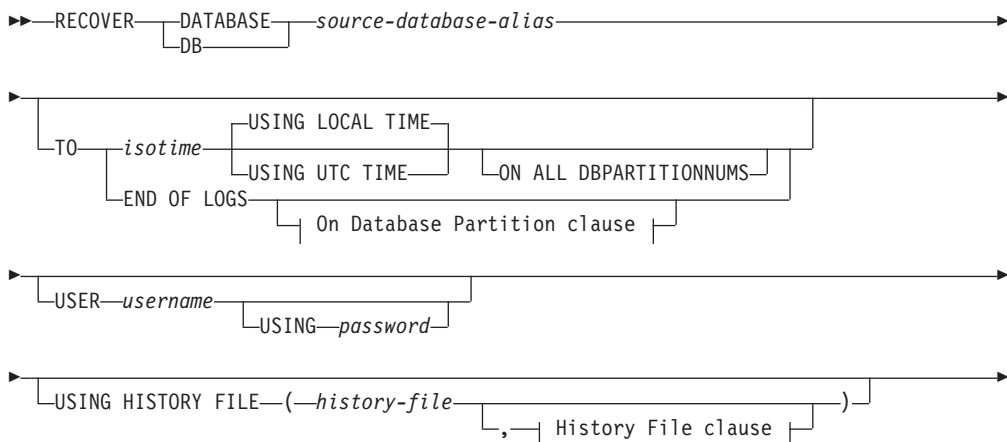
To recover to a new database, one of the following authorities:

- SYSADM
- SYSCTRL

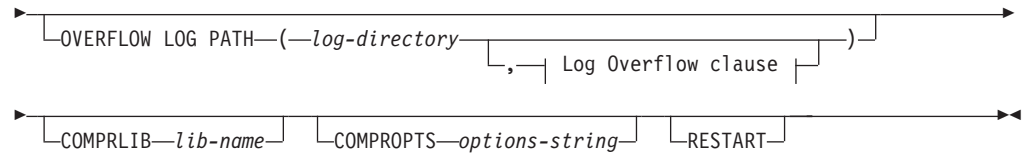
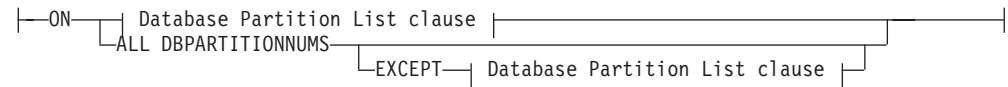
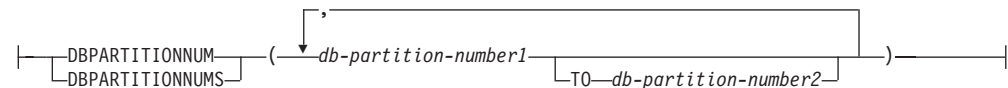
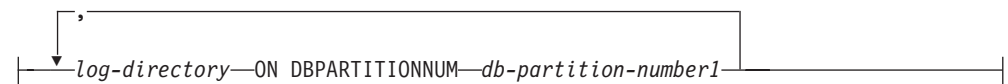
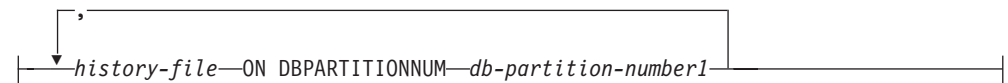
### Required connection

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





**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). A recover operation to a point in time returns a success message only if there is a transaction with a larger timestamp value in the log files. Even if there is no transaction with a larger timestamp, you can still issue a rollforward operation with the **COMPLETE** option.

This value is specified as a time stamp, a 7-part character string that identifies a combined date and time. The format is *yyyy-mm-dd-hh.mm.ss.nnnnnn* (year, month, day, hour, minutes, seconds, microseconds). The time stamp in a backup image is

## RECOVER DATABASE

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. This is the default option.

#### Note:

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.
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.
4. It is important to specify a valid timestamp when recovering a database. A valid timestamp would be the time that the last backup in the partitioned database system was completed.
5. When issuing multiple **RECOVER DATABASE** commands, the timestamp you specify for each subsequent command must be greater than the timestamp you specified in the previous command.

### USING UTC TIME

Specifies the point in time to which to recover.

### END OF LOGS

Specifies that all committed transactions from all logs in the path specified by `logpath`, plus logs on the highest-numbered log chain that can be retrieved from the locations specified by the `logarchmeth1` and `logarchmeth2` database configuration parameters, are to be rolled forward.

### 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.

**TO** *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 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. A subsequent **RECOVER DATABASE** command attempts to continue the previous recover operation, if possible. Using the **RESTART** keyword forces the recover operation to start with a fresh restore and then rollforward to the point in time 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

The following examples apply to a single-partition database environment or a DB2 pureScale environment, where the database being recovered currently exists, and the most recent version of the history file is available on the default database path specified in the database manager configuration file (**dftdbpath** parameter):

## RECOVER DATABASE

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 point in time, issue the following. The most recent image that can be used will be restored, and logs applied until the point in time 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 point in time 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 DATABASE** followed by a **ROLLFORWARD DATABASE** command. However, to use **RECOVER DATABASE** command, you would first have to extract the history file from the image to some location, for example `/home/user/fromimage/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 DATABASE**.)  

```
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 FILE** 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 2001-12-31-04.00.00
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 in the **dftdbpath** on all database partitions:

1. To recover the database to a point in time on all database partitions. DB2 database systems will verify that the PIT is reachable on all database partitions before starting any restore operations.  

```
RECOVER DB SAMPLE TO 2001-12-31-04.00.00
```
2. To recover the database to this point in time on all database partitions. DB2 database systems will verify that the specified point in time is reachable on all database partitions before starting any restore operations. The recover operation on each database partition is identical to a single-partition recovery  

```
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 database partitions 2

and 4. Database partition 2 will use: /home/user/node2files/db2rhist.asc, and database partition 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 database partitions instead of all database partitions, however a point-in-time recover operation cannot be done in this case; the recover must be done to the end of logs.

```
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 DATABASE** followed by a **ROLLFORWARD DATABASE**. However, to use **RECOVER DATABASE**, you would first have to extract the history file from the image to some location, for example, /home/user/fromimage/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 the recover.)

```
RECOVER DB SAMPLE TO 2001-12-31-04.00.00
  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 FILE** 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.
- In a DB2 pureScale environment, you must specify database partition 0 if you use the **RECOVER DATABASE** command with the **ON DBPARTITIONNUMS** clause or with the **ON DBPARTITIONNUM** clause (either as part of the database partition list clause or the log overflow clause).

## RECOVER DATABASE

Related information:

## REDISTRIBUTE DATABASE PARTITION GROUP

Redistributes data across the partitions in a database partition group. This command affects all objects present in the database partition group and cannot be restricted to one object alone.

This command can be issued only 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

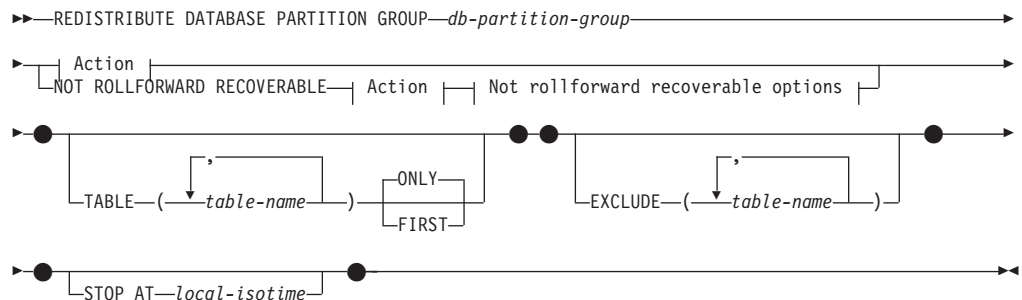
One of the following authorities is required:

- SYSADM
- SYSCTRL
- DBADM

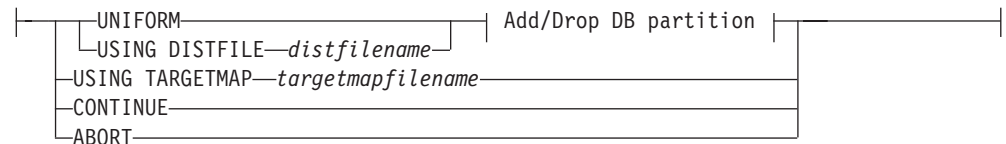
In addition, one of the following groups of authorizations is also required:

- DELETE, INSERT, and SELECT privileges on all tables in the database partition group being redistributed
- DATAACCESS authority

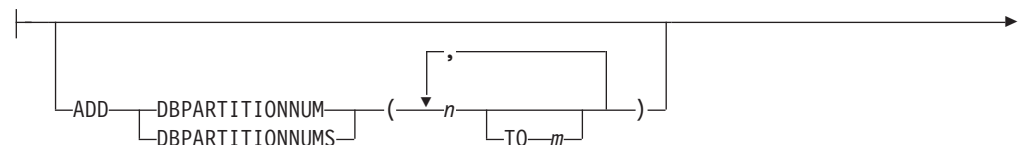
### Command syntax



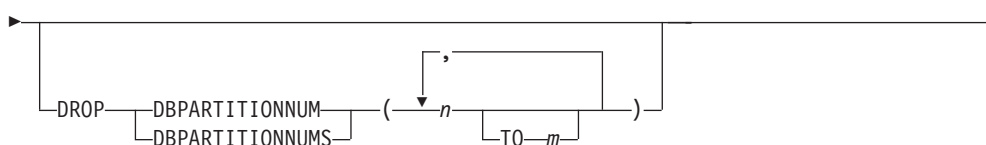
### Action:



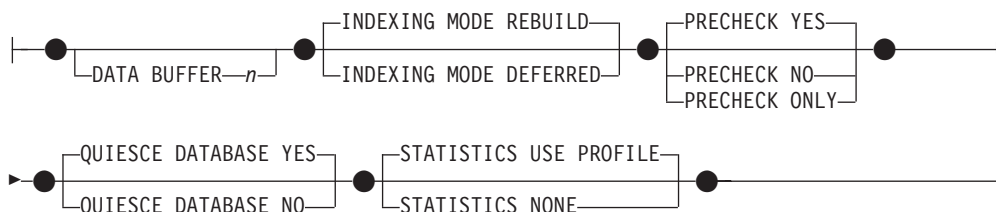
### Add/Drop DB partition:



## REDISTRIBUTE DATABASE PARTITION GROUP



### Not rollforward recoverable options:



### Command parameters

#### **DATABASE PARTITION GROUP** *db-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.

**Note:** Tables in the IBMCATGROUP and the IBMTEMPGROUP database partition groups cannot be redistributed.

#### **NOT ROLLFORWARD RECOVERABLE**

When this option is used, the **REDISTRIBUTE DATABASE PARTITION GROUP** command is not rollforward recoverable.

- Data is moved in bulk instead of by internal insert and delete operations. This reduces the number of times that a table must be scanned and accessed, which results in better performance.
- Log records are no longer required for each of the insert and delete operations. This means that you no longer need to manage large amounts of active log space and log archiving space in your system when performing data redistribution.
- When using the **REDISTRIBUTE DATABASE PARTITION GROUP** command with the **NOT ROLLFORWARD RECOVERABLE** option, the redistribute operation uses the **INDEXING MODE DEFERRED** option for tables that contain XML columns. If a table does not contain an XML column, the redistribute operation uses the indexing mode specified when issuing the command.

When this option is *not* used, extensive logging of all row movement is performed such that the database can be recovered later in the event of any interruptions, errors, or other business need.

#### **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.



## REDISTRIBUTE DATABASE PARTITION GROUP

### USING DISTFILE *distfilename*

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 *distfilename* to indicate the current distribution of data across the 32 768 hash partitions.

Use row counts, byte volumes, or any other measure to indicate the amount of data represented by each hash partition. The utility reads the integer value associated with a partition as the weight of that partition. When a *distfilename* 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 hash 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 112000, and partition 3 (with a weight of 0) has no data mapping to it at all.

The *distfilename* should contain 32 768 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 *distfilename* is not specified, the current directory is used.

### USING TARGETMAP *targetmapfilename*

The file specified in *targetmapfilename* 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.

The *targetmapfilename* should contain 32 768 integers, each representing a valid database partition number. The number on any row maps a hash value to a database partition. This means that if row *X* contains value *Y*, then every record with HASHEDVALUE() of *X* is to be located on database partition *Y*.

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 statement before running **REDISTRIBUTE DATABASE PARTITION GROUP** command.

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 statement either before or after the **REDISTRIBUTE DATABASE PARTITION GROUP** command.

### CONTINUE

Continues a previously failed or stopped **REDISTRIBUTE DATABASE PARTITION GROUP** operation. If none occurred, an error is returned.

### ABORT

Aborts a previously failed or stopped **REDISTRIBUTE DATABASE PARTITION GROUP** operation. If none occurred, an error is returned.

## REDISTRIBUTE DATABASE PARTITION GROUP

### ADD

**DBPARTITIONNUM** *n*

**TO** *m*

*n* or *n TO m* specifies a list or lists of database partition numbers which are to be added into the database partition group. Any specified partition must not already be defined in the database partition group (SQLSTATE 42728). This is equivalent to executing the ALTER DATABASE PARTITION GROUP statement with ADD DBPARTITIONNUM clause specified.

**DBPARTITIONNUMS** *n*

**TO** *m*

*n* or *n TO m* specifies a list or lists of database partition numbers which are to be added into the database partition group. Any specified partition must not already be defined in the database partition group (SQLSTATE 42728). This is equivalent to executing the ALTER DATABASE PARTITION GROUP statement with ADD DBPARTITIONNUM clause specified.

#### Note:

1. When a database partition is added using this option, containers for table spaces are based on the containers of the corresponding table space on the lowest numbered existing partition in the database partition group. If this would result in a naming conflict among containers, which could happen if the new partitions are on the same physical machine as existing containers, this option should not be used. Instead, the ALTER DATABASE PARTITION GROUP statement should be used with the WITHOUT TABLESPACES option before issuing the **REDISTRIBUTE DATABASE PARTITION GROUP** command. Table space containers can then be created manually specifying appropriate names.
2. Data redistribution might create table spaces for all new database partitions if the **ADD DBPARTITIONNUMS** parameter is specified.

### DROP

**DBPARTITIONNUM** *n*

**TO** *m*

*n* or *n TO m* specifies a list or lists of database partition numbers which are to be dropped from the database partition group. Any specified partition must already be defined in the database partition group (SQLSTATE 42729). This is equivalent to executing the ALTER DATABASE PARTITION GROUP statement with the DROP DBPARTITIONNUM clause specified.

**DBPARTITIONNUMS** *n*

**TO** *m*

*n* or *n TO m* specifies a list or lists of database partition numbers which are to be dropped from the database partition group. Any specified partition must already be defined in the database partition group (SQLSTATE 42729). This is equivalent to executing the ALTER DATABASE PARTITION GROUP statement with the DROP DBPARTITIONNUM clause specified.

**TABLE** *tablename*

Specifies a table order for redistribution processing.

## REDISTRIBUTE DATABASE PARTITION GROUP

### ONLY

If the table order is followed by the **ONLY** keyword (which is the default), then, only the specified tables will be redistributed. The remaining tables can be later processed by **REDISTRIBUTE CONTINUE** commands. This is the default.

### FIRST

If the table order is followed by the **FIRST** keyword, then, the specified tables will be redistributed with the given order and the remaining tables in the database partition group will be redistributed with random order.

### EXCLUDE *tablename*

Specifies tables to omit from redistribution processing. For example, you can temporarily omit a table until you can configure it to meet the requirements for data redistribution. The omitted tables can be later processed by **REDISTRIBUTE CONTINUE** commands.

### STOP AT *local-isotime*

When this option is specified, before beginning data redistribution for each table, the *local-isotime* is compared with the current local timestamp. If the specified *local-isotime* is equal to or earlier than the current local timestamp, the utility stops with a warning message. Data redistribution processing of tables in progress at the stop time will complete without interruption. No new data redistribution processing of tables begins. The unprocessed tables can be redistributed using the **CONTINUE** option. This *local-isotime* value is specified as a time stamp, a 7-part character string that identifies a combined date and time. The format is *yyyy-mm-dd-hh.mm.ss.nnnnnn* (year, month, day, hour, minutes, seconds, microseconds) expressed in local time.

### DATA BUFFER *n*

Specifies the number of 4 KB pages to use as buffered space for transferring data within the utility. This command parameter can be used only when the **NOT ROLLFORWARD RECOVERABLE** parameter is also specified.

If the value specified is lower than the minimum supported value, the minimum value is used and no warning is returned. If a **DATA BUFFER** value is not specified, an intelligent default is calculated by the utility at runtime at the beginning of processing each table. Specifically, the default is to use 50% of the memory available in the utility heap at the time redistribution of the table begins and to take into account various table properties as well.

This memory is allocated directly from the utility heap, whose size can be modified through the **util\_heap\_sz** database configuration parameter. The value of the **DATA BUFFER** parameter of the **REDISTRIBUTE DATABASE PARTITION GROUP** command can temporarily exceed **util\_heap\_sz** if more memory is available in the system.

### INDEXING MODE

Specifies how indexes are maintained during redistribution. This command parameter can be used only when the **NOT ROLLFORWARD RECOVERABLE** parameter is also specified.

Valid values are:

#### REBUILD

Indexes will be rebuilt from scratch. Indexes do not have to be valid to use this option. As a result of using this option, index pages will be clustered together on disk.

#### DEFERRED

Redistribute will not attempt to maintain any indexes. Indexes will be

## REDISTRIBUTE DATABASE PARTITION GROUP

marked as needing a refresh. The first access to such indexes might force a rebuild, or indexes might be rebuilt when the database is restarted.

**Note:** For non-MDC and non-ITC tables, if there are invalid indexes on the tables, the **REDISTRIBUTE DATABASE PARTITION GROUP** command automatically rebuilds them if you do not specify **INDEXING MODE DEFERRED**. For an MDC or ITC table, even if you specify **INDEXING MODE DEFERRED**, a composite index that is invalid is rebuilt before table redistribution begins because the utility needs the composite index to process an MDC or ITC table.

### PRECHECK

Verifies that the database partition group can be redistributed. This command parameter can be used only when the **NOT ROLLFORWARD RECOVERABLE** parameter is also specified.

#### YES

This is the default value. The redistribution operation begins only if the verification completes successfully. If the verification fails, the command terminates and returns an error message related to the first check that failed.

**NO** The redistribution operation begins immediately; no verification occurs.

#### ONLY

The command terminates after performing the verification; no redistribution occurs. By default it will not quiesce the database. If the **QUIESCE DATABASE** command parameter was set to YES or defaulted to a value of YES, the database remains quiesced. To restore connectivity to the database, perform the redistribution operation or issue **UNQUIESCE DATABASE** command.

### QUIESCE DATABASE

Specifies to force all users off the database and put it into a quiesced mode. This command parameter can be used only when the **NOT ROLLFORWARD RECOVERABLE** parameter is also specified.

#### YES

This is the default value. Only users with SYSADM, SYSMAINT, or SYSCTRL authority or users who have been granted QUIESCE\_CONNECT authority will be able to access the database or its objects. Once the redistribution completes successfully, the database is unquiesced.

**NO** The redistribution operation does not quiesce the database; no users are forced off the database.

For more information, refer to the **QUIESCE DATABASE** command.

### STATISTICS

Specifies that the utility should collect statistics for the tables that have a statistics profile. This command parameter can be used only when the **NOT ROLLFORWARD RECOVERABLE** parameter is also specified.

Specifying this option is more efficient than separately issuing the **RUNSTATS** command after the data redistribution is completed.

### USE PROFILE

Statistics will be collected for the tables with a statistics profile. For tables without a statistics profile, nothing will be done. This is the default.

## REDISTRIBUTE DATABASE PARTITION GROUP

### NONE

Statistics will not be collected for tables.

### Examples

Redistribute database partition group DBPG\_1 by providing the current data distribution through a data distribution file, `distfile_for_dbpg_1`. Move the data onto two new database partitions, 6 and 7.

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1
  USING DISTFILE /home/user1/data/distfile_for_dbpg_1
  ADD DATABASE PARTITION (6 TO 7)
```

Redistribute database partition group DBPG\_2 such that:

- The redistribution is not rollforward recoverable;
- Data is uniformly distributed across hash partitions;
- Indexes are rebuilt from scratch;
- Statistics are not collected;
- 180,000 4 KB pages are used as buffered space for transferring the data.

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_2
  NOT ROLLFORWARD RECOVERABLE
  UNIFORM
  INDEXING MODE REBUILD
  DATA BUFFER 180000
  STATISTICS NONE
```

This redistribution operation also quiesces the database and performs a precheck due to the default values for the **QUIESCE DATABASE** and **PRECHECK** command parameters.

### Usage notes

- Before starting a redistribute operation, ensure that the tables are in normal state and not in "load pending" state or "reorg pending" state. Table states can be checked by using the **LOAD QUERY** command.
- When the **NOT ROLLFORWARD RECOVERABLE** option is specified and the database is a recoverable database, the first time the utility accesses a table space, it is put into the **BACKUP PENDING** state. All the tables in that table space will become read-only until the table space is backed-up, which can only be done when all tables in the table space have finished being redistributed.
- When a redistribution operation is running, it produces an event log file containing general information about the redistribution operation and information such as the starting and ending time of each table processed. This event log file is written to:
  - The `homeinst/sqllib/redist` directory on Linux and UNIX operating systems, using the following format for subdirectories and file name:  
`database-name.database-partition-group-name.timestamp.log`.
  - The `DB2INSTPROF\instance\redist` directory on Windows operating systems (where **DB2INSTPROF** is the value of the **DB2INSTPROF** registry variable), using the following format for subdirectories and file name: `database-name.database-partition-group-name.timestamp.log`.
  - The time stamp value is the time when the command was issued.
- This utility performs intermittent **COMMITs** during processing.
- 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

## REDISTRIBUTE DATABASE PARTITION GROUP

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.

- By default, the redistribute utility will update the statistics for those tables that have a statistics profile. For the tables without a statistics profile, it is recommended that you separately update the table and index statistics for these tables by calling the db2Runstats API or by issuing the **RUNSTATS** command after the redistribute operation has completed.
- Database partition groups containing replicated materialized query 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 or created temporary tables in the database partition group.
- Options such as **INDEXING MODE** are ignored on tables, on which they do not apply, without warning. For example, **INDEXING MODE** will be ignored on tables without indexes.
- The **REDISTRIBUTE DATABASE PARTITION GROUP** command might fail (SQLSTATE 55071) if an add database partition server request is either pending or in progress. This command might also fail (SQLSTATE 55077) if a new database partition server is added online to the instance and not all applications are aware of the new database partition server.

### Compatibilities

Tables containing XML columns that use the DB2 Version 9.5 or earlier XML record format cannot be redistributed. Use the ADMIN\_MOVE\_TABLE stored procedure to migrate the table to the new format.

## REFRESH LDAP

Refreshes the cache on a local machine with updated information when the information in Lightweight Directory Access Protocol (LDAP) has been changed.

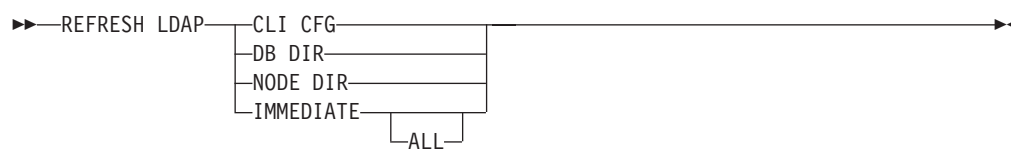
### Authorization

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.

#### IMMEDIATE

Specifies that the local database and node directories are to be refreshed immediately.

**ALL** Specifies that all database and node entries contained within the LDAP server are to be added into the local database and node directories.

### 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.

The **REFRESH LDAP IMMEDIATE** command updates entries from the local database and node directories using the latest information found in LDAP. This update

## REFRESH LDAP

occurs immediately and regardless if **DB2LDAPCACHE** is enabled or not. Only database and node entries that originated from LDAP will be updated. Entries that were added manually remain unchanged.

The **REFRESH LDAP IMMEDIATE ALL** command immediately populates the local database and node directories with all the information found in LDAP. If an entry found in LDAP matches an existing local entry, the command will update the entry. This update will only occur if the local entry originated from LDAP. Entries that were added manually remain unchanged. This update is performed regardless if **DB2LDAPCACHE** is enabled or not.

When LDAP is disabled, performing either **REFRESH LDAP IMMEDIATE** or **REFRESH LDAP IMMEDIATE ALL** will result in SQLCODE -3279 (The command did not complete successfully because LDAP is disabled).



## 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] nodename |
| AS |
▶ PROTOCOL [TCPIP] [HOSTNAME hostname] [SVCENAME svcename] [SECURITY SOCKS] |
| [TCPIP4] [HOSTNAME hostname] [SVCENAME svcename] [SECURITY SOCKS] |
| [TCPIP6] [HOSTNAME hostname] [SVCENAME svcename] |
| [NPIPE] |
▶ [REMOTE computer-name] [INSTANCE instance] [NODETYPE [SERVER] [MPP] [DCS]] [OSTYPE ostype] |
▶ [WITH "comment-string"] [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** *svcname*

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-name*

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** "*comment-string*"

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.

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.

## REGISTER XMLSCHEMA

Registers an XML schema with the XML schema repository (XSR).

### Authorization

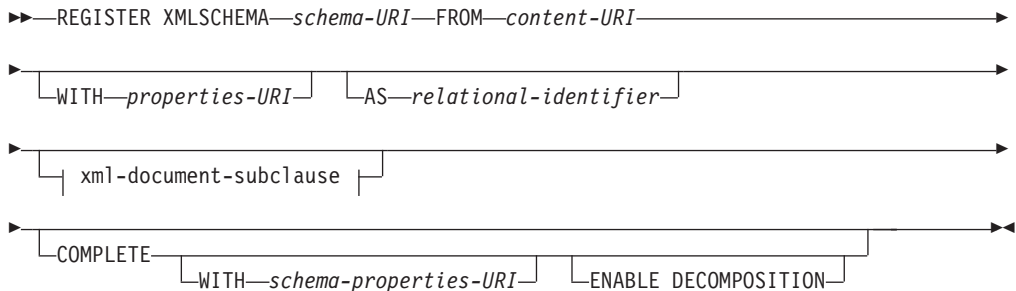
One of the following authorities:

- DBADM
- IMPLICIT\_SCHEMA database authority if the SQL schema does not exist
- CREATEIN privilege if the SQL schema exists

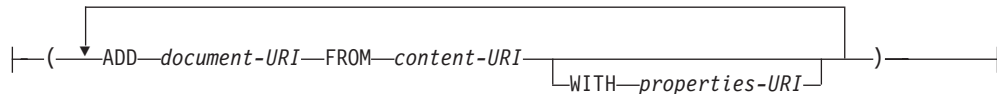
### Required connection

Database

### Command syntax



### xml-document-subclause:



### Command parameters

*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.

**Examples**

```
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.

## REGISTER XSROBJECT

Registers an XML object in the database catalogs. Supported objects are DTDs and external entities.

### Authorization

One of the following authorities:

- 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)>
]
<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)>
]
<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
```

## 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. On a data partitioned table, you can reorganize a specific nonpartitioned index on a partitioned table, or you can reorganize all the partitioned indexes on a specific data partition.

If you specify the **CLEANUP** option of the index clause, cleanup is performed without rebuilding the indexes. This command cannot be used against indexes on declared temporary tables or created temporary tables (SQLSTATE 42995).

The table option reorganizes a table by reconstructing the rows to eliminate fragmented data, and by compacting information. On a partitioned table, you can reorganize a single partition.

### Scope

This command affects all database partitions in the database partition group.

### Authorization

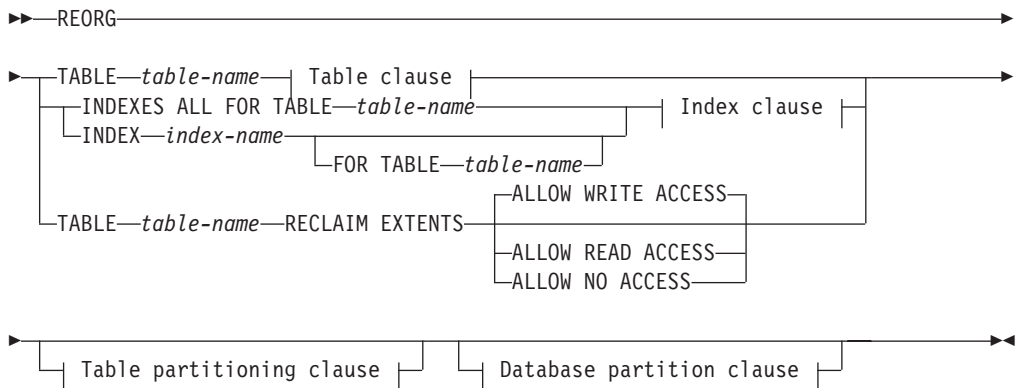
One of the following authorities:

- SYSADM
- SYSCtrl
- SYSMaint
- DBADM
- SQLADM
- CONTROL privilege on the table.

### Required connection

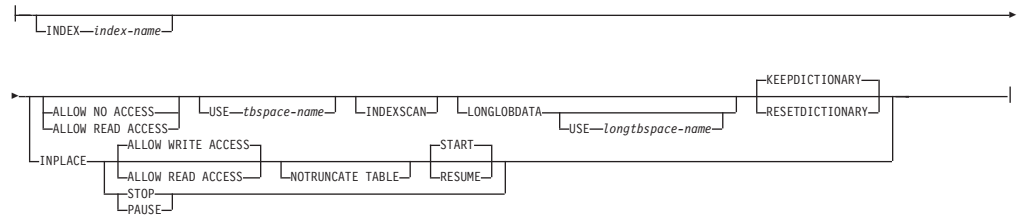
Database

### Command syntax

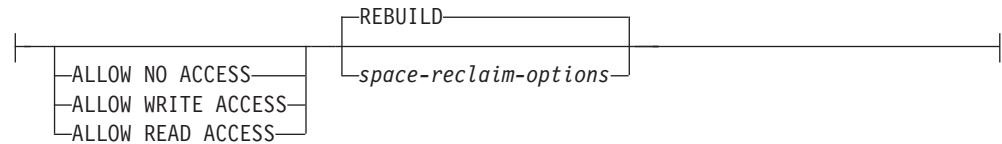


### Table clause:





**Index clause:**



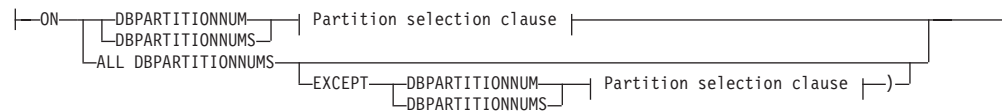
**space-reclaim-options:**



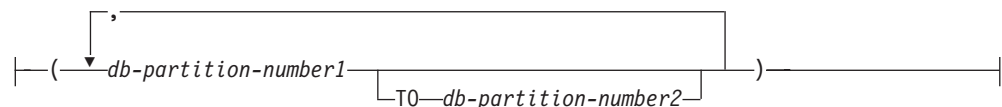
**Table partitioning clause:**



**Database partition clause:**



**Partition selection 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 data partitioned table. Reorganization of individual indexes are *only* supported for nonpartitioned indexes on a partitioned table. This parameter is not supported for block indexes.

## REORG INDEXES/TABLE

### FOR TABLE *table-name*

Specifies the name of the table on which the nonpartitioned index *index-name* is created. This parameter is optional, given that index names are unique across the database.

#### ALLOW NO ACCESS

For **REORG INDEXES**, specifies that no other users can access the table while the indexes are being reorganized. If the **ON DATA PARTITION** clause is specified for a partitioned table, only the specified partition is restricted to the access mode level.

For **REORG INDEX**, specifies that no other users can access the table while the nonpartitioned index is being reorganized.

#### ALLOW READ ACCESS

For **REORG INDEXES**, specifies that other users can have read-only access to the table while the indexes are being reorganized. **ALLOW READ ACCESS** mode is not supported for **REORG INDEXES** of a partitioned table unless the **CLEANUP** or **RECLAIM EXTENTS** option or the **ON DATA PARTITION** clause is specified. If the **ON DATA PARTITION** clause is specified for a partitioned table, only the specified partition is restricted to the access mode level.

For **REORG INDEX**, specifies that can have read-only access to the table while the nonpartitioned index is being reorganized.

#### ALLOW WRITE ACCESS

For **REORG INDEXES**, specifies that other users can read from and write to the table while the indexes are being reorganized. **ALLOW WRITE ACCESS** mode is not supported for a partitioned table unless the **CLEANUP** or **RECLAIM EXTENTS** option or the **ON DATA PARTITION** clause is specified. If the **ON DATA PARTITION** clause is specified for a partitioned table, only the specified partition is restricted to the access mode level.

For **REORG INDEX**, specifies that can read from and write to the table while the nonpartitioned index is being reorganized.

**ALLOW WRITE ACCESS** mode is not supported for multidimensional clustering (MDC) or insert time clustering (ITC) tables or extended indexes unless the **CLEANUP** or **RECLAIM EXTENTS** option is specified.

The following items apply for a data partitioned table when the **ON DATA PARTITION** clause is specified with the **REORG INDEXES ALL** command:

- Only the specified data partition is restricted to the access mode level. Users are allowed to read from and write to the other partitions of the table while the partitioned indexes of a specified partition are being reorganized.

The following table lists the access modes supported and the concurrent access allowed on other partitions of the table when the **ON DATA PARTITION** clause is specified:

Table 39. Access modes supported and concurrent access allowed when the **ON DATA PARTITION** clause is specified with **REORG INDEXES ALL**

Access mode	Concurrent access allowed on the specified partition	Concurrent access allowed on other partitions
<b>ALLOW NO ACCESS</b>	No access	Read and write access
<b>ALLOW READ ACCESS</b>	Read on the partition up until index is updated	Read and write access

Table 39. Access modes supported and concurrent access allowed when the ON DATA PARTITION clause is specified with REORG INDEXES ALL (continued)

Access mode	Concurrent access allowed on the specified partition	Concurrent access allowed on other partitions
<b>ALLOW WRITE ACCESS</b>	Read and write access on the partition up until index is updated	Read and write access

- Only the partitioned indexes for the specified partition are reorganized. The nonpartitioned indexes on the partitioned table are not reorganized. If there are any nonpartitioned indexes on the table marked "invalid" or "for rebuild", these indexes are rebuilt before reorganization. If not, only the partitioned indexes on the specified partition are reorganized or rebuilt if the index object is marked "invalid" or "for rebuild".
- Only partitioned indexes for the specified partition are cleaned when the **CLEANUP** or **RECLAIM EXTENTS** option is also specified.

The following table lists the supported access modes for index reorganization of partitioned and nonpartitioned tables:

Table 40. Supported access modes for index reorganization on partitioned and nonpartitioned table

Command	Table type	Table partitioning clause	Additional parameters specified for index clause	Supported access mode
<b>REORG INDEXES</b>	Nonpartitioned table	Not applicable	Any	<b>ALLOW NO ACCESS, ALLOW READ ACCESS<sup>1</sup>, ALLOW WRITE ACCESS</b>
<b>REORG INDEX</b>	Partitioned table	Not applicable	Any	<b>ALLOW READ ACCESS<sup>1</sup></b>
<b>REORG INDEXES</b>	Partitioned table	None	<b>REBUILD</b> (this is the default if none specified)	<b>ALLOW NO ACCESS<sup>1</sup></b>
<b>REORG INDEXES</b>	Partitioned table	<b>ON DATA PARTITION</b>	<b>REBUILD</b> (this is the default if none specified)	<b>ALLOW NO ACCESS, ALLOW READ ACCESS<sup>1</sup>, ALLOW WRITE ACCESS</b>
<b>REORG INDEXES</b>	Partitioned table	With or without the <b>ON DATA PARTITION</b> clause	<b>CLEANUP</b> or <b>RECLAIM EXTENTS</b> specified	<b>ALLOW NO ACCESS, ALLOW READ ACCESS<sup>1</sup>, ALLOW WRITE ACCESS</b>

**Note:**

1. Default mode when an access clause is not specified.

**CLEANUP**

When **CLEANUP** is requested, a cleanup rather than a **REBUILD** is done. The indexes are not rebuilt and any pages freed up are available for reuse by indexes defined on this table only.

**ALL**

Specifies that indexes should be cleaned up by removing committed pseudo deleted keys and committed pseudo empty pages.

The **CLEANUP 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.

**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.

The **CLEANUP 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.

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. No access to the table is allowed when rebuilding an index during the period in which the reorganized copies of the indexes are made available.

### **INDEX** *index-name* **REBUILD**

The **REBUILD** option is the default and represents the same functionality provided by index reorganization in previous releases when the **CLEANUP** and **CONVERT** clauses were not specified. The **REBUILD** option of index reorganization rebuilds the index data into physically contiguous pages. The default access mode is dependent on the table type.

### **INDEX** *index-name* **RECLAIM EXTENTS**

Specifies the index to reorganize and reclaim extents that are not being used. This action moves index pages around within the index object to create empty extents, and then free these empty extents from exclusive use by the index object and makes the space available for use by other database objects within the table space. Extents are reclaimed from the index object back to the table space. **ALLOW READ ACCESS** is the default, however all access modes are supported.

### **TABLE** *table-name* **RECLAIM EXTENTS**

Specifies the table to reorganize and reclaim extents that are not being used. The *table-name* variable must specify a multidimensional clustering (MDC) or insert time clustering (ITC) table. 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 **REORG TABLE RECLAIM EXTENTS** when the **ON DATA PARTITION** clause is specified, the access clause only applies to the named partition. Users can

read from and write to the rest of the table while the extents on the specified partition are being reclaimed. This situation also applies to the default access levels.

**ALLOW NO ACCESS**

For **REORG TABLE RECLAIM EXTENTS**, specifies that no other users can access the table while the extents are being reclaimed.

**ALLOW READ ACCESS**

For **REORG TABLE RECLAIM EXTENTS**, specifies that other users can have read-only access to the table while the extents are being reclaimed.

**ALLOW WRITE ACCESS**

For **REORG TABLE RECLAIM EXTENTS**, specifies that other users can read from and write to the table while the extents are being reclaimed.

**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) or insert time clustering (ITC) table. In place reorganization of tables cannot be used for MDC or ITC tables.

When the **ON DATA PARTITION** clause is specified for a table reorganization of a data partitioned table, only the specified data partition is reorganized:

- If there are no nonpartitioned indexes (except system-generated XML path indexes) defined on the table, the access mode applies only to the specified partition, users are allowed to read from and write to the other partitions of the table.
- If there are nonpartitioned indexes defined on the table (excluding system-generated XML path indexes), the **ALLOW NO ACCESS** mode is the default and only supported access mode. In this case, the table is placed in **ALLOW NO ACCESS** mode. If **ALLOW READ ACCESS** is specified, SQL1548N is returned (SQLSTATE 5U047).

Table 41. Supported access mode for table reorganization on nonpartitioned and partitioned table

Command	Table type	Table partitioning clause	Supported access mode
<b>REORG TABLE</b>	Nonpartitioned table	Not applicable	<b>ALLOW NO ACCESS, ALLOW READ ACCESS<sup>1</sup></b>
<b>REORG TABLE</b>	Partitioned table	Not specified	<b>ALLOW NO ACCESS<sup>1</sup></b>
<b>REORG TABLE</b> (There are no indexes or only partitioned indexes defined on the table.)	Partitioned table	<b>ON DATA PARTITION</b>	<b>ALLOW NO ACCESS, ALLOW READ ACCESS<sup>1</sup></b>
<b>REORG TABLE</b> (there are nonpartitioned indexes defined on the table, excluding system-generated XML path indexes.)	Partitioned table	<b>ON DATA PARTITION</b>	<b>ALLOW NO ACCESS<sup>1</sup></b>

### Note:

1. Default mode when an access clause is not specified.

For a data partitioned table, a table reorganization rebuilds the nonpartitioned indexes and partitioned indexes on the table after reorganizing the table. If the **ON DATA PARTITION** clause is used to reorganize a specific data partition of a data partitioned table, a table reorganization rebuilds the nonpartitioned indexes and partitioned indexes only for the specified partition.

### **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 the 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 or ITC table.

If a table reorganization uses both the **INDEX** and **ON DATA PARTITION** clauses, only the specified partition is reorganized using the index *index-name*.

### **ALLOW NO ACCESS**

Specifies that no other users can access the table while the table is being reorganized.

The **ALLOW NO ACCESS** mode is the default and only supported access mode when reorganizing a partitioned table without the **ON DATA PARTITION** clause.

If the **ON DATA PARTITION** clause is specified for a data partitioned table, only the specified data partition is reorganized:

- If there are no nonpartitioned indexes defined on the table (except system-generated XML path indexes), only the specified partition is restricted to the **ALLOW NO ACCESS** mode. Users are allowed to read from and write to the other partitions of the table.
- If there are nonpartitioned indexes defined on the table (except system-generated XML path indexes), the **ALLOW NO ACCESS** mode is the default and only supported access mode. In this case, the table is placed in **ALLOW NO ACCESS** mode.

### **ALLOW READ ACCESS**

Allow only read access to the table during reorganization.

The **ALLOW READ ACCESS** mode is the default mode for a nonpartitioned table.

If the **ON DATA PARTITION** clause is specified for a data partitioned table, only the specified data partition is reorganized:

- If there are no nonpartitioned indexes defined on the table (except system-generated XML path indexes), the **ALLOW READ**

**ACCESS** mode is the default mode and only the specified partition is restricted to the access mode level. Users are allowed to read from and write to the other partitions of the table.

- If there are nonpartitioned indexes defined on the table (except system-generated XML path indexes), the **ALLOW READ ACCESS** mode is not supported. If **ALLOW READ ACCESS** is specified in this case, SQL1548N is returned (SQLSTATE 5U047)

### **INPLACE**

Reorganizes the table while permitting user access.

In place table reorganization is allowed only on nonpartitioned, non-MDC, and non-ITC tables without extended indexes and with no indexes defined over XML columns in the table. In place table reorganization can only be performed on tables that are at least three pages in size.

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** *tblspace-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 8 KB, 16 KB, or 32 KB 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.

For partitioned tables, the temporary table space is used as temporary storage for the reorganization of data partitions in the table. Reorganization of the entire partitioned table reorganizes a single data partition at a time. The temporary table space must be

## REORG INDEXES/TABLE

able to hold the largest data partition in the table, and not the entire table. When the **ON DATA PARTITION** clause is specified, the temporary table space must be able to hold the specified partition.

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. However, running a reorganization with the **LONGLOBDATA** option on tables with XML columns will reclaim unused space and thereby reduce the size of the XML storage object.

This parameter is required when converting existing LOB data into inlined LOB data.

### USE *longtbspace-name*

This is an optional parameter, which can be used to specify the name of a temporary table space to be used for rebuilding long data. If no temporary table space is specified for either the table object or for the long objects, the objects will be constructed in the table space they currently reside. If a temporary table space is specified for the table but this parameter is not specified, then the table space 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 *longtbspace-name*** is specified, **USE *tbspace-name*** must also be specified. If it is not, the *longtbspace-name* 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 is **YES** and a compression dictionary doesn't exist for the table, a dictionary will only be created (and the table compressed) in this scenario if the table is of a certain size (approximately 1 to 2 MB) and sufficient data exists within this table. If, instead, you explicitly state **REORG RESETDICTIONARY**, then a dictionary is built as long as there is at least 1 row in the table. If the **COMPRESS** attribute for the table is **NO** and the table has a compression dictionary, then reorg processing will preserve the dictionary and all the rows in the newly



reorganized table will be in noncompressed format. It is not possible to compress some data such as LOB data not stored in the base table row.

When the **LONGLOBDATA** option is not specified, only the table row data is reorganized. The following table describes the behavior of **KEEPDICTIONARY** syntax in **REORG** command when the **LONGLOBDATA** option is not specified.

Table 42. 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

The following table describes the behavior of **KEEPDICTIONARY** syntax in **REORG** command when the **LONGLOBDATA** option is specified.

Table 43. REORG KEEPDICTIONARY when LONGLOBDATA option is specified.

Compress	Table row data dictionary exists	XML storage object dictionary exists <sup>1</sup>	Compression dictionary	Data compression
Y	Y	Y	Preserve dictionaries.	Existing data is compressed. New data will be compressed.
Y	Y	N	Preserve table row dictionary and create an XML storage object dictionary.	Existing data is compressed. New data will be compressed.
Y	N	Y	Create table row dictionary and preserve the XML dictionary.	Existing data is compressed. New data will be compressed.
Y	N	N	Create table row and XML dictionaries.	Existing data is compressed. New data will be compressed.
N	Y	Y	Preserve table row and XML dictionaries.	Table data is uncompressed. New data will be not be compressed.
N	Y	N	Preserve table row dictionary.	Table data is uncompressed. New data will be not be compressed.
N	N	Y	Preserve XML dictionary.	Table data is uncompressed. New data will be not be compressed.

## REORG INDEXES/TABLE

Table 43. REORG KEEPDICTIONARY when LONGLOBDATA option is specified. (continued)

Compress	Table row data dictionary exists	XML storage object dictionary exists <sup>1</sup>	Compression dictionary	Data compression
N	N	N	No effect.	Table data is uncompressed. New data will be not be compressed.

### Note:

1. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in DB2 V9.7 or later, or if the table is migrated using the ONLINE\_TABLE\_MOVE stored procedure.

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 (that is, 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 noncompressed format. It is not possible to compress some data such as LOB data not stored in the base table row.

If the **LONGLOBDATA** option is not specified, only the table row data is reorganized. The following table describes the behavior of **RESETDICTIONARY** syntax in **REORG** command when the **LONGLOBDATA** option is not specified.

Table 44. REORG RESETDICTIONARY

Compress	Dictionary Exists	Result; outcome
Y	Y	Build new dictionary*; rows compressed. If DATA CAPTURE CHANGES option is specified on the CREATE TABLE or ALTER TABLE statements, the current dictionary is kept (referred to as the <i>historical compression dictionary</i> ).
Y	N	Build new dictionary; rows compressed
N	Y	Remove dictionary; all rows uncompressed. If the DATA CAPTURE NONE option is specified on the CREATE TABLE or ALTER TABLE statements, the <i>historical compression dictionary</i> is also removed for the specified table.
N	N	No effect; all rows uncompressed

\* - If a dictionary exists and the compression attribute is enabled but there currently isn't any data in the table, 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.

The following table describes the behavior of **RESETDICTIONARY** syntax in **REORG** command when the **LONGLOBDATA** option is specified.

Table 45. REORG RESETDICTIONARY when LONGLOBDATA option is specified.

Compress	Table row data dictionary exists	XML storage object dictionary exists <sup>1</sup>	Data dictionary	Data compression
Y	Y	Y	Build dictionaries <sup>2 3</sup> .	Existing data is compressed. New data will be compressed.
Y	Y	N	Build new table row dictionary and create a new XML dictionary <sup>3</sup> .	Existing data is compressed. New data will be compressed.
Y	N	Y	Create table row data dictionary and build a new XML dictionary.	Existing data is compressed. New data will be compressed.
Y	N	N	Create dictionaries.	Existing data is compressed. New data will be compressed.
N	Y	Y	Remove dictionaries. Existing and new data is not compressed.	Existing table data is uncompressed. New data will be not be compressed.
N	Y	N	Remove table row dictionary. All data is uncompressed.	Existing table data is uncompressed. New data will be not be compressed.
N	N	Y	Remove XML storage object dictionary.	Existing table data is uncompressed. New data will be not be compressed.
N	N	N	No effect.	Existing table data is uncompressed. New data will be not be compressed.

**Note:**

1. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in DB2 V9.7 or later, or if the table is migrated using an online table move.
2. If a dictionary exists and the compression attribute is enabled but there currently isn't any data in the table, the **RESETDICTIONARY** operation will keep the existing dictionary.

## REORG INDEXES/TABLE

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.

3. If DATA CAPTURE CHANGES option is specified on the CREATE TABLE or ALTER TABLE statements, the current data dictionary is kept (referred to as the *historical compression dictionary*).

### ON DATA PARTITION *partition-name*

For data partitioned tables, specifies the data partition for the reorganization.

For DB2 V9.7 Fix Pack 1 and later releases, the clause can be used with the **REORG INDEXES ALL** command to reorganize the partitioned indexes on a specific partition and the **REORG TABLE** command to reorganize data of a specific partition.

When using the clause with a **REORG TABLE** or **REORG INDEXES ALL** command on a partitioned table, the reorganization fails and returns SQL2222N with reason code 1 if the partition *partition-name* does not exist for the specified table. The reorganization fails and returns SQL2222N with reason code 3 if the partition *partition-name* is in the attached or detached state.

If the **REORG INDEX** command is issued with the **ON DATA PARTITION** clause, the reorganization fails and returns SQL2222N with reason code 2.

The **REORG TABLE** command fails and returns SQL1549N (SQLSTATE 5U047) if the partitioned table is in the reorg pending state and there are nonpartitioned indexes defined on the table.

### 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 database partition 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 database partition 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.

## 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 partition group consisting of database partitions 1, 2, 3, and 4 of a four-partition 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 tempSPACE1
```

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 or created 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.
- Concurrent table reorganization sharing the same temporary DMS table space is not supported.
- Before running a reorganization operation against a table to which event monitors write, you need to deactivate the event monitors on that table.
- For data partitioned tables:
  - 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 the **ON DATA PARTITION** clause is specified, that partition must be fully accessible.

## REORG INDEXES/TABLE

- If an error occurs during table reorganization, some indexes or index partitions might be left invalid. The nonpartitioned indexes of the table will be marked invalid if the reorganization has reached or passed the replace phase for the first data partition. The index partitions for any data partition that has already reached or passed the replace phase will be marked invalid. Indexes will be rebuilt on the next access to the table or data partition.
- If an error occurs during index reorganization when the **ALLOW NO ACCESS** mode is used, some indexes on the table might be left invalid. For nonpartitioned RID indexes on the table, only the index that is being reorganized at the time of the failure will be left invalid. For MDC tables with nonpartitioned block indexes, one or more of the block indexes might be left invalid if an error occurs. For MDC or ITC tables with partitioned indexes, only the index object on the data partition being reorganized will be left invalid. Any indexes marked invalid will be rebuilt on the next access to the table or data partition.
- When a data partitioned table with only partitioned indexes defined on the table is in the reorg pending state, issuing a **REORG TABLE** command with the **ON DATA PARTITION** clause brings only the specified data partition out of the reorg pending state. To bring the remaining partitions of the table out of the reorg pending state, either issue **REORG TABLE** command on the entire table (without the **ON DATA PARTITION** clause), or issue a **REORG TABLE** command with the **ON DATA PARTITION** clause for each of the remaining partitions.

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 inplace 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** with the **REBUILD** option can be used to reorganize one or all of the indexes on a table. Index reorganization rebuild will remove any fragmentation and restore physical clustering to the leaf pages. Use the **REORGCHK** command 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**.

A classic table reorganization (offline reorganization) rebuilds the indexes during the last phase of the reorganization. When more than one temporary table space exists, it is possible that a temporary table space in addition to the one specified on the **REORG TABLE** command may be utilized for additional sorts that can accompany table reorg processing. However, the inplace table reorganization (online reorganization) does not rebuild the indexes. It is recommended that you issue a **REORG INDEXES** command after the completion of an inplace table reorganization. An inplace table reorganization is asynchronous, therefore care

must be taken to ensure that the inplace table reorganization is complete before issuing the **REORG INDEXES** command. Issuing the **REORG INDEXES** command before the inplace table reorganization is complete, might cause the reorganization to fail (SQLCODE -2219).

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 the **REORGCHK** command 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.

With DB2 V9.7 Fix Pack 1 and later, **REORG TABLE** commands and **REORG INDEXES ALL** commands can be issued on a data partitioned table to concurrently reorganize different data partitions or partitioned indexes on a partition. When concurrently reorganizing data partitions or the partitioned indexes on a partition, users can access the unaffected partitions but cannot access the affected partitions. All the following criteria must be met to issue **REORG** commands that operate concurrently on the same table:

- Each **REORG** command must specify a different partition with the **ON DATA PARTITION** clause.
- Each **REORG** command must use the **ALLOW NO ACCESS** mode restrict access to the data partitions.
- The partitioned table must have only partitioned indexes if issuing **REORG TABLE** commands. No nonpartitioned indexes (except system-generated XML path indexes) can be defined on the table.

For a partitioned table T1 with no nonpartitioned indexes (except system-generated XML path indexes) and with partitions P1, P2, P3, and P4, the following **REORG** commands can run concurrently:

```
REORG INDEXES ALL FOR TABLE T1 ALLOW NO ACCESS ON DATA PARTITION P1
REORG TABLE T1 ALLOW NO ACCESS ON DATA PARTITION P2
REORG INDEXES ALL FOR TABLE T1 ALLOW NO ACCESS ON DATA PARTITION P3
```

Operations such as the following are not supported when using concurrent **REORG** commands:

- Using a **REORG** command without the **ON DATA PARTITION** clause on the table.
- Using an **ALTER TABLE** statement on the table to add, attach, or detach a data partition.
- Loading data into the table.
- Performing an online backup that includes the table.

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.

## REORG INDEXES/TABLE

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 rollforward recovery following a table reorganization, both regular and large table spaces must be enabled for rollforward 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).

Indexes over XML data may be re-created by the **REORG INDEXES/TABLE** command. For details, see "Recreation of indexes over XML data".

**Related information:**



## 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

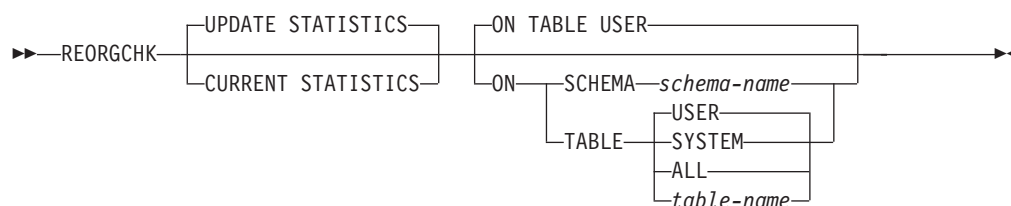
One of the following authorities:

- 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 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 current schema is assumed. 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) or insert time clustering (ITC) 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 multidimensional clustering (MDC) or insert time clustering (ITC) 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 or ITC table. An "\*" suffix to an index name indicates it is an MDC or ITC 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). Depending on whether the index is partitioned, this value comes from the NLEAF column of SYSCAT.INDEXES or SYSCAT.SYSINDEXPARTITIONS.

#### **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 following table 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 following table 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 following table for easy reference.

Table 46. LEAF\_RECSIZE\_OVERHEAD, NLEAF\_RECSIZE\_OVERHEAD, and DUPKEYSIZE values are a function of index type, table partitioning, and table space type (REGULAR table space)

Variable	Regular Table (XML paths or regions index)	Regular Table (All other indexes)	Partitioned Table (All indexes)
LEAF_RECSIZE_OVERHEAD	9	7	9
NLEAF_RECSIZE_OVERHEAD	12	10	12
DUPKEYSIZE	7	5	7

Table 47. LEAF\_RECSIZE\_OVERHEAD, NLEAF\_RECSIZE\_OVERHEAD, and DUPKEYSIZE values are a function of index type, table partitioning, and table space type (LARGE table space\*\*)

Variable	Regular Table (XML paths or regions index)	Regular Table (All other indexes)	Partitioned Table (All indexes)
LEAF_RECSIZE_OVERHEAD	9	9	11
NLEAF_RECSIZE_OVERHEAD	12	12	14
DUPKEYSIZE	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.
- F6 Results of Formula 6. The notation +++ indicates that the result exceeds 999, and might be invalid. Rerun **REORGCHK** with the **UPDATE STATISTICS** option, or issue **RUNSTATS**, followed by the **REORGCHK** command. If the statistics are current and valid, you should reorganize.
- 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** 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 PAGES** option of index reorganization is recommended.
- The **RECLAIMABLE\_SPACE** column in the **admin\_get\_tab\_info** and **admin\_get\_index\_info** functions show the amount of reclaimable space, in kilobytes, for tables and indexes. You can use this value to determine when to run a reorganization with the **RECLAIM EXTENTS** option. For more details on **RECLAIMABLE\_SPACE**, and how to evaluate the effectiveness of space reclaim, see the related links.

On a partitioned table the results for formulas (5 to 8) can be misleading depending on when the statistics are collected. When data partitions are detached, the index keys for the detached partition are not cleaned up immediately. Instead, the cleanup is deferred and eventually the keys are removed by index cleaners which operate asynchronously in the background (this is known as Asynchronous Index Cleanup or AIC). While the index keys pending cleanup exist in the index, they will not be counted as part of the keys in the statistics because they are invisible and no longer part of the table. 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 SYSCAT.DATAPARTITIONS catalog view and look for the value I (index cleanup), L (logically detached), or D (detached with dependent MQT).

### Usage notes

This command does not display statistical information for declared temporary tables or created temporary tables.

This utility does not support the use of nicknames.

A new server release might introduce new table and index features. These new features might impact **REORGCHK** logic, that is, how **REORGCHK** computes **REORG** recommendations. If **REORGCHK** is issued from a client not at the same level as the server, it might report different results than those reported from a client at the same level as the server. **REORGCHK** is a client application, therefore, **REORGCHK** should be run from a client running the same level as the server. Doing so ensures the most accurate report is generated. For server administrative work, in general, use a client and a server at the same level.

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 (  $\text{NPARTITIONS} * 1 \text{ extent size}$  ) of pages, no table reorganization is recommended based on each formula. More specifically,

- For non-partitioned tables (  $\text{NPARTITIONS} = 1$  ), the threshold is:  
(  $\text{FPAGES} \leq 1 \text{ extent size}$  )
- For partitioned tables, it is:  
(  $\text{FPAGES} \leq \text{NPARTITIONS} * 1 \text{ extent size}$  )

- In a multi-partitioned database, after the number of database partitions in a database partition group of the table is considered, this threshold for not recommending table reorganization changes to:

$FPAGES \leq \text{'number of database partitions in a database partition group of the table'} * NPARTITIONS * 1 \text{ extent size}$

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:

$100 * OVERFLOW / CARD < 5$

The total number of overflow rows in the table should be less than 5 percent of the total number of rows. Overflow rows can be created when rows are updated and the new rows contain more bytes than the old ones (VARCHAR fields), or when columns are added to existing tables.

- Formula F2:

For regular tables:

$100 * TSIZE / ((100 - TPCTFREE) / 100 * (FPAGES - NPARTITIONS) * (TABLEPAGESIZE - 68)) > 70$

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).

For MDC tables:

$100 * TSIZE / ((ACTBLK - FULLKEYCARD) * EXTENTSIZE * (TABLEPAGESIZE - 68)) > 70$

FULLKEYCARD represents the cardinality of the composite dimension index for the MDC table. Extentsize 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 previously noted, 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 or ITC tables, the formula is:

$100 * \text{activeblocks} / (((\text{fpages\_adjust} / \text{tExtentSize}) - (\text{numberOfTablePartitions} * \text{numberOfDatabasePartitions}))$

**REORGCHK** uses the following formulas to analyze the indexes and their the relationship to the table data:

- Formula F4:

– For non-partitioned tables:

$CLUSTERRATIO \text{ or } \text{normalized CLUSTERFACTOR} > 80$

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:  
 AVGPARTITION\_CLUSTERRATIO or normalized AVGPARTITION\_CLUSTERFACTOR > 80  
 AVGPARTITION\_CLUSTERFACTOR and AVGPARTITION\_CLUSTERRATIO 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.

**Note:** If readahead prefetching is enabled, formula F4 might not be a good indicator for issuing the **REORG** command. It is more appropriate to rely on the performance of the query when deciding on whether to issue the **REORG** command.

- Formula F5:
  - For a single database partition:  

$$\frac{100 * (\text{KEYS} * (\text{LEAF\_RECSIZE} + \text{LEAF\_RECSIZE\_OVERHEAD}) + (\text{INDCARD} - \text{KEYS}) * \text{DUPKEYSIZE})}{((\text{LEAF} - \text{NUM\_EMPTY\_LEAFS} - 1) * (\text{INDEXPAGESIZE} - \text{LEAF\_PAGE\_OVERHEAD}))}$$

$$> \text{MIN}(50, (100 - \text{PCTFREE}))$$

The space in use at the leaf level of the index should be greater than the minimum of 50 and 100-PCTFREE percent (only checked when LEAF > 1).
  - For a multi-partition database environment:  

$$\frac{100 * (\text{KEYS} * (\text{LEAF\_RECSIZE} + \text{LEAF\_RECSIZE\_OVERHEAD}) + (\text{INDCARD} - \text{KEYS}) * \text{DUPKEYSIZE})}{((\text{LEAF} - \text{NUM\_EMPTY\_LEAFS} - \text{NPARTITIONS}) * (\text{INDEXPAGESIZE} - \text{LEAF\_PAGE\_OVERHEAD}))}$$

$$> \text{MIN}(50, (100 - \text{PCTFREE}))$$

- Formula F6:  

$$\left( \frac{(100 - \text{PCTFREE}) * ((\text{FLOOR}((100 - \text{LEVEL2PCTFREE}) / 100 * (\text{INDEXPAGESIZE} - \text{NLEAF\_PAGE\_OVERHEAD}) / (\text{NLEAF\_RECSIZE} + \text{NLEAF\_RECSIZE\_OVERHEAD}))) * (\text{FLOOR}((100 - \text{MIN}(10, \text{LEVEL2PCTFREE})) / 100 * (\text{INDEXPAGESIZE} - \text{NLEAF\_PAGE\_OVERHEAD}) / (\text{NLEAF\_RECSIZE} + \text{NLEAF\_RECSIZE\_OVERHEAD}))) ** (\text{NLEVELS} - 3)) * (\text{INDEXPAGESIZE} - \text{LEAF\_PAGE\_OVERHEAD})}{(\text{KEYS} * (\text{LEAF\_RECSIZE} + \text{LEAF\_RECSIZE\_OVERHEAD}) + (\text{INDCARD} - \text{KEYS}) * \text{DUPKEYSIZE})} \right) < 100$$

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 in the following form:

$$\frac{\text{Amount of space needed for an index if it was one level smaller}}{\text{Amount of space needed for all the entries in the index}} < 1$$



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) =

$$\left\{ \text{FLOOR} \left( \frac{(100 - \text{LEVEL2PCTFREE})}{100} * \frac{(\text{PageSize} - \text{Overhead})}{(\text{Avg. size of each nonleaf index entry})} \right) * \text{FLOOR} \left( \frac{(100 - \text{MIN}(10, \text{LEVEL2PCTFREE}))}{100} * \frac{(\text{PageSize} - \text{Overhead})}{(\text{Avg. size of each nonleaf index entry})} \right) \right\}^{**} (\text{NLEVELS}-3) \}$$

(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)) =  
(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:

$$100 * (\text{NUMRIDS\_DELETED} / (\text{NUMRIDS\_DELETED} + \text{INDCARD})) < 20$$

The number of pseudo-deleted RIDs on non-pseudo-empty pages should be less than 20 percent.

- Formula F8:

$$100 * (\text{NUM\_EMPTY\_LEAFS}/\text{LEAF}) < 20$$

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.

### Usage notes for index compression

Formula F5 determines the ratio between the amount of space needed by the keys and the amount of space allocated. Formula F6 determines if recreating the index would result in a tree having fewer levels. The following 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. This formula relies on the amount of space needed for all index entries. Both formulae use the amount of space needed for all index entries.

The amount of space needed for all index entries in an uncompressed index is as follows:

$$\text{KEYS} * (\text{LEAF\_RECSIZE} + \text{LEAF\_RECSIZE\_OVERHEAD}) + (\text{INDCARD} - \text{KEYS}) * \text{DUPKEYSIZE}$$

where LEAF\_RECSIZE is the average size of index key and DUPKEYSIZE is the size of a RID.

In a compressed index, LEAF\_RECSIZE is affected by prefix compression. DUPKEYSIZE is not a reliable way to measure the size of duplicate keys on indexes. The amount of space needed in a compressed index is the amount of space needed for all uncompressed index entries multiplied by the index compression ratio.

$$(\text{KEYS} * (\text{LEAF\_RECSIZE} + \text{LEAF\_RECSIZE\_OVERHEAD}) + (\text{INDCARD} - \text{KEYS}) * \text{DUPKEYSIZE}) * \text{COMPRESSION\_RATIO}$$

where COMPRESSION\_RATIO is the estimated index compression ratio in the index. The COMPRESSION\_RATIO is calculated as:

$$(100 - \text{PCT\_PAGES\_SAVED}) / 100$$

where PCT\_PAGES\_SAVED is the estimated percentage of leaf pages saved from index compression. This value is taken from the catalogs. If statistics are not collected, PCT\_PAGES\_SAVED is -1 in the catalogs, and COMPRESSION\_RATIO is 1.

Both the **REORGCHK** command and the REORGCHK\_IX\_STATS procedure show the PCT\_PAGES\_SAVED value.

### Usage notes for partitioned tables

For a data partitioned table, **REORGCHK** returns statistics and reorganization recommendations for the table and the data partitions of the table.

For table statistics and reorganization recommendations, **REORGCHK** lists the table information that contains the SCHEMA.NAME for the table, the table level statistics, and reorg recommendation. After the table information, the information for each data partition information is listed. For each partition, the information includes the SCHEMA.NAME for the table, the partition name, the table statistics for the partition, and reorganization recommendation for the partition.

For index statistics and reorganization recommendations, **REORGCHK** returns the SCHEMA.NAME for each table followed by the fully qualified index name and index statistics and index reorganization recommendation for each nonpartitioned index on the table. If the table has partitioned indexes, **REORGCHK** returns the information for partitioned indexes after the nonpartitioned indexes. **REORGCHK** returns the following information for each data partition of the table, the fully

qualified index name, the partition name, index statistics for the partition, and index reorganization recommendations for the partition.

To provide better data availability to a data partitioned table, a reorganization of a specific data partition can be performed if recommended. **REORG TABLE** with the **ON DATA PARTITION** clause supports reorganizing a partition of a table.

For partitioned indexes, index reorganization of all indexes for a specific data partition can be performed using **REORG INDEXES ALL** with the **ON DATA PARTITION** clause if recommended.

The **REORG INDEX** command can be used to reorganize a nonpartitioned index on a data partitioned table.

See the REORG column for information about reorganization recommendations for data partitioned tables.

The following sample table statistics output is for the partitioned table MYDPARTT1. The table has three data partitions with default partition names PART0, PART1, and PART2.

Table statistics:

SCHEMA.NAME	CARD	OV	NP	FP	ACTBLK	TSIZE	F1	F2	F3	REORG
-----										
Table: USER1.MYDPARTT1	-	-	-	-	-	-	-	-	-	----
Table: USER1.MYDPARTT1 Data Partition: PART0	-	-	-	-	-	-	-	-	-	----
Table: USER1.MYDPARTT1 Data Partition: PART1	-	-	-	-	-	-	-	-	-	----
Table: USER1.MYDPARTT1 Data Partition: PART2	-	-	-	-	-	-	-	-	-	----
-----										

The following sample index statistics output is a partitioned table MYDPARTT1. The table has three data partitions, one nonpartitioned index MYNONPARTIDX1, and one partitioned index MYDPARTIDX1.

Index statistics:

SCHEMA.NAME	INDCARD	LEAF	ELEAF	LVLS	NDEL	KEYS ...	F4	F5	F6	F7	F8	REORG
-----												
Table: USER1.MYDPARTT1 Index: USER1.MYNONPARTIDX1	-	-	-	-	-	...	-	-	-	-	-	----
Index: USER1.MYDPARTIDX1 Data Partition: PART0	-	-	-	-	-	...	-	-	-	-	-	----
Index: USER1.MYDPARTIDX1 Data Partition: PART1	-	-	-	-	-	...	-	-	-	-	-	----
Index: USER1.MYDPARTIDX1 Data Partition: PART2	-	-	-	-	-	...	-	-	-	-	-	----
-----												

---

# 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 ADMIN CONFIGURATION** command.

**Important:** The DB2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in DB2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see “DB2 administration server (DAS) has been deprecated” at .

### Scope

This command resets the DAS configuration file on the administration node of the system to which you are connected.

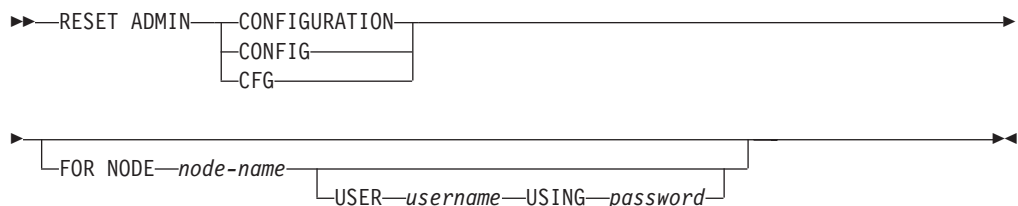
### Authorization

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** *node-name*

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.

## RESET ADMIN CONFIGURATION

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 online take place immediately. Other changes become effective only after they are loaded into memory when you restart the DAS with the **db2admin** command.

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 re-create the DAS to reset the its configuration file.

### 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.

**Important:** This command or API has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in DB2 pureScale environments. For more information, see “Health monitor has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

#### Authorization

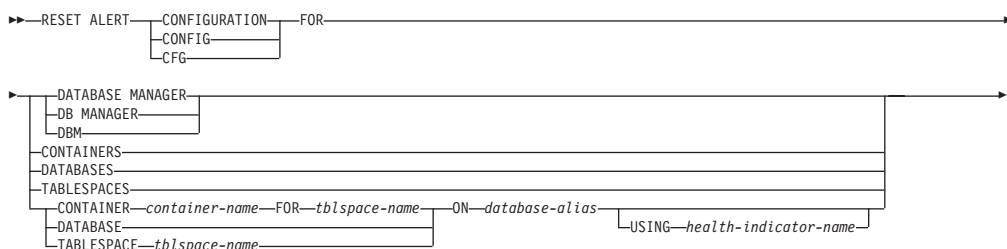
One of the following authorities:

- SYSADM
- SYSMANT
- SYSCTRL

#### Required connection

Instance. An explicit attachment is not required.

#### Command syntax



#### Command parameters

##### DATABASE MANAGER | DB MANAGER | DBM

Resets alert settings for the database manager.

##### 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** *container-name* **FOR** *tblspace-name* **ON** *database-alias* clause.

##### 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.

##### 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

## RESET ALERT CONFIGURATION

spaces that do not have custom settings. Custom settings are defined using the **TABLESPACE** *tblspace-name* **ON** *database-alias* clause.

**CONTAINER** *container-name* **FOR** *tblspace-name* **ON** *database-alias*

Resets the alert settings for the table space container called *container-name*, for the table space specified using the **FOR** *tblspace-name* clause, on the database 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.

**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** *tblspace-name* **ON** *database-alias*

Resets the alert settings for the table space called *tblspace-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.

## RESET DATABASE CONFIGURATION

Resets the configuration of a specific database to the system defaults.

### Scope

This command updates all members by default, except when **MEMBER** is specified to reset only one member.

### Authorization

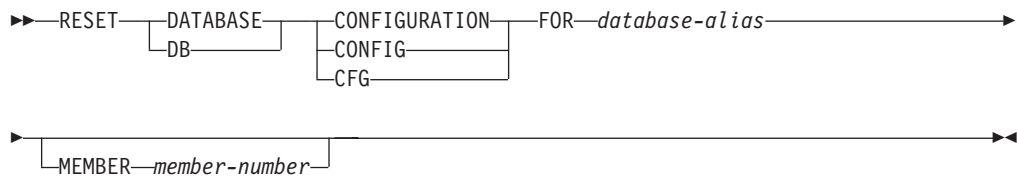
One of the following authorities:

- SYSADM
- SYCTRL
- SYSMANT

### 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.

#### **MEMBER** *member-number*

If a database configuration reset is to be applied to a specific member, this parameter may be used. If this parameter is not provided, the reset will take effect on all members.

### Example

#### Reset the database configuration on a multi-partition instance

A user has a partitioned database environment that has 4 database partitions as defined in the `db2nodes.cfg`:

```

10 gilera 0
20 gilera 1
30 motobi 0
40 motobi 1
  
```

The user has created the `SAMPLE` database on the instance. The catalog partition for `SAMPLE` is on database partition number 10. Let's assume the user is logged on to system `motobi`.



## RESET DATABASE CONFIGURATION

The following command will reset all the database configuration values for database SAMPLE on all database partitions:

```
db2 reset db cfg for sample
```

The following commands will reset all the database configuration values for database SAMPLE, only at database partition number 30:

```
db2 reset db cfg for sample dbpartitionnum 30
```

or

```
export DB2NODE=30
db2 reset db cfg for sample
```

### 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 documented default configuration values, where **auto\_runstats** will be ON. **Self\_tuning\_mem** will be reset to ON on non-partitioned database environments and to OFF on partitioned database environments.

### Compatibilities

For compatibility with previous versions:

- **DBPARTITIONNUM** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.

---

# 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 online, 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 **svccname** 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 **svccname** 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 about which parameters are configurable online 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 DATABASE MANAGER CONFIGURATION

reset the next time you restart the application. If the client is the command line processor, it is necessary to invoke **TERMINATE**.

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 you edit the configuration file manually and do not use the appropriate command. If the checksum is invalid, you must re-create the instance.

## 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.

### Scope

This command is invoked on the currently attached member and only returns information for that member by default. The currently attached member defaults to the host on which the command is run. If there are multiple members per host, the currently attached member is the first member listed in the `db2nodes.cfg` file on that host.

To run the command on a member that is not the current attach member, specify the **AT MEMBER** option to run this command on a specific member, or specify the **GLOBAL** option to run this command on all members to receive an aggregated result.

To change the current attach member from which to run any subsequent commands, use the **ATTACH\_MEMBER** parameter of the **SET CLIENT** command. You must **DETACH** and then again **ATTACH** your application for this client setting to take effect.

### Authorization

One of the following authorities:

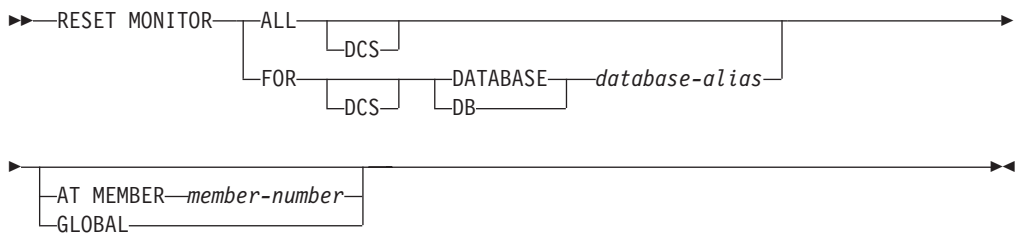
- SYSADM
- SYSCtrl
- SYSMAINT
- SYSMON

### 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 MEMBER** *member-number*

Specifies the member for which the internal counters are to be reset.

**GLOBAL**

Resets the internal counters for all members in a partitioned database environment or in a DB2 pureScale environment.

**Usage notes**

Each process (attachment) has its own private view of the monitor data. If one user resets, or turns off a monitor switch, other users are not affected. Change the setting of the monitor switch configuration parameters to make global changes to the monitor switches.

If **ALL** is specified, some database manager information is also reset to maintain consistency of the returned data, and some member-level counters are reset.

**Compatibilities**

For compatibility with previous versions:

- **DBPARTITIONNUM** or **NODE** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.

## 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 database partition where the command is run. In DB2 pureScale environments, this command resumes I/O write operations for all suspended members.

In DB2 pureScale environments, this command might trigger, when needed, a group crash recovery, which performs the crash recovery for all members of the group, or a member crash recovery.

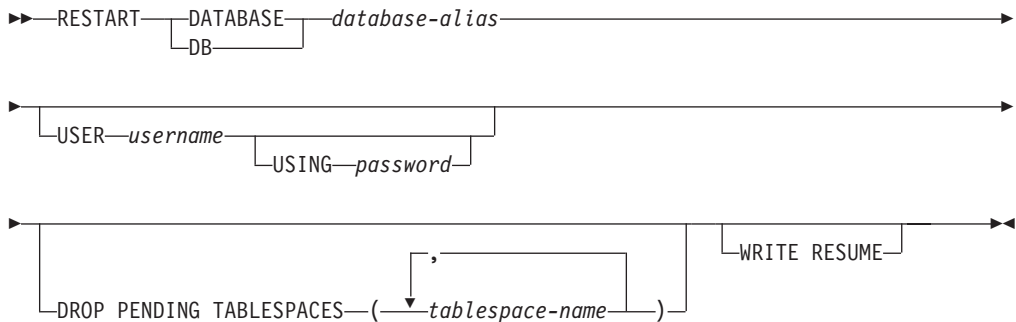
### Authorization

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.

If a problem occurs with a container during a member crash recovery, the member crash recovery operation will fail. To correct this situation, shut down all members using a **db2stop -force** command and manually initiate a group crash recovery by reissuing the **RESTART DATABASE** command. Following the group crash recovery, the table space corresponding to the problematic container will not be available (for more information, see the "Recovering with damaged table spaces" topic).

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 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

Forces a restart for databases that crashed while I/O write operations were suspended. Before performing crash recovery, this parameter will resume I/O write operations. In DB2 pureScale environments, this parameter resumes I/O write operations for all suspended members.

You can also use the **WRITE RESUME** parameter if the connection that you used to suspend I/O write operations is currently hung and all subsequent connection attempts are also hanging. When you use the parameter in this case, the **RESTART DATABASE** command resumes I/O write operations for the database without performing crash recovery. The **RESTART DATABASE** command with the **WRITE RESUME** parameter performs crash recovery only when you use it after a database crash.

### Usage notes

Issue this command if an attempt to connect to a database returns an error message indicating that the database must be restarted. This error message is generated only if the previous session with this database terminated abnormally, for example, because of a power failure.

### Partitioned database environments

On a partitioned database system, to resolve the indoubt transactions, you should issue the **RESTART DATABASE** command on all database partitions, as shown in the following example: `db2_all "db2 restart database database-alias"`

If you restart the database on only a single database partition within a partitioned database environment, a message indicating that you must restart the database might be returned on a subsequent database query. This message is returned because you did not restart the database on a database partition on which the query depends. Restarting the database on all database partitions solves the problem.

### DB2 pureScale environments

In a DB2 pureScale environment, the **RESTART DATABASE** command is the only way to manually invoke crash recovery. The database manager automatically determines whether a group crash recovery or a member crash recovery is required. After group crash recovery completes, if any indoubt transactions exist on any of the members that did not perform group crash recovery, a member crash recovery on those members must be performed to enable resolution of those indoubt transactions.

## RESTART DATABASE

If the write operations for the database were in the process of being suspended at the time of the crash, perform crash recovery by using the **RESTART DATABASE** command. If a **SET WRITE SUSPEND** operation is running on another member, you might have to wait for the **SET WRITE SUSPEND** operation to complete before performing crash recovery. After it is complete, submit the **RESTART DATABASE** command with the **WRITE RESUME** parameter to restart the database and resume write operations for all the suspended members.

If the write operations were successfully suspended on the database at the time of the crash, there are two ways that you can perform crash recovery:

- You can use the **RESTART DATABASE** command with the **WRITE RESUME** parameter.
- You can issue a **SET WRITE** command with the **RESUME** parameter from an active member and then issue the **RESTART DATABASE** command from the crashed member.

### Related information:



---

## RESTORE DATABASE

The **RESTORE DATABASE** command re-creates 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 perform the following services:

- Overwrite a database with a different image or restore the backup copy to a new database.
- The restore utility in DB2 Version 9.8 cannot be used to restore backup images that are backed up on any other DB2 software version.
- Restore backup images in DB2 Version 9.7 that were backed up on DB2 Version 9.5.
  - If a database upgrade 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.
- Restore a table space level backup.
- Transport a set of table spaces, storage groups and SQL schemas from database backup image to a database using the **TRANSPORT** option (in DB2 Version 9.7 Fix Pack 2 and later fix packs). The **TRANSPORT** option is not supported in the DB2 pureScale environment.
- If the database name exists when this command is issued, it will replace and redefine all storage groups as they were at the time the backup image was produced, unless otherwise redirected by the user.

For information about 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 *Data Recovery and High Availability Guide and Reference*.

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.)

A restore operation run over an existing database and existing containers reuses the same containers and table space map.

## RESTORE DATABASE

A restore operation run against a new database reacquires all containers and rebuilds an optimized table space map. A restore operation run over an existing database with one or more missing containers also reacquires all containers and rebuilds an optimized table space map.

### Scope

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

You cannot restore SYSCATSPACE online.

### Authorization

To restore to an existing database, one of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT

To restore to a new database, one of the following authorities:

- SYSADM
- SYSCTRL

### Required connection

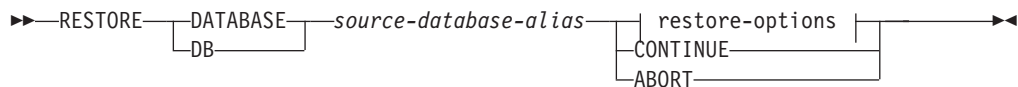
The required connection will vary based on the type of restore action:

- 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.

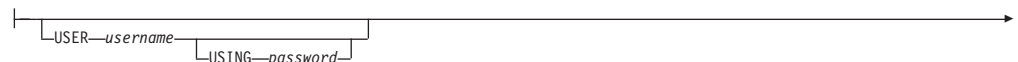
To restore to a new database at an instance different from the current instance, it is necessary to first attach to the instance where the new database will reside. The new instance can be local or remote. The current instance is defined by the value of the **DB2INSTANCE** environment variable.

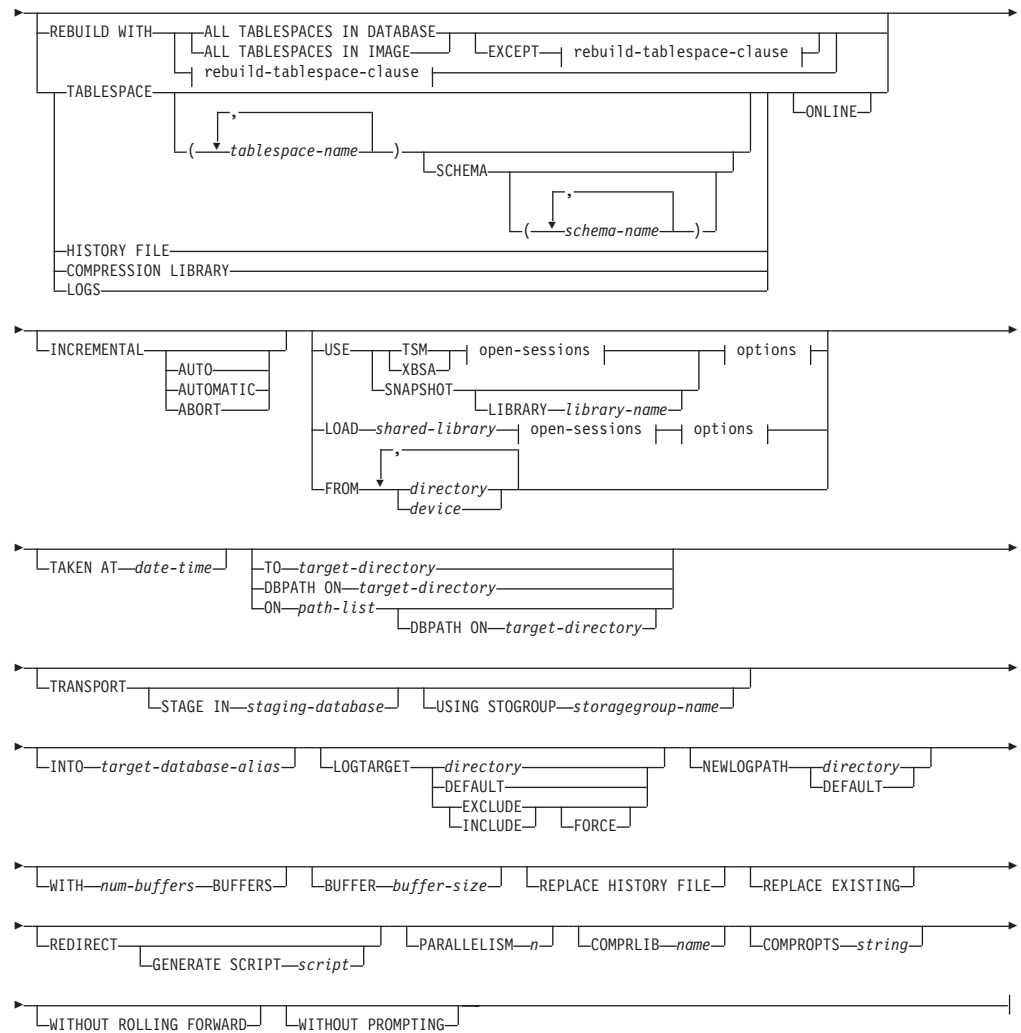
- For snapshot restore, *instance* and *database* connections are required.

### Command syntax

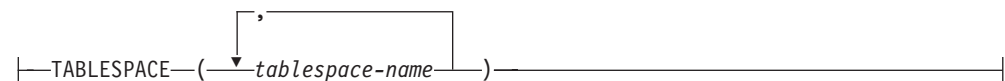


#### restore-options:

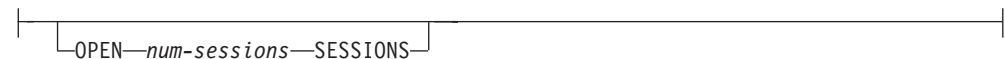




## rebuild-tablespace-clause:



## open-sessions:



## options:



## RESTORE DATABASE

### 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.

#### **TABLESPACE** *tablespace-name*

A list of names used to specify the table spaces that are to be restored.

Table space names are required when the **TRANSPORT** option is specified.

#### **SCHEMA** *schema-name*

A list of names used to specify the schemas that are to be restored.

Schema names are required if the **TRANSPORT** option is specified. The **SCHEMA** option is only valid when the **TRANSPORT** option is specified.

#### **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 by using Tivoli Storage Manager (TSM) as the target device.

**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.

#### SNAPSHOT

Specifies that the data is to be restored from a snapshot backup.

You cannot use the **SNAPSHOT** parameter with any of the following parameters:

- **INCREMENTAL**
- **TO**
- **ON**
- **DBPATH ON**
- **INTO**
- **NEWLOGPATH**
- **WITH *num-buffers* BUFFERS**
- **BUFFER**
- **REDIRECT**
- **REPLACE HISTORY FILE**
- **COMPRESSION LIBRARY**

## RESTORE DATABASE

- **PARALLELISM**
- **COMPRLIB**
- **OPEN** *num-sessions* **SESSIONS**
- **HISTORY FILE**
- **LOGS**

Also, you cannot use the **SNAPSHOT** parameter with any restore operation that involves a table space list, which includes the **REBUILD WITH** option.

The default behavior when restoring data from a snapshot backup image will be a FULL DATABASE OFFLINE restore of all paths that make up the database including all containers, local volume directory and database path (DBPATH). By default, the logs are excluded from snapshot restore (**LOGTARGET EXCLUDE** is the default for all snapshot restores unless **LOGTARGET INCLUDE** is explicitly stated). If a timestamp is provided, then that snapshot backup image will be restored.

### **LIBRARY** *library-name*

Integrated into IBM Data Server is a DB2 ACS API driver for the following storage hardware:

- IBM TotalStorage SAN Volume Controller
- IBM Enterprise Storage Server Model 800
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS

If you have other storage hardware, and a DB2 ACS API driver for that storage hardware, you can use the **LIBRARY** parameter to specify the DB2 ACS API driver.

The value of the **LIBRARY** parameter is a fully-qualified library file name.

## OPTIONS

### *"options-string"*

Specifies options to be used for the restore operation. The string will be passed exactly as it was entered, without the double quotation marks.

### *@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. The file must be a fully qualified file name.

You cannot use the **VENDOROPT** database configuration parameter to specify vendor-specific options for snapshot restore operations. You must use the **OPTIONS** parameter of the restore utilities instead.

### **OPEN** *num-sessions* **SESSIONS**

Specifies the number of I/O sessions that are to be used with TSM or the vendor product.

**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.

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.

**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 a database. If the database contains multiple storage groups this option will redirect all storage groups to the specified paths, such that every defined storage group will use *path-list* as its new storage group paths. Using this parameter with a database that has no storage groups defined or is not enabled for automatic storage results in an error (SQL20321N). The existing

## RESTORE DATABASE

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 this option is specified with the **REDIRECT** option, then this option takes effect before the initial **RESTORE ... REDIRECT** command returns to the caller, and before any **SET STOGROUP PATHS** or **SET TABLESPACE CONTAINERS** statements are issued. If any storage group paths are subsequently redirected, those modifications override any paths specified in the initial **RESTORE ... ON path-list** command.

Any storage groups that have their paths redefined during a restore operation do not have any storage path related operations replayed during a subsequent roll forward operation.

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 database partitions in a **RESTORE\_PENDING** state. The non-catalog database partitions 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 before 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.

Using the **RESTORE** command with the **ON** clause has the same implications as a redirected restore operation.

You cannot use the **ON** parameter to re-define storage paths for schema transport. Schema transport will use existing storage paths on the target database.

### **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.

### **TRANSPORT INTO** *target-database-alias*

Specifies the existing target database alias for a transport operation. The table spaces and schemas being transported are added to the database.

The **TABLESPACE** and **SCHEMA** options must specify table space names and schema names that define a valid transportable set or the transport operation fails. **SQLCODE=SQL2590N rc=1**

The system catalogs cannot be transported. **SQLCODE=SQL2590N rc=4.**



After the schemas have been validated by the **RESTORE** command, the system catalog entries describing the objects in the table spaces being transported are created in the target database. After completion of the schema recreation, the target database takes ownership of the physical table space containers.

The physical and logical objects contained in the table spaces being restored are re-created in the target database and the table space definitions and containers are added to the target database. Failure during the creation of an object, or the replay of the DDL returns an error.

#### **STAGE IN** *staging-database*

Specifies the name of a temporary staging database for the backup image that is the source for the transport operation. If the **STAGE IN** option is specified, the temporary database is not dropped after the transport operation completes. The database is no longer required after the transport has completed and can be dropped by the DBA.

The following is true if the **STAGE IN** option is not specified:

- The database name is of the form SYSTGxxx where xxx is an integer value.
- The temporary staging database is dropped after the transport operation completes.

#### **USING STOGROUP** *storagegroup-name*

For automatic storage table spaces, this specifies the target storage group that will be associated with all table spaces being transported. If the storage group is not specified, then the currently designated default storage group of the target database is used. This clause only applies to automatic storage table spaces and is only valid during a schema transport operation.

Identifies the storage group in which table space data will be stored. *storagegroup-name* must identify a storage group that exists at the *target-database-alias* of the **TRANSPORT** operation. (SQLSTATE 42704). This is a one-part name.

#### **LOGTARGET** *directory*

Non-snapshot restores:

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. This option automatically appends the database partition number and a log stream ID to the path.

#### **DEFAULT**

Restore log files from the backup image into the database's default log directory, for example /home/db2user/db2inst/NODE0000/SQL00001/LOGSTREAM0000.

Snapshot restores:

#### **INCLUDE**

Restore log directory volumes from the snapshot image. If this option is specified and the backup image contains log directories, then they will be restored. Existing log directories and log files on

## RESTORE DATABASE

disk will be left intact if they do not conflict with the log directories in the backup image. If existing log directories on disk conflict with the log directories in the backup image, then an error will be returned.

### EXCLUDE

Do not restore log directory volumes. If this option is specified, then no log directories will be restored from the backup image. Existing log directories and log files on disk will be left intact if they do not conflict with the log directories in the backup image. If a path belonging to the database is restored and a log directory will implicitly be restored because of this, thus causing a log directory to be overwritten, an error will be returned.

### FORCE

Allow existing log directories in the current database to be overwritten and replaced when restoring the snapshot image. Without this option, existing log directories and log files on disk which conflict with log directories in the snapshot image will cause the restore to fail. Use this option to indicate that the restore can overwrite and replace those existing log directories.

**Note:** Use this option with caution, and always ensure that you have backed up and archived all logs that might be required for recovery.

**Note:** If **LOGTARGET** is not specified for snapshot restores, then the default **LOGTARGET** directory is **LOGTARGET EXCLUDE**.

### 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.

### DEFAULT

After the restore completes, the database should use the default log directory: `/home/db2user/db2inst/NODE0000/SQL00001/LOGSTREAM0000` for logging.

### 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.

**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, 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 or **SET STOGROUP PATHS** commands, and then by a **RESTORE DATABASE** command with the **CONTINUE** option. For example:

```
RESTORE DB SAMPLE REDIRECT

SET STOGROUP PATHS FOR sg_hot ON '/ssd/fs1', '/ssd/fs2'
SET STOGROUP PATHS FOR sg_cold ON '/hdd/path1', '/hdd/path2'

RESTORE DB SAMPLE CONTINUE
```

If a storage group has been renamed since the backup image was produced, the storage group name specified on the **SET STOGROUP PATHS** command refers to the storage group name from the backup image, not the most recent name.

All commands associated with a single redirected restore operation must be invoked from the same window or CLP session.

**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 following examples 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.

**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.

## RESTORE DATABASE

### COMPRLIB *name*

Indicates the name of the library to be used to perform the decompression (for example, db2compr.dll for Windows; libdb2compr.so for Linux or UNIX systems). The name must be a fully qualified path referring to a file on the server. If this parameter is not specified, the DB2 database system 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:

```
wbdb.0.db2inst1.DBPART000.200802241234.001
wbdb.0.db2inst1.DBPART001.200802241234.001
wbdb.0.db2inst1.DBPART002.200802241234.001
wbdb.0.db2inst1.DBPART003.200802241234.001
```

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_a11 '<<+0< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234149
INTO wsdb REPLACE EXISTING'
db2_a11 '<<+1< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234427
INTO wsdb REPLACE EXISTING'
db2_a11 '<<+2< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234828
INTO wsdb REPLACE EXISTING'
db2_a11 '<<+3< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331235235
INTO wsdb REPLACE EXISTING'
```

The **db2\_a11** utility issues the restore command to each specified database partition. When performing a restore using **db2\_a11**, 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\_a11** 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)
```

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 example 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
```

## RESTORE DATABASE

- 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`

- 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`

- 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/SAMPLE/NODE0000/LOGSTREAM0000/'  
-- 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'
);
-- *****
-- ** 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

```

## RESTORE DATABASE

```
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_MEMBER 0;
SET CLIENT CONNECT_MEMBER 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/TEST/NODE0000/LOGSTREAM0000/'
-- WITH <num-buff> BUFFERS
-- BUFFER <buffer-size>
-- REPLACE HISTORY FILE
-- REPLACE EXISTING
REDIRECT
-- PARALLELISM <n>
-- WITHOUT ROLLING FORWARD
-- WITHOUT PROMPTING
;
-- *****
-- ** storage group definition
-- ** Default storage group ID          = 0
-- ** Number of storage groups          = 3
-- *****
-- ** Storage group name                 = SG_DEFAULT
-- ** Storage group ID                   = 0
-- ** Data tag                            = None
-- *****
-- SET STOGROUP PATHS FOR SG_DEFAULT
-- ON '/hdd/path1'
-- , '/hdd/path2'
-- ;
-- *****
-- ** Storage group name                 = SG_HOT
-- ** Storage group ID                   = 1
-- ** Data tag                            = 1
-- *****
-- SET STOGROUP PATHS FOR SG_HOT
-- ON '/ssd/fs1'
-- , '/ssd/fs2'
-- ;
-- *****
-- ** Storage group name                 = SG_COLD
-- ** Storage group ID                   = 2
-- ** Data tag                            = 9
-- *****
-- SET STOGROUP PATHS FOR SG_COLD
```



```

-- ON '/hdd/slowpath1'
-- ;
-- *****
-- ** 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
-- ** Storage group ID               = 0
-- ** Source storage group ID        = -1
-- ** Data tag                       = None
-- ** 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
-- ** Storage group ID               = 1
-- ** Source storage group ID        = -1
-- ** Data tag                       = 1
-- ** Auto-resize enabled            = Yes
-- ** Total number of pages          = 256
-- ** Number of usable pages         = 224
-- ** High water mark (pages)       = 96
-- *****
-- ** 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
-- ** Storage group ID               = 2
-- ** Source storage group ID        = -1
-- ** Data tag                       = 9
-- ** 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

```

## RESTORE DATABASE

```
-- ** 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
-- *****
```

10. The following are examples of the **RESTORE DB** command using the **SNAPSHOT** option:

Restore log directory volumes from the snapshot image and do not prompt.

```
db2 restore db sample use snapshot LOGTARGET INCLUDE without prompting
```

Do not restore log directory volumes and do not prompt.

```
db2 restore db sample use snapshot LOGTARGET EXCLUDE without prompting
```

Do not restore log directory volumes and do not prompt. When **LOGTARGET** is not specified, then the default is **LOGTARGET EXCLUDE**.

```
db2 restore db sample use snapshot without prompting
```

Allow existing log directories in the current database to be overwritten and replaced when restoring the snapshot image containing conflicting log directories, without prompting.

```
db2 restore db sample use snapshot LOGTARGET EXCLUDE FORCE without prompting
```

Allow existing log directories in the current database to be overwritten and replaced when restoring the snapshot image containing conflicting log directories, without prompting.

```
db2 restore db sample use snapshot LOGTARGET INCLUDE FORCE without prompting
```

11. The following are examples of a transport operation using the **RESTORE** command with the **TRANSPORT REDIRECT** option:

Given a source database (TT\_SRC) backup image, with storage paths on /src , and a target database (TT\_TGT) with storage paths on /tgt :

```
> RESTORE DB TT_SRC TABLESPACE (AS1) SCHEMA (KRODGER) TRANSPORT INTO TT_TGT REDIRECT
```

```
SQL1277W A redirected restore operation is being performed. Table space configuration can now be viewed and table spaces that do not use automatic storage can have their containers reconfigured.
DB20000I The RESTORE DATABASE command completed successfully.
```

Table space 'AS1' is transported into a container path, similar to:

```
/tgt/krodger/NODE0000/TT_TGT/T0000003/C0000000.LRG
```

To specify a target storage group for the transported table spaces, the **USING STOGROUP** option of the **RESTORE** command can be used. In the following example both table spaces TS1 and TS2 will be restored into the SG\_COLD storage group:

```
RESTORE DB TT_SRC TABLESPACE (TS1, TS2) SCHEMA (KRODGER) TRANSPORT INTO TT_TGT USING STOGROUP SG_COLD
```

**Note:** The **USING STOGROUP** option of the **RESTORE** command is only valid during a transport operation, and can not be used to specify a target storage group during any other restore operation.

To perform a transport into the default storage group of the target database, the **USING STOGROUP** option does not need to be specified:

```
RESTORE DB TT_SRC TABLESPACE (TS3) SCHEMA (KRODGER) TRANSPORT INTO TT_TGT
```

The storage group name specified on the **RESTORE** command during the **TRANSPORT** operation must currently be defined in the target database. It does not need to be defined within the backup image or source database.

If the parameter **AT DBPARTITIONNUM** is used to re-create 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.

### 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.
- You can use the **OPTIONS** parameter to enable restore operations in TSM environments supporting proxy nodes. For more information, see the “Configuring a Tivoli Storage Manager client” topic.
- 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 the library 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 the DB2 database system normally supports cross-platform restores involving the two systems.
- A backed up SMS table space can only be restored into a SMS table space. You cannot restore it into a DMS table space, or vice versa.
- 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.

## RESTORE DATABASE

- 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.
- Offline full database backups as well as offline incremental database backups can be restored to a later database version, whereas online backups cannot. For multi-partition databases, the catalog partition must first be restored individually, followed by the remaining database partitions (in parallel or serial). However, the implicit database upgrade done by the restore operation can fail. In a multi-partition database it can fail on one or more database partitions. In this case, you can follow the **RESTORE DATABASE** command with a single **UPGRADE DATABASE** command issued from the catalog partition to upgrade the database successfully.
- In a partitioned database environment, a table space can have a different storage group association on different database partitions. When a redirected restore modifies table space containers from DMS to automatic storage, the table space is associated with the default storage group. If a new default storage group is selected in between redirected restores of different database partitions, then the table space will have an inconsistent storage group association across the partitioned database environment. If this occurs then use the **ALTER TABLESPACE** statement to alter the table space to use automatic storage on all database partitions, and rebalance if necessary.
- The **TRANSPORT** option is supported only when the client and database code page are equal.

### Snapshot restore

Like a traditional (non-snapshot) restore, the default behavior when restoring a snapshot backup image will be to NOT restore the log directories —**LOGTARGET EXCLUDE**.

If the DB2 database manager detects that any log directory's group ID is shared among any of the other paths to be restored, then an error is returned. In this case, **LOGTARGET INCLUDE** or **LOGTARGET INCLUDE FORCE** must be specified, as the log directories must be part of the restore.

The DB2 database manager will make all efforts to save existing log directories (primary, mirror and overflow) before the restore of the paths from the backup image takes place.

If you want the log directories to be restored and the DB2 database manager detects that the pre-existing log directories on disk conflict with the log directories in the backup image, then the DB2 database manager

will report an error. In such a case, if you have specified **LOGTARGET INCLUDE FORCE**, then this error will be suppressed and the log directories from the image will be restored, deleting whatever existed beforehand.

There is a special case in which the **LOGTARGET EXCLUDE** option is specified and a log directory path resides under the database directory (for example, `/NODExxxx/SQLxxxxx/LOGSTREAMxxxxx/`). In this case, a restore would still overwrite the log directory as the database path, and all of the contents beneath it, would be restored. If the DB2 database manager detects this scenario and log files exist in this log directory, then an error will be reported. If you specify **LOGTARGET EXCLUDE FORCE**, then this error will be suppressed and those log directories from the backup image will overwrite the conflicting log directories on disk.

### Transporting table spaces and schemas

The complete list of table spaces and schemas must be specified.

The target database must be active at the time of transport.

If an online backup image is used, then the staging database is rolled forward to the end of the backup. If an offline backup image is used, then no rollforward processing is performed.

A staging database consisting of the system catalog table space from the backup image is created under the path specified by the **dfbdbpath** database parameter. This database is dropped when the **RESTORE DATABASE** command completes. The staging database is required to extract the DDL used to regenerate the objects in the table spaces being transported.

When transporting table spaces, the DB2 database manager attempts to assign the first available buffer pool of matching page size to the table space that is transported. If the target database does not have buffer pools that are of matching page size of table spaces transported, then a hidden buffer pool might be assigned. Hidden buffer pools are temporary place holders for transported table spaces. You can check buffer pools assigned to transported table spaces after transport completes. You can issue the **ALTER TABLESPACE** command to update buffer pools.

If database rollforward detects a table space schema transport log record, the corresponding transported table space will be taken offline and moved into drop pending state. This is because database does not have complete logs of transported table spaces to rebuild transported table spaces and their contents. You can take a full backup of the target database after transport completes, so subsequent rollforward does not pass the point of schema transport in the log stream.

### Transporting storage groups

A transport operation can not modify the currently defined storage groups of the target database, and new storage groups can not be explicitly created during a transport.

The default target storage group of the transport is the default storage group of the target database of the operation. It is also possible to explicitly redirect all table spaces being restored during a transport operation into a specific storage group on the target database.

During a transport operation, when a **RESTORE** command using the **TRANSPORT REDIRECT** option is issued, the default storage group configuration for automatic storage table spaces is not the configuration contained in the backup image, but instead the storage groups and storage

## RESTORE DATABASE

group paths of the target database. This is because automatic storage table spaces must be restored and redirected directly into existing storage group paths, as defined on the target database.

**Related information:**

---

## 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 authorities:

- SYSADM
- SYSCTRL
- SYSMANT

### Required connection

None

### Command syntax

►► REWIND TAPE [ON *device*]

### Command parameters

**ON** *device*

Specifies a valid tape device name. The default value is \\.\TAPE0.

---

### ROLLFORWARD DATABASE

The ROLLFORWARD DATABASE command recovers a database by applying transactions recorded in the database log files. This command can be run after a database or a table space backup image was restored, or if any table spaces were 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 be invoked only from the catalog partition but each partition participates in the rollforward operation. 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.

In a DB2 pureScale environment, this command can be issued from any member, and online table space-level rollforward operation can be performed while other members are online. Unlike in partitioned database environments in which users can choose to rollforward through a subset of the database partitions, in a DB2 pureScale environment the logs from all members are automatically applied. If a rollforward operation is in progress on a member when it fails, the **ROLLFORWARD** command can be reissued from any member. The rollforward resumes from where it was left off when the original member failed.

For partitioned tables, you are also required to roll forward related table spaces to the same point in time. This requirement applies to table spaces that contain 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.

It is not possible to roll forward through log files created on a previous DB2 release version. This restriction is an important consideration when upgrading to a new DB2 database release version.

#### Authorization

One of the following authorities:

- SYSADM
- SYSCtrl
- SYSMaint

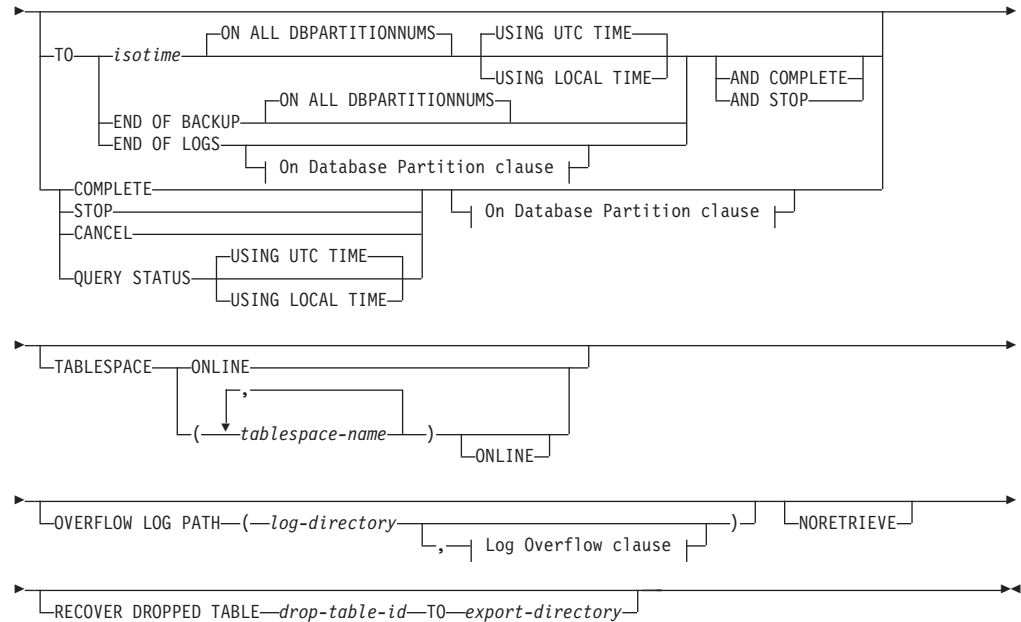
#### Required connection

None. This command establishes a database connection.

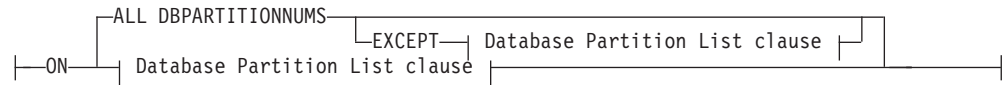
#### Command syntax

```
►► ROLLFORWARD DATABASE database-alias [DB] USER username [USING password]
```





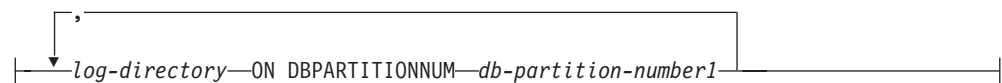
**On Database Partition clause:**



**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.

## ROLLFORWARD DATABASE

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). A rollforward operation to a point in time returns a success message only if there is a transaction with a larger timestamp value in the log files. Even if there is no transaction with a larger timestamp, you can still issue a rollforward operation with the COMPLETE option.

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 saving 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 UTC TIME

Allows you to rollforward to a point in time that is specified as UTC time. This is the default option.

### USING LOCAL TIME

Allows you to rollforward to a point in time that is the server's local time rather than UTC time.

#### Note:

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.
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.
5. It is important to choose the **USING LOCAL TIME** or the **USING UTC TIME** (formerly known as GMT time) correctly. If not specified, the default is **USING UTC TIME**. Any mistake in the selection may cause rollforward to reach a different point in time than expected and truncate the logs after that point in time. Mistaking a local timestamp as a UTC timestamp may cause

the required logs to be truncated undesirably and prevent further rollforwards to a point later than the mistaken time.

6. Specify a valid timestamp when recovering a database. A valid timestamp would be the time that the last backup in the partitioned database system was completed.
7. When issuing multiple **ROLLFORWARD DATABASE** commands, the timestamp you specify for each subsequent command must be greater than the timestamp you specified in the previous command.

### 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.

### END OF BACKUP

Specifies that all partitions in the partitioned database, or all members in the DB2 pureScale environment, should be rolled forward to the *minimum recovery time*. See here for an example.

### ALL DBPARTITIONNUMS | 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.

### 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.

## ROLLFORWARD DATABASE

- 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 processed, 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 and a DB2 pureScale environment, this status information is returned for each database partition or member. The information returned contains the following fields:

#### Member number

The member number or database partition number. In a single-partition environment, this is always 0.

#### 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. In a DB2 pureScale environment, this string lists all log

files that have been processed by the **ROLLFORWARD** command on all DB2 members. 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. This field is not updated in case of a table space rollforward recovery operation.

#### 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. The table spaces containing the index partitions are included in the list of pieces of a partitioned table. 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 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 DATABASE

rollforward recovery is in progress. To support concurrency with other applications in a DB2 pureScale environment, the table space-level rollforward operation might need to test for certain internal locks. A rollforward operation in this environment might fail with lock timeout even if **locktimeout** is set to -1.

### **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 by disabling the retrieval of archived logs.

### **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, in the Backup ID column of the output listing. 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 rollforward 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 (that is, 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. 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 offline 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 DATABASE

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.

- *Example 10 (partitioned database environment or DB2 pureScale environment):* You can use the **TO END OF BACKUP** clause with the **ROLLFORWARD** command to roll forward all database partitions or members to the minimum recovery time. The minimum recovery time is the earliest point in time during a rollforward when a database is consistent (when the objects listed in the database catalogs match the objects that physically exist on disk). Manually determining the correct point in time to which to roll forward a database is difficult, particularly for a partitioned database or in a DB2 pureScale instance. The **END OF BACKUP** option makes it easy.

```
db2 rollforward db sample to end of backup and complete
```

- *Example 11 (partitioned database environment or DB2 pureScale environment):* The user restores an online database backup that includes logs which are restored to /backup\_logs:

```
db2 restore database dwtest ... logtarget /backup_logs ...
```

Then the user does a rollforward to end of backup using the logs from the backup through an overflow log path:

```
db2 rollforward database dwtest to end of backup and stop overflow log path (/backup_logs)
```

- *Example 12 (DB2 pureScale environment):* The user wants to provide an alternate set of log files for the rollforward using the overflow log path. Assume that there are three log streams: 0, 1, and 2. To do this the user must create the necessary set of directories:



```
mkdir -p /overflow/NODE0000/LOGSTREAM0000
mkdir -p /overflow/NODE0000/LOGSTREAM0001
mkdir -p /overflow/NODE0000/LOGSTREAM0002
```

Then the user must populate these directories with the log files that will be used.

**Note:** If any of the log files in the overflow log path is also present in the active log path, then the log file from the active log path will be used instead of the one in the overflow log path.

After the directories are populated, the user can issue a rollforward command such as:

```
rollforward database dwtest to end of logs overflow log path (/overflow)
```

After verifying that the rollforward got far enough, the user can complete the rollforward by issuing the command:

```
rollforward database dwtest to end of logs and stop overflow log path (/overflow)
```

## 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 are 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. In a DB2 pureScale environment, the **LIST TABLESPACES** command is deprecated; use the following monitoring UDF: `SELECT * FROM TABLE(SYSPROC.MON_GET_TABLESPACE('TBSpace_1,0'))`

In a DB2 pureScale environment, ensure that there is adequate free disk space in the retrieval path before starting a rollforward operation. This allows the operation to retrieve the larger number of files from the archive, as required in a DB2 pureScale environment, without affecting performance. Use the following formula to calculate how much space you need to retrieve the active log space for all members: **(logprimary + logsecond) \* number of members**.

If a table space that contains a system-period temporal table or bitemporal table is rolled forward to a point in time, the **ROLLFORWARD** command must include the table space that contains the associated history table. You can roll forward the table space for the temporal table or the table space for the history table individually, but only to the end of logs.

Rolling databases forward might require a load recovery by using tape devices. If prompted for another tape, you can respond with one of the following inputs:

- c** Continue. Continue to use the device that generated the warning message (for example, when a new tape was mounted)
- d** Device terminate. Stop using the device that generated the warning message (for example, when there are no more tapes)

## ROLLFORWARD DATABASE

- t** Terminate. Take all affected table spaces 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.

If database rollforward detects a table space schema transport log record, the corresponding transported table space is taken offline and moved into drop pending state. This is because database does not have complete logs of transported table spaces to rebuild transported table spaces and their contents. You can take a full backup of the target database after transport completes, so subsequent rollforward does not pass the point of schema transport in the log stream.

**Note:** Rolling forward through a redistribute operation cannot restore the database content since log records are not recorded for data redistribution. See the “**REDISTRIBUTE DATABASE PARTITION GROUP** command”.

**Related information:**

## 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.

---

# 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, call the RUNSTATS command when the table has had many updates, or after reorganizing the table. For a statistical view, call the RUNSTATS command 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 by using the ALTER VIEW statement.

### Scope

The RUNSTATS 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 authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- DBADM
- SQLADM
- CONTROL privilege on the table
- LOAD authority

You do not need any explicit privilege to use this command on any declared temporary table that exists within its connection.

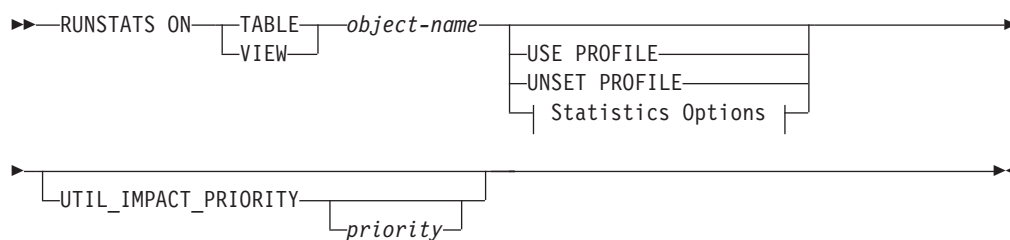
For statistical views, one of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- DBADM
- SQLADM
- CONTROL privilege on the statistical view

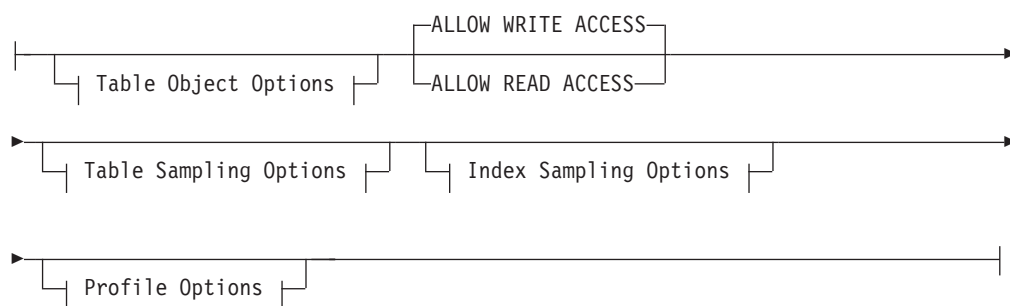
### Required connection

Database

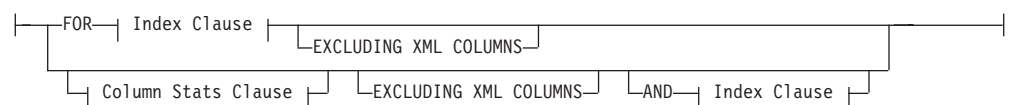
### Command syntax



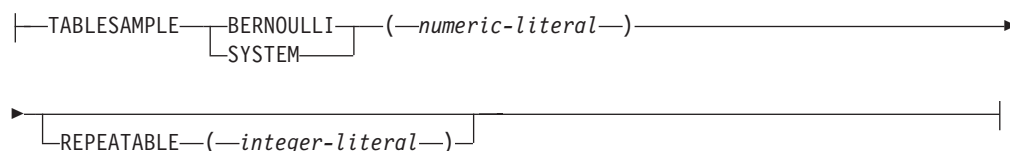
### Statistics Options:



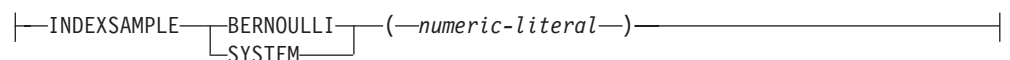
### Table Object Options:



### Table Sampling Options:



### Index Sampling Options:

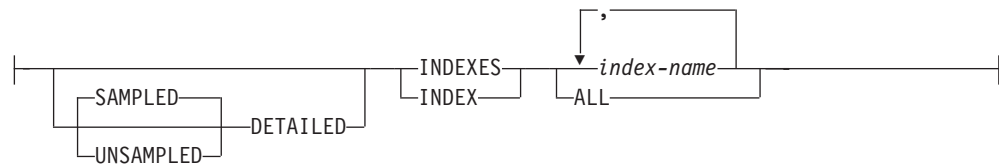


### Profile Options:

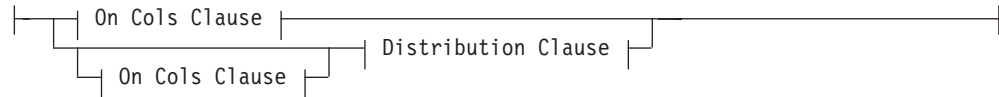


# RUNSTATS

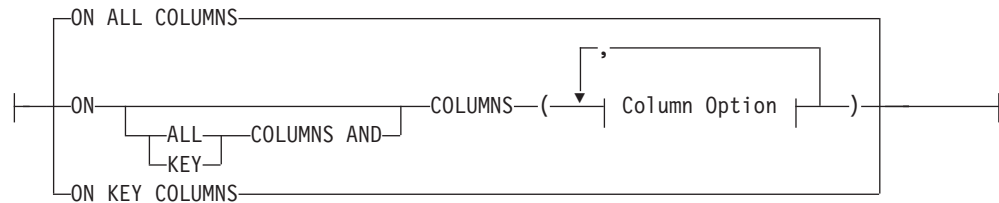
## Index Clause:



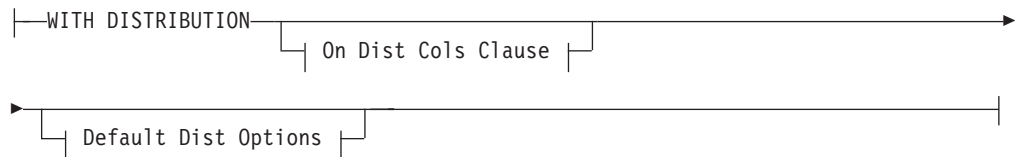
## Column Stats Clause:



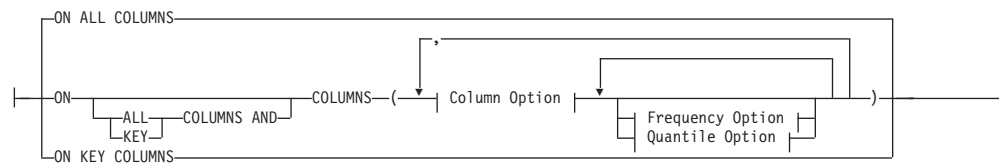
## On Cols Clause:



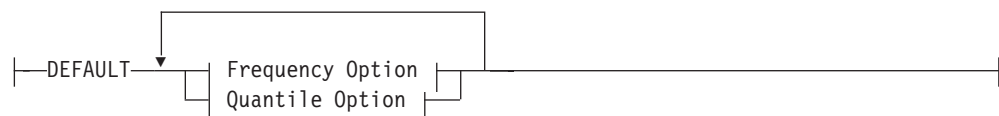
## Distribution Clause:



## On Dist Cols Clause:



## Default Dist Option:



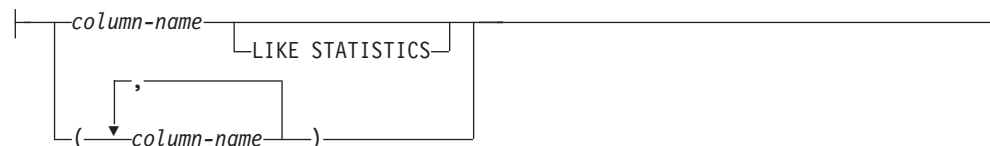
## Frequency Option:



**Quantile Option:**



**Column Option:**



**Command parameters**

*object-name*

Identifies the table or statistical view on which statistics are to be collected. This parameter must not be a hierarchy table. For typed tables, *object-name* must be the name of the root table of the table hierarchy. The fully qualified name or alias in the form: *schema.object-name* must be used. The schema is the user name under which the table was created.

*index-name*

Identifies an existing index defined on the table. If you do not specify the fully qualified name in the form: *schema.index-name*, the default schema is assumed. This option cannot be used for views.

**USE PROFILE**

This option allows **RUNSTATS** to employ a previously stored statistics profile to gather statistics for a table or statistical view. The statistics profile is created using the **SET PROFILE** options and is updated using the **UPDATE PROFILE** options.

**UNSET PROFILE**

Specify this option to remove an existing statistics profile. For example,  
 RUNSTATS ON tablemyschema.mytable UNSET PROFILE

**FOR INDEXES**

Collects and updates statistics for the indexes only. If no table statistics had been previously collected on the table, basic table statistics are also collected. These basic statistics do not include any distribution statistics. This option cannot be used for views.

**AND INDEXES**

Collects and updates statistics for both the table and the indexes. This option cannot be used for views.

**DETAILED**

Calculates extended index statistics. The extended index statistics are the CLUSTERFACTOR and PAGE\_FETCH\_PAIRS statistics that are gathered for relatively large indexes. Not all index entries are examined, a CPU sampling technique is employed instead to improve performance. This option cannot be used for views.

**SAMPLED**

Used together only with the **DETAILED** parameter. Specifying this option does not change the default functionality from **DETAILED**. This option is left in for compatibility with previous versions of DB2. This option cannot be used for views.

## UNSAMPLED

This option, when used with the **DETAILED** option, forces **RUNSTATS** to examine every entry in the index to compute the extended index statistics. This option cannot be used for views and it cannot be used together with scan index sampling (**INDEXSAMPLE** keyword). This option significantly increases **RUNSTATS** resource consumption, while rarely providing significant improvement over the **DETAILED** or **SAMPLED DETAILED** options, which are equivalent.

## ON ALL COLUMNS

To collect statistics on all eligible columns, use the **ON ALL COLUMNS** clause. Columns can be specified either for basic statistics collection (**On Co1s** clause) or in conjunction with the **WITH DISTRIBUTION** clause (**On Dist Co1s** clause). The **ON ALL COLUMNS** specification is the default option if neither of the column specific clauses are specified.

If it is specified in the **On Co1s** clause, all columns will have only basic column statistics collected unless specific columns are chosen as part of the **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 Co1s** clause is redundant and therefore not necessary.

## ON COLUMNS

To collect statistics on specific columns, column groups, or both, use the **ON COLUMNS**. A column group is a parenthesized comma-separated list of columns for which you want to collect combined statistics.

The column and column groups are specified as a parenthesized comma-separated list.

When you run the **RUNSTATS** command on a table without gathering index statistics and specify a subset of columns for which statistics are to be gathered:

- Statistics for columns not specified in the **RUNSTATS** command but which are the first column in an index are not reset.
- Statistics for all other columns not specified in the **RUNSTATS** command are reset.

This clause can be used in the **On Co1s** clause and the **On Dist Co1s** 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 are ignored for collecting distinct values for the group. However, basic XML column statistics are collected for the XML type columns in the column group.

## EXCLUDING XML COLUMNS

Use this clause to omit all XML type columns from statistics collection. Using 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. For DB2 V9.7 Fix



Pack 1 and later releases, distribution statistics over XML type columns are not collected when this clause is specified.

#### **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 Col's` clause or the `On Dist Col's` clause. It is redundant in the `On Col's` 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 nonexistent 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.

For DB2 V9.7 Fix Pack 1 and later releases, distribution statistics for each index over XML data uses a maximum of 250 quantiles as the default. The default can be changed by specifying the **NUM\_QUANTILES** parameter in the **ON COLUMNS** or the **DEFAULT** clause. The **num\_quantiles** database configuration parameter is ignored while collecting XML distribution statistics.

#### **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.

## RUNSTATS

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**. The statistics are collected for columns of type **CHAR** and **VARCHAR** with a code page attribute of single-byte character set (**SBCS**), **FOR BIT DATA**, or **UTF-8**. 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 tables 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 tables 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 tables. *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 views whose definitions are a select over a single base table. If the view contains multiple tables, SYSTEM sampling is also possible if:

- the tables are joined using equality predicates on all the primary key and foreign key columns included in a referential integrity constraint defined between the tables,
- no search condition filters rows in any parent tables in the relationship, and
- a single child table, that is also not a parent table, can be identified among all the tables.

If the statistical view does not meet those conditions, Bernoulli sampling will be used instead and a warning will be returned (SQL2317W).

#### **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.

#### **INDEXSAMPLE BERNOLLI**

Use this option to collect index statistics on a sample of the rows in the index. *Bernoulli sampling* considers each row individually, including the row with probability  $P/100$  (where  $P$  is the value of the 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. Each execution of **RUNSTATS** is likely to yield a different sample of the index. All index pages are retrieved through an index scan but only the percentage of rows as specified through the numeric-literal parameter is used for the statistics collection. This option is not supported on statistical views.

#### **INDEXSAMPLE SYSTEM**

Use this option to collect statistics on a sample of the index pages. *System sampling* considers each page individually, including the page with probability  $P/100$  (where  $P$  is the value of the numeric-literal) and excluding it with probability  $1-P/100$ . Each execution of the **RUNSTATS** command usually yields a different sample of the index. The size of the sample is controlled by the numeric-literal parameters in parentheses, representing an approximate percentage  $P$  of the index to be returned. Only a percentage of the index pages as specified through the numeric-literal parameter is retrieved and used for the statistics collection. This option is not supported on statistical views.

## RUNSTATS

### *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 as an error (SQL1197N) by the DB2 database system.

### **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. You cannot use the **UPDATE PROFILE** option to remove clauses that are in a statistics profile.

### **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. You cannot use the **UPDATE PROFILE ONLY** option to remove clauses that are in a 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.

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.

## Examples

1. Collect statistics on the table only, on all columns without distribution statistics:

```
RUNSTATS ON TABLE employee
```

2. Collect statistics on the table only, on columns empid and empname with distribution statistics:  

```
RUNSTATS ON TABLE 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 employee WITH DISTRIBUTION DEFAULT
  NUM_FREQVALUES 50
```
4. Collect statistics on the table only, on all columns with distribution statistics, and on column group JOB, WORKDEPT, and SEX:  

```
RUNSTATS ON TABLE employee ON ALL COLUMNS AND COLUMNS ((JOB, WORKDEPT, SEX))
  WITH DISTRIBUTION
```
5. Collect statistics on a set of indexes:  

```
RUNSTATS ON TABLE employee for indexes emp11, emp12
```
6. Collect statistics on all indexes only:  

```
RUNSTATS ON TABLE employee FOR INDEXES ALL
```
7. Collect basic statistics on the table and all indexes using sampling for the detailed index statistics collection:  

```
RUNSTATS ON TABLE employee AND SAMPLED DETAILED INDEXES ALL
```

This is equivalent to:

```
RUNSTATS ON TABLE employee AND DETAILED INDEXES ALL
```

8. 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 employee
  WITH DISTRIBUTION ON COLUMNS (empid, empname, empdept NUM_FREQVALUES
    50 NUM_QUANTILES 100)
  DEFAULT NUM_FREQVALUES 5 NUM_QUANTILES 10
  AND INDEXES Xempid, Xempname
```
9. Collect statistics on all columns used in indexes and on all indexes:  

```
RUNSTATS ON TABLE employee ON KEY COLUMNS AND INDEXES ALL
```
10. Collect statistics on all indexes and all columns without distribution except for one column. Consider T1 containing columns c1, c2, ..., c8  

```
RUNSTATS ON TABLE 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 T1
  WITH DISTRIBUTION ON COLUMNS (c3 NUM_FREQVALUES 20 NUM_QUANTILES 40)
  AND INDEXES ALL
```
11. Collect statistics on table T1 for the individual columns c1 and c5 as well as on the column groups (c2, c3) and (c2, c4). Multicolumn 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 T1 ON COLUMNS (c1, (c2, c3),
  (c2, c4), c5)
```
12. Collect statistics on table T1 for the individual columns c1 and c2. For column c1 also collect the LIKE predicate statistics.  

```
RUNSTATS ON TABLE T1 ON COLUMNS (c1 LIKE STATISTICS, c2)
```

## RUNSTATS

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. The command also updates the statistics as specified.

```
RUNSTATS ON TABLE employee WITH DISTRIBUTION DEFAULT
  NUM_FREQVALUES 50 SET PROFILE
```
14. 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 employee WITH DISTRIBUTION
  DEFAULT NUM_FREQVALUES 50 SET PROFILE ONLY
```
15. 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 employee WITH DISTRIBUTION
  DEFAULT NUM_FREQVALUES 30 UPDATE PROFILE
```
16. Modify the previously registered statistics profile by changing the **NUM\_FREQVALUES** value from 50 to 30. Statistics are not collected.

```
RUNSTATS ON TABLE employee WITH DISTRIBUTION
  DEFAULT NUM_FREQVALUES 30 UPDATE PROFILE ONLY
```
17. 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 employee
  ON COLUMNS (empl_address, (empl_title, empl_salary))
  UPDATE PROFILE
```
18. 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 employee
  ON COLUMNS (empl_address, (empl_title, empl_salary))
  UPDATE PROFILE ONLY
```
19. Collect statistics on a table using the options recorded in the statistics profile for that table:

```
RUNSTATS ON TABLE employee USE PROFILE
```
20. Query the **RUNSTATS** command options corresponding to the previously registered statistics profile stored in the catalogs of the table:

```
SELECT STATISTICS_PROFILE FROM SYSCAT.TABLES WHERE TABNAME =
  'EMPLOYEE'
```
21. Collect statistics, including distribution statistics, on 30 percent of the rows:

```
RUNSTATS ON TABLE employee WITH DISTRIBUTION
  TABLESAMPLE BERNOULLI(30)
```
22. 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 employee WITH DISTRIBUTION
  TABLESAMPLE BERNOULLI(30) REPEATABLE(4196)
```

Issuing the preceding statement will result in the same set of statistics as long as the data has not changed in the interim.
23. 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 employee AND INDEXES ALL TABLESAMPLE SYSTEM(1.5)
24. Collect table statistics on 1.5 percent of the data pages and index statistics on 2.5 percent of the index pages. Both table data pages and index pages are sampled:
- ```
RUNSTATS ON TABLE employee AND INDEXES ALL TABLESAMPLE SYSTEM(1.5)
INDEXSAMPLE SYSTEM(2.5)
```
25. Collect statistics for a statistical view, on all columns, without distribution statistics:
- ```
RUNSTATS ON VIEW product_sales_view
```
26. 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 VIEW product_sales_view
WITH DISTRIBUTION ON COLUMNS (category NUM_FREQVALUES 100 NUM_QUANTILES 100,
type, product_key) DEFAULT NUM_FREQVALUES 50 NUM_QUANTILES 50
```
27. Collect statistics for a statistics view, including distribution statistics, on 10 percent of the rows using row level sampling:
- ```
RUNSTATS ON VIEW product_sales_view
WITH DISTRIBUTION TABLESAMPLE BERNOULLI (10)
```
28. Collect statistics for a statistics view, 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 issued.
- ```
RUNSTATS ON VIEW product_sales_view
WITH DISTRIBUTION TABLESAMPLE SYSTEM (2.5)
```
29. Register a statistics profile to collect statistics on the view and on all columns with distribution statistics as specified:
- ```
RUNSTATS ON VIEW product_sales_view
WITH DISTRIBUTION DEFAULT NUM_FREQVALUES 50 NUM_QUANTILES 50
SET PROFILE
```
30. Modify the previously registered statistics profile. This command also updates the statistics as specified:
- ```
RUNSTATS ON VIEW salesdb.product_sales_view
WITH DISTRIBUTION DEFAULT NUM_FREQVALUES 25 NUM_QUANTILES 25
UPDATE PROFILE
```

## Usage notes

- 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 SYSCAT.DATAPARTITIONS catalog view and look for the value L (logically detached), I (index cleanup), or D (detached with dependent MQT).

The **RUNSTATS** command collects statistics for all index partitions of a partitioned index. Statistics in the SYSSTAT.INDEXES view for the partitioned index represent an index partition, except for FIRSTKEYCARD, FIRST2KEYCARD, FIRST3KEYCARD, FIRST4KEYCARD, and FULLKEYCARD statistics. Because these statistics are used in cardinality estimates, they are for the entire index and not for an index partition. Distribution statistics (frequent values and quantiles) are not collected for partitioned indexes, but are gathered if **RUNSTATS** is run on the table. Statistics on the leading columns of a partitioned index might not be as accurate as statistics on the leading columns of a nonpartitioned index.

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 time to collect statistics is very limited and index statistics are to be gathered, consider using the **INDEXSAMPLE** option to collect statistics on a subset of the index data.
  - 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.
  6. For DB2 V9.7 Fix Pack 1 and later releases, distribution statistics are collected on indexes over XML data defined on an XML column. When the **RUNSTATS**

command is run on a table with the **WITH DISTRIBUTION** clause, the following apply to the collection of distribution statistics on a column of type XML:

- Distribution statistics are collected for each index over XML data specified on an XML column.
- The **RUNSTATS** command must collect both distribution statistics and table statistics to collect distribution statistics for indexes over XML data defined on an XML column. Table statistics must be gathered in order for distribution statistics to be collected since XML distribution statistics are stored with table statistics.

An index clause is not required to collect XML distribution statistics.

Specifying only an index clause does not collect XML distribution statistics

By default, XML distribution statistics use a maximum of 250 quantiles for each index over XML data. When collecting distribution statistics on an XML column, you can change the maximum number of quantiles by specifying a value with **NUM\_QUANTILES** parameter in the **ON COLUMNS** or the **DEFAULT** clause.

- Distribution statistics are collected for indexes over XML data of type VARCHAR, DOUBLE, TIMESTAMP, and DATE. Distribution statistics are not collected over indexes of type VARCHAR HASHED.
  - Distribution statistics are not collected for partitioned indexes over XML data defined on a partitioned table.
7. 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.
  8. 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.
  9. 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
  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 SYSCAT.TABLES 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 statistics 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.

13. Statistics are not collected for columns with structured types. If they are specified, columns with these data types are ignored.
14. Only AVGCOLLEN and NUMNULLS are collected for columns with LOB or LONG data types.
15. AVGCOLLEN represents the average space in bytes when the column is stored in database memory or a temporary table. This value represents the length of the data descriptor for LOB or LONG data types, except when LOB data is inlined on the data page.

**Note:** The average space required to store the column on disk may be different than the value represented by this statistic.

16. The **UNSAMPLED DETAILED** option is available to change the way index statistics are collected, but it should be used only in cases where its clear that the default or **DETAILED** doesnt work.
17. When using the **INDEXSAMPLE** keyword you cannot specify different index sampling rates for different indexes within a single command. For example:  

```
runstats on table orders and index o_ck indexsample system(5),
      index o_ok indexsample system(10)
```

is invalid. The following two **RUNSTATS** commands can be used to achieve the required result:

```
runstats on table orders and index o_ck indexsample system(5)
runstats on table orders for index o_ok indexsample system(10)
```

**Related information:**

## SET CLIENT

Specifies connection settings for the back-end process.

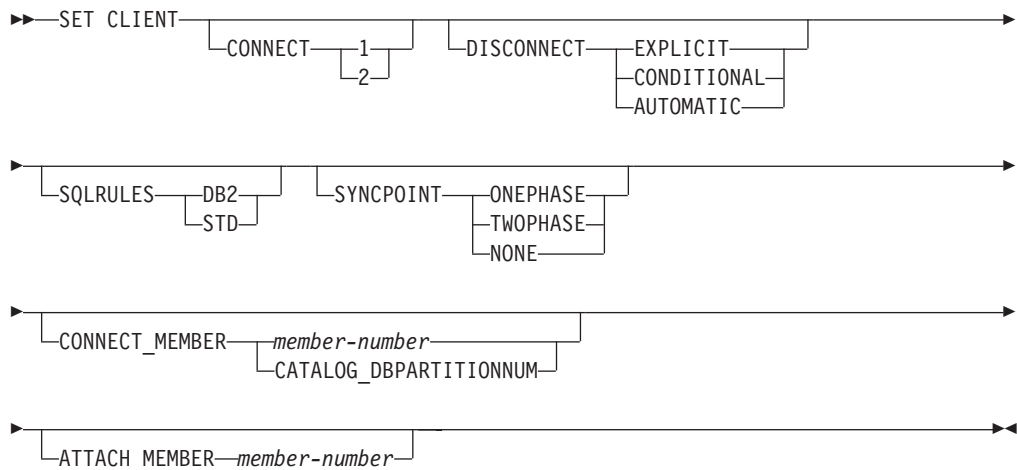
### Authorization

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.

#### 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\_MEMBER (partitioned database or DB2 pureScale environments)**

*member-number*

Specifies the member to which a connect is to be made. For a partitioned database environment, valid values are between zero and 999, inclusive; for a DB2 pureScale environment, valid values are between 0 and 127, inclusive. Overrides the value of the **DB2NODE** environment variable.

#### **CATALOG\_DBPARTITIONNUM**

In a partitioned database environment, 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\_MEMBER *member-number* (partitioned database or DB2 pureScale environments)**

Specifies the member to which an attach is to be made. For a change to **ATTACH\_MEMBER** to take effect, you must first detach from the instance using the **DETACH** command, then attach to the instance using the **ATTACH** command. For a partitioned database environment, valid values are between zero and 999, inclusive; for a DB2 pureScale environment, valid values are between 0 and 127, inclusive. Overrides the value of the **DB2NODE** environment variable .

## **Examples**

To set specific values:

```
db2 set client connect 2 disconnect automatic sqlrules std
syncpoint twophase
```

To change **SQLRULES** back to DB2 rules, but keep the other settings:

```
db2 set client sqlrules db2
```

The connection settings revert to default values after the **TERMINATE** command is issued.

## SET CLIENT

### 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.

In partitioned databases or DB2 pureScale environments, the connection settings could have an impact on acquiring trusted connections. For example, if the **CONNECT\_MEMBER** option is set to a node such that the establishment of a connection on that node requires going through an intermediate node (a hop node), it is the IP address of that intermediate node and the communication protocol used to communicate between the hop node and the connection node that are considered when evaluating this connection in order to determine whether or not it can be marked as a trusted connection. In other words, it is not the original node from which the connection was initiated that is considered. Rather, it is the hop node that is considered.

### Compatibilities

For compatibility with previous versions:

- **CONNECT\_DBPARTITIONNUM** or **CONNECT\_NODE** can be substituted for **CONNECT\_MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.
- **CATALOG\_NODE** can be substituted for **CATALOG\_DBPARTITIONNUM**.
- **ATTACH\_DBPARTITIONNUM** or **ATTACH\_NODE** can be substituted for **ATTACH\_MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.



## SET RUNTIME DEGREE

of intra-partition parallelism for an SQL statement can be specified at statement compilation time using the CURRENT DEGREE special register or the **DEGREE** bind option. The maximum run time degree of intrapartition 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.

The value in the CURRENT DEGREE special register and the **intra\_parallel** setting can be overridden in a workload by setting the MAXIMUM DEGREE workload attribute.



## SET SERVEROUTPUT

Specifies whether output from the DBMS\_OUTPUT message buffer is redirected to standard output.

### Authorization

EXECUTE privilege on the DBMS\_OUTPUT module.

### Required connection

Database

### Command syntax

```

▶▶ SET SERVEROUTPUT { OFF | ON }

```

### Command parameters

**ON** Specifies that messages in the message buffer are redirected to standard output.

#### OFF

Specifies that messages in the message buffer are not redirected to standard output.

### Examples

To redirect messages in the DBMS\_OUTPUT message buffer to standard output, specify SET SERVEROUTPUT ON. In this example, the PUT procedure adds partial lines to the DBMS\_OUTPUT message buffer. When proc1 runs, because SET SERVEROUTPUT ON is specified, the text stored in the DBMS\_OUTPUT message buffer is displayed.

```

SET SERVEROUTPUT ON@

DROP PROCEDURE proc1@

CREATE PROCEDURE proc1()
BEGIN
  CALL DBMS_OUTPUT.PUT( 'p1 = ' || p1 );
  CALL DBMS_OUTPUT.PUT( 'p2 = ' || p2 );
  CALL DBMS_OUTPUT.NEW_LINE;
END@

CALL proc1( 10, 'Peter' )@

SET SERVEROUTPUT OFF@

```

This example results in the following output:

```

SET SERVEROUTPUT ON
DB20000I The SET SERVEROUTPUT command completed successfully.

DROP PROCEDURE PROC1
DB20000I The SQL command completed successfully.

CREATE PROCEDURE proc1()
BEGIN
  CALL DBMS_OUTPUT.PUT( 'p1 = ' || p1 );
  CALL DBMS_OUTPUT.PUT( 'p2 = ' || p2 );

```

## SET SERVEROUTPUT

```
CALL DBMS_OUTPUT.NEW_LINE;
END@
DB20000I The SQL command completed successfully.

CALL procl( 10, 'Peter' )@

Return Status = 0

p1 = 10
p2 = Peter

SET SERVEROUTPUT OFF
DB20000I The SET SERVEROUTPUT command completed successfully.
```

### Usage notes

Messages are added to the DBMS\_OUTPUT message buffer by the PUT, PUT\_LINE, and NEW\_LINE procedures.

When the command SET SERVEROUTPUT ON executes, it calls the DBMS\_OUTPUT.ENABLE procedure with the default buffer size of 20000 bytes and sets an internal flag in the command line processor (CLP) or command line processor plus (CLPPlus). When this flag is enabled, the application calls the GET\_LINES procedure after executing each SELECT or CALL statement, and redirects the messages from the message buffer to standard output. To increase the DBMS\_OUTPUT buffer size, call DBMS\_OUTPUT.ENABLE procedure with a larger buffer size after executing SET SERVER OUTPUT ON, for example: CALL DBMS\_OUTPUT.ENABLE( 50000 );

When the command SET SERVEROUTPUT OFF executes: it calls the DBMS\_OUTPUT.DISABLE procedure, messages that are in the message buffer are discarded, and calls to PUT, PUT\_LINE, and NEW\_LINE procedures are ignored. The DBMS\_OUTPUT.GET\_LINES procedure will not be called after each SELECT or CALL statement.

# SET TABLESPACE CONTAINERS

Sets the table space container during a redirected restore operation.

A redirected restore is a restore where the storage (table space containers or storage group paths) for the restored database is different from the storage for the source database at the time the backup was done. Using this command you can add, change, or remove table space containers during a redirected restore operation. If, for example, one or more containers become inaccessible for any reason, the restore fails if it is not redirected to different table space containers.

This command can be used to convert existing regular or large database managed table spaces to use automatic storage. It can also be used to re-stripe existing automatic storage table spaces more evenly over the storage paths available to the database.

### Authorization

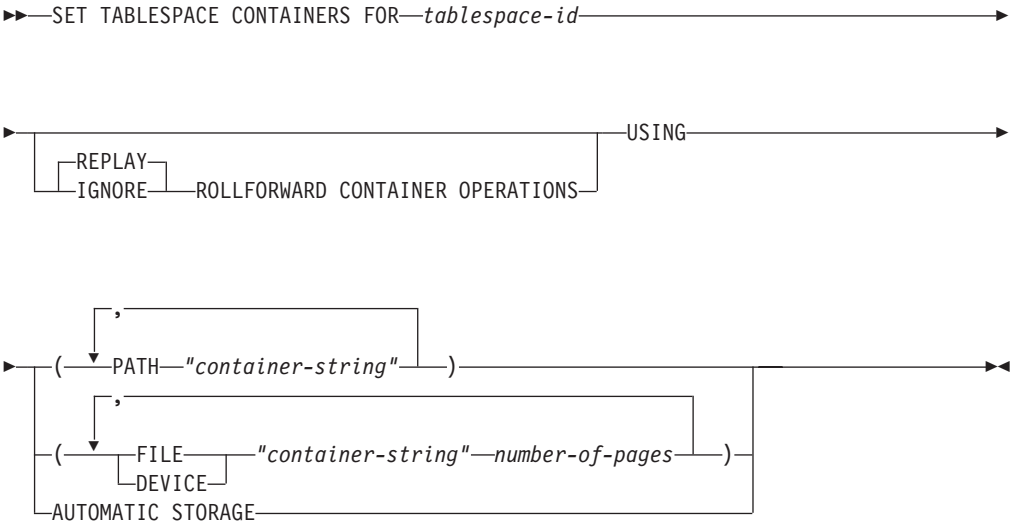
One of the following authorities:

- 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.

## SET TABLESPACE CONTAINERS

### IGNORE ROLLFORWARD CONTAINER OPERATIONS

Specifies that ALTER TABLESPACE operations in the log are to be ignored when performing a roll forward.

### USING PATH *"container-string"*

For an SMS table space, identifies one or more containers that will belong to the table space and into which the table space data will be stored. It is an absolute or relative directory name. If the directory name is not absolute, it is relative to the database directory. The string cannot exceed 240 bytes in length.

### USING FILE | DEVICE *"container-string" number-of-pages*

For a DMS table space, identifies one or more containers that will belong to the table space and into which the table space data will be stored. The container type (either **FILE** or **DEVICE**) and its size 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.

### USING AUTOMATIC STORAGE

Specifies that the table space should be converted to use automatic storage and will be associated with the default storage group. The database will create new containers on the storage paths of the default storage group. Once a table space has been redirected to use automatic storage, no container operations can be applied to the table space.

This option can be used to provide better striping across existing storage paths by redefining the containers of table spaces that are already managed by automatic storage.

**Note:** The table space will be offline while being restored.

This option is not supported for system managed table spaces.

This is the only **USING** clause that can be specified in a DB2 pureScale environment.

## 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.

The **IGNORE/REPLAY ROLLFORWARD CONTAINER OPERATIONS** option is ignored when specified with the **USING AUTOMATIC STORAGE** option.

A redirected restore of a table space in a multi-partition environment using the **USING AUTOMATIC STORAGE** option of the **SET TABLESPACE CONTAINERS** command will only convert the table space to automatic storage on the partition being restored. It will not redefine the containers on any other database partition.

By not redefining the containers on other database partitions, the definition of the table space differs on each partition. Later, when adding a database partition, use the **ADD DBPARTITIONNUM** command with the **LIKE DBPARTITIONNUM** option. Depending on the database partition chosen in this option, the new database partition will have either the table space defined with automatic storage or the table space defined without automatic storage. To remove both the inconsistency in the definitions of the table spaces and the need to decide between the definitions each time a new database partition is added, ensure that the table space definition is the same on all database partitions. For example, if all of the database partitions were subject to a redirected restore followed by using the **USING AUTOMATIC STORAGE** option of the **SET TABLESPACE CONTAINERS** command, then the table space will be converted to automatic storage on all the database partitions. Adding another database partition later will have the same definition for the table space as that found on the other database partitions.

In a partitioned database environment, if a single table space had been redirected on only a subset of the database partitions using the **SET TABLESPACE CONTAINERS** command to alter storage types, then a single table space can be defined as DMS on some database partitions, automatic storage on other database partitions, and a combination of DMS and automatic storage on yet another database partition.

Once you restore a table space, run a subsequent **ALTER TABLESPACE** statement to update the table space's storage group ID in the catalog views.

---

# 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 authorities:

- 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.

## 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 authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON

### Required connection

Instance. If there is more than one partition on the local machine, the attachment should be made to the correct partition. For example, suppose there are two partitions and a **LIST UTILITIES** command resulted in the following output:

```
ID = 2
Type = BACKUP
Database Name = IWZ
Partition Number = 1
Description = online db
Start Time = 07/19/2007 17:32:09.622395
State = Executing
Invocation Type = User
Throttling:
Priority = Unthrottled
Progress Monitoring:
Estimated Percentage Complete = 10
Total Work = 97867649689 bytes
Completed Work = 10124388481 bytes
```

The instance attachment must be made to partition 1 in order to issue a **SET UTIL\_IMPACT\_PRIORITY** command against the utility with ID 2. To do this, set DB2NODE=1 in the environment and then issue the instance attachment command.

### 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

## SET UTIL\_IMPACT\_PRIORITY

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
```

### Relationship between UTIL\_IMPACT\_LIM and UTIL\_IMPACT\_PRIORITY settings

The database manager configuration parameter **util\_impact\_lim** sets the limit on the impact throttled utilities can have on the overall workload of the machine. 0-99 is a throttled percentage, 100 is no throttling.

The **SET UTIL\_IMPACT\_PRIORITY** command sets the priority that a particular utility has over the resources available to throttled utilities as defined by the **util\_impact\_lim** configuration parameter. (0 = unthrottled)

Using the backup utility as an example, if the **util\_impact\_lim**=10, all utilities can have no more than a 10% average impact upon the total workload as judged by the throttling algorithm. Using two throttled utilities as an example:

- Backup with **util\_impact\_priority** 70
- Runstats with **util\_impact\_priority** 50

Both utilities combined should have no more than a 10% average impact on the total workload, and the utility with the higher priority will get more of the available workload resources. For both the backup and **RUNSTATS** operations, it is also possible to declare the impact priority within the command line of that utility. If you do not issue the **SET UTIL\_IMPACT\_PRIORITY** command, the utility will run unthrottled (irrespective of the setting of **util\_impact\_lim**).

To view the current priority setting for the utilities that are running, you can use the **LIST UTILITIES** command.

### Usage notes

Throttling requires having an impact policy defined by setting the **util\_impact\_lim** configuration parameter.



## SET WORKLOAD

Specifies the workload to which the database connection is to be assigned. This command can be issued before connecting to a database or it can be used to reassign the current connection once the connection has been established. If the connection has been established, the workload reassignment will be performed at the beginning of the next unit of work.

### Authorization

None, but see usage notes

### Required connection

None

### Command syntax

```

▶▶ SET WORKLOAD TO {
    AUTOMATIC
    SYSDEFAULTADMWORKLOAD
}

```

### Command parameters

#### AUTOMATIC

Specifies that the database connection will be assigned to a workload chosen by the workload evaluation that is performed automatically by the server.

#### SYSDEFAULTADMWORKLOAD

Specifies that the database connection will be assigned to the `SYSDEFAULTADMWORKLOAD`, allowing users with *accessctrl*, *dataaccess*, *wlmadm*, *secadm* or *dbadm* authority to bypass the normal workload evaluation.

### Examples

To assign the connection to the `SYSDEFAULTADMWORKLOAD`:

```
SET WORKLOAD TO SYSDEFAULTADMWORKLOAD
```

To reset the workload assignment so that it uses the workload that is chosen by the workload evaluation performed by the server:

```
SET WORKLOAD TO AUTOMATIC
```

### Usage notes

If the session authorization ID of the database connection does not have *accessctrl*, *dataaccess*, *wlmadm*, *secadm* or *dbadm* authority, the connection cannot be assigned to the `SYSDEFAULTADMWORKLOAD` and an SQL0552N error will be returned. If the **SET WORKLOAD TO SYSDEFAULTADMWORKLOAD** command is issued before connecting to a database, the SQL0552N error will be returned after the database connection has been established, at the beginning of the first unit of work. If the command is issued when the database connection has been established, the SQL0552N error will be returned at the beginning of the next unit of work, when the workload reassignment is supposed to take place.

## SET WRITE

The **SET WRITE** command allows a user to suspend I/O write operations or to resume I/O write operations 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. This state guarantees that the existing write operations are completed and no new write operations can be performed. All table spaces need not be in `NORMAL` state for the command to execute successfully.

### Scope

This command only affects the database partition where the command is issued. In partitioned database environments, you must issue it on all the database partitions. In DB2 pureScale environments, you can issue it from any member to suspend I/O write operations for all the members, or to resume I/O write operations for all the suspended members.

### Authorization

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

Suspends the I/O write operations, including writing to the logs. All database operations, apart from online backup and restore, function normally while I/O write operations are suspended. However, some operations might wait while attempting to flush dirty pages from the buffer pool or log buffers to the logs. These operations continue after you resume the I/O write operations for the database.

#### RESUME

Resumes the I/O write operations. In DB2 pureScale environments, this parameter resumes the I/O write operations for all suspended members.

## Usage notes

You can determine whether the I/O write operations are suspended by viewing the setting of the **suspend\_io** database configuration parameter. To view database configuration information, you can use the **GET DATABASE CONFIGURATION** command, the DBCFG administrative view, the **db2pd** command with the **-dbcfg** parameter, or the **db2CfgGet** API.

You can use the **FLUSH BUFFERPOOLS** statement before using the **SET WRITE** command to minimize the recovery time of a split-mirror database. Issuing the statement can be useful if you plan on using a split-mirror database as a backup image or if you plan to create a backup image using the split-mirror database.

A connection attempt will fail if dirty pages must be flushed from the buffer pool to disk but it is impossible to resume I/O write operations by using the **SET WRITE** command. To resolve the connection failure, issue the **RESTART DATABASE** command with the **WRITE RESUME** parameter. In this scenario, the **RESTART DATABASE** command resumes write operations without performing crash recovery. The **RESTART DATABASE** command with the **WRITE RESUME** parameter performs crash recovery only when you use the command after a database crash.

The table spaces can be in transient states such as **SQLB\_MOVE\_IN\_PROGRESS** or **SQLB\_BACKUP\_IN\_PROGRESS** for this command to succeed. Note that **REBAL\_IN\_PROGRESS** is another state that snapshot monitor might report when database is suspended.

## START DATABASE MANAGER

Starts the database manager on the target member or all members. In a DB2 pureScale environment, this command also starts the cluster caching facility (CF).

This command is not valid on a client.

### Scope

In a multi-partitioned database environment, this command affects all database partitions that are listed in the \$HOME/sql1lib/db2nodes.cfg file, unless the **DBPARTITIONNUM** parameter is used.

### Authorization

One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT

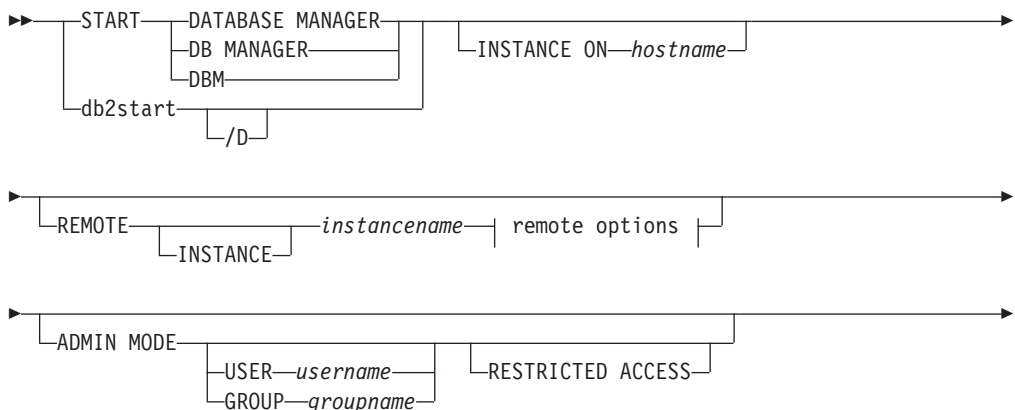
The **ADD DBPARTITIONNUM** start option requires either SYSADM or SYSCTRL authority.

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.

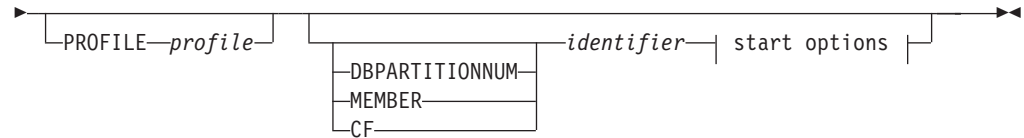
### Required connection

None

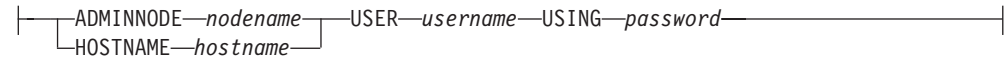
### Command syntax



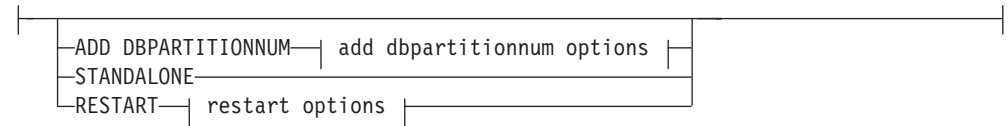
## START DATABASE MANAGER



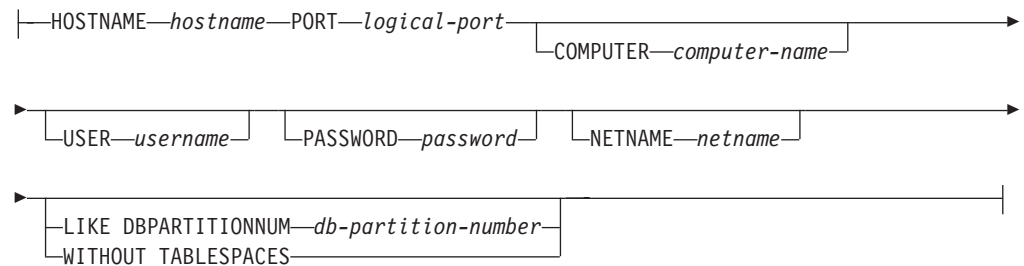
### remote options:



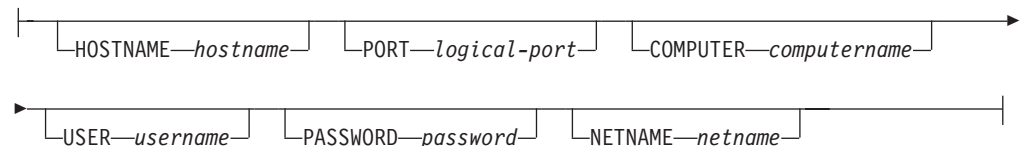
### start options:



### add dbpartitionnum options:



### restart options:



## Command parameters

**/D** Allows the DB2 product installation on Windows to be run as a process. Note that you cannot use the **/D** parameter to start a DB2 instance as a process in a partitioned database environment.

### **INSTANCE ON** *hostname*

Specifies to start the DB2 instance on a particular host of the DB2 pureScale instance after maintenance operations have been completed. If used outside of a DB2 pureScale environment, an SQL1695N error is returned.

### **REMOTE** [**INSTANCE**] *instancename*

Specifies the name of the remote instance you want to start.

## START DATABASE MANAGER

**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 **USER**, specifies the password of the user.

### ADMIN MODE

Starts the instance in quiesced mode for administration purposes. This parameter 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.

If the **ADMIN MODE** option is specified alone, the databases within the quiesced instance will be activated to do authorization checking for all connect attempts to the database. This is necessary to determine if the connecting user ID has DBADM authority; this authority is stored in the database catalog and the database must be activated to determine if the user ID has it. To prevent this authorization checking from happening, specify the **RESTRICTED ACCESS** option.

**USER** *username*

With **ADMIN MODE**, specifies the name of the user.

**GROUP** *groupname*

With **ADMIN MODE**, specifies the name of the group.

### RESTRICTED ACCESS

Specify this option to prevent authorization checking for all connect attempts to the databases of a quiesced instance to determine if the user ID has DBADM authority. Instance-level authorization checking can still occur; checking a user ID for SYSADM, SYSCTRL, or SYSMAINT authority does not require a database to be activated.

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.

*identifier*

Specifies the numeric identifier without having to specify the **DBPARTITIONNUM**, **MEMBER**, or **CF** parameters.

**DBPARTITIONNUM** *identifier*

This parameter option is valid only in a partitioned database environment.

Specifies the member to be started. If no identifier is specified, a normal startup is done on the local database partition. Valid values are from 0 to 999 inclusive.

**MEMBER** *identifier*

Specifies the member to be started. In a DB2 pureScale environment, valid values are from 0 to 127 inclusive.

**CF** *identifier*

This parameter option is valid only in a DB2 pureScale environment. Specifies the cluster caching facility (CF) to be started. Valid values are 128 and 129. If used outside of a DB2 pureScale environment, an SQL1695N error is returned.

**ADD DBPARTITIONNUM**

Specifies that the new database partition server is added to the `db2nodes.cfg` file of the instance owner with the **HOSTNAME** and **PORT** parameter values. This option is valid in only a partitioned database environment.

Ensure that the combination of the **HOSTNAME** and **PORT** parameter values is unique.

The add database partition server utility is executed internally to create all existing databases on the database server partition being added. The new database partition server is automatically added to the `db2nodes.cfg` file.

**Note:** Any uncataloged database is not recognized when adding a new database partition. The uncataloged database will not be present on the new database partition. An attempt to connect to the database on the new database partition returns the error message SQL1013N.

If the ADD request is made in an environment that has two or more active database partition servers, the new database partition server is visible to the environment when the ADD processing completes.

If the ADD request is made in an environment that has one database partition server and it is active, after ADD processing completes, the new database partition server is inactive. The instance must be restarted by using **db2stop** and **db2start** before the new database partition server can participate in the partitioned database environment. If the ADD request is made in an environment that has one database partition server and it is inactive, after ADD processing completes, the new database partition server (or servers, if more than one is added) is active. Only the original database partition server needs to be started.

A newly added database partition is configured during ADD processing as follows:

1. In a multi-partition environment, the new database partition is configured using the database configuration parameter values from a noncatalog database partition.
2. In a single-partition environment, the new database partition is configured using the database configuration parameter values from the catalog partition.
3. If a problem occurs in copying the database configuration parameter values to the new database partition, the new database partition is configured using the default database configuration parameter values.

## START DATABASE MANAGER

**HOSTNAME** *hostname*

With the **ADD DBPARTITIONNUM** parameter, 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 the **HOSTNAME** parameter.

**LIKE DBPARTITIONNUM** *db-partition-number*

Specifies that the containers for the system temporary table spaces are the same as the containers on the specified `db-partition-number` value 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 use automatic storage, the containers might not necessarily match those containers from the partition specified. Instead, containers are automatically assigned by the database manager based on the storage paths that are associated with the database. This automatic assignment might 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 (this refers to 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 are automatically assigned by the database manager based on the storage paths that are associated with the database.

**STANDALONE**

This option is valid only in a partitioned database environment. 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

This option is valid only in a partitioned database environment. Starts the database manager after a failure. Other database partitions are still operating, and this database partition attempts to connect to the others. If the **HOSTNAME** and the **LOGICAL-PORT** parameters are not specified, the database manager is restarted using the host name and port values specified for the partition in `db2nodes.cfg`. If either value 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*

You can use the **HOSTNAME** parameter with the **RESTART** parameter to restart a database partition on a different machine than is specified in the database partition configuration file, `db2nodes.cfg`.

#### **Restriction:**

When you are using the DB2 High Availability Feature, do not use the **HOSTNAME** option with the **RESTART** parameter to restart a database partition on a different machine. To restart or move a database partition from one machine in a DB2 pureScale instance to another machine, use DB2 High Availability Instance Configuration Utility (**db2haicu**).

#### **PORT** *logical-port*

With **RESTART**, specifies the logical port number to be used to override that in the database partition configuration file. If not specified, this parameter defaults to the logical port number for the database partition as specified 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 a *netname* value to override that specified in the `db2nodes.cfg` file. If not specified, this parameter defaults to the *netname* value that corresponds to the logical database partition number in the `db2nodes.cfg` file.

## Examples

The following example is sample output from **db2start** issued against a DB2 instance with members 10, 20, and 30:

## START DATABASE MANAGER

```
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

On Microsoft Windows Vista or later versions, when using an id that has local or domain administrator authority, you must execute this command from a DB2 command window running with full administrator privileges.

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 database partitions 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
```

The **RESTRICTED ACCESS** option will prevent the databases within the quiesced instance from being activated to do authorization checking. Any user ID trying to connect to a database, which has DBADM authority or QUIESCE\_CONNECT privilege on the database, will not be allowed to connect. Only user IDs which have SYSADM, SYSCTRL, or SYSMAINT authority and the user or group specified with the command will be allowed to connect to the database.

The **RESTRICTED ACCESS** option should be used when there is a need to have exclusive connections to a database within the quiesced instance. Such cases can include making an offline backup or performing other maintenance activities.

When adding a new database partition server, **START DATABASE MANAGER** must determine whether 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.

A new database partition server cannot be added when any of the following commands, statements, or operations are in progress. Otherwise SQL6074N is returned.

- **QUIESCE INSTANCE**
- **UNQUIESCE INSTANCE**
- **STOP DB2 (db2stop)**
- **STOP DATABASE MANAGER DBPARTITIONNUM**
- **START DB2 (db2start)**
- **START DATABASE MANAGER DBPARTITIONNUM**
- **START DATABASE MANAGER** with restart options
- **CREATE DATABASE**
- **DROP DATABASE**
- **QUIESCE DATABASE**
- **UNQUIESCE DATABASE**
- **ACTIVATE DATABASE**
- **DEACTIVATE DATABASE**
- A Z lock on a database object
- Backing up the database on all database partition servers
- Restoring the database
- **ALTER, ALTER, or DROP** of a table space
- Updating of automatic storage paths

On UNIX operating systems, 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 to the `$HOME/sql1lib/log/db2start.timestamp.log` error log file. Members that are already started are not affected. If CTRL+C is issued on a member that is starting, **db2stop** must be issued on that member before an attempt is made to start it again.

On Windows operating systems, the **db2start** command and the **NET START** command do not return 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 previous versions:

## START DATABASE MANAGER

- **LIKE DBPARTITIONNUM** or **LIKE NODE** can be substituted for **LIKE MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.
- **ADD DBPARTITIONNUM** or **ADD NODE** can be substituted for **ADD MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.
- **DBPARTITIONNUM** or **NODENUM** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON. If **DBPARTITIONNUM** is specified in a DB2 pureScale environment and the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON, an SQL1694N error is returned.

## START HADR

Starts HADR operations for a database.

### Authorization

One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT

### 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

```

▶▶ START HADR ON DATABASE database-alias
                   |
                   | DB
                   |
▶ USER user-name AS PRIMARY
                   |
                   | USING password
                   |
                   | STANDBY BY FORCE
                   |
▶▶

```

### 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. If the primary database cannot connect to the HADR standby database within the time specified by the **hadr\_timeout** database configuration parameter, the primary will not start.

**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

## START HADR

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 <b>START HADR</b> command with the <b>AS PRIMARY</b> option | Behavior upon <b>START HADR</b> command with the <b>AS STANDBY</b> 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: SQL1767N, SQL1769N, or SQL1770N 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 information:**

## STOP DATABASE MANAGER

Stops the database manager on the target member or all members. In a DB2 pureScale environment, it is also used to stop the cluster caching facility (CF).

### Scope

In a DB2 pureScale environment or, a partitioned database environment, this command affects all members that are listed in the `db2nodes.cfg` file, unless either a **DBPARTITIONNUM**, **MEMBER**, or **CF** parameter is specified.

The command does not shut down the member if there are active database connections. If there are no active database connections, but there are instance attachments, then the instance attachments are forced off first and then the member is stopped. This command also deactivates any outstanding database activations before stopping the member. This command can also be used to stop instance services on a given host.

This command is not valid on a client.

### Authorization

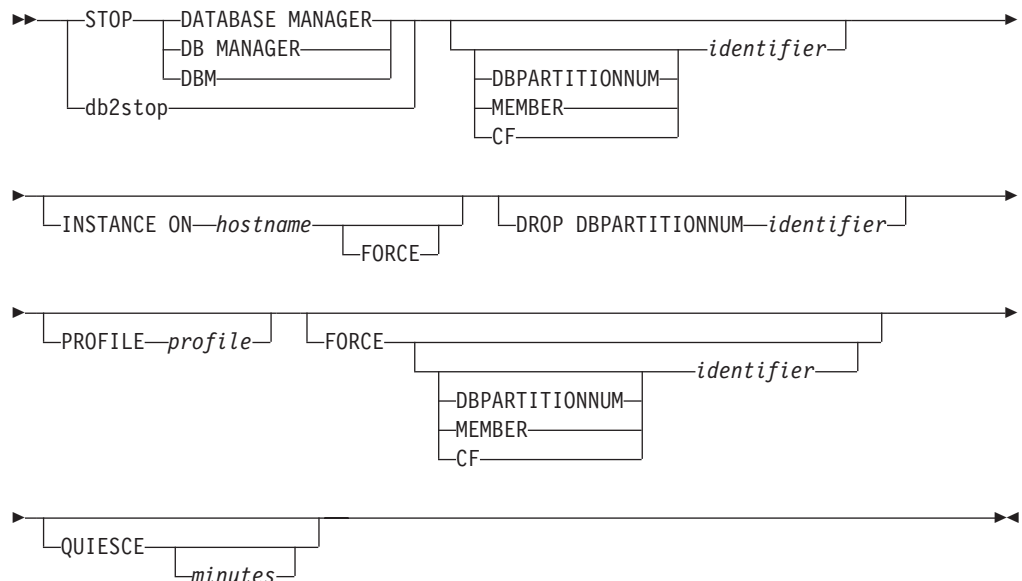
One of the following authorities:

- SYSADM
- SYSCtrl
- SYSMAINT

### Required connection

None

### Command syntax



## STOP DATABASE MANAGER

### Command parameters

#### **INSTANCE ON** *hostname*

Specifies to stop the DB2 instance on a particular host of the DB2 pureScale instance for maintenance operations. This command temporarily prevents the host from being restarted with the global **db2start** command. The host can no longer act as guest host for a restart light situation. To restart the DB2 instance on the host after maintenance operations have been completed, issue the **START DBM INSTANCE ON** *hostname* command. This command fails if active members or active cluster caching facilities (CFs) are running on the host. If used outside of a DB2 pureScale environment, an SQL1695N error is returned.

**FORCE** When specified, the **FORCE** parameter immediately stops the DB2 instance on the host. Any active member running on the host fails over to other active hosts, or any active cluster caching facility (CF) is forced to stop. A member that is failed over to another host is known as a *restart light member*, meaning that it uses minimal resources on the host and does not accept database connections. For more information, see "Restart light."

#### *identifier*

Specifies the numeric identifier without having to specify the **DBPARTITIONNUM**, **MEMBER**, or **CF** parameters.

#### **DBPARTITIONNUM** *identifier*

This parameter option is valid only in a partitioned database environment. Specifies the member to be stopped. Valid values are from 0 to 999 inclusive.

**Note:** If the **FORCE** option is used without this parameter, all applications on all database partitions are forced before all the database partitions are stopped.

**Note:** The *identifier* value must 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 stopped.

#### **MEMBER** *identifier*

Specifies the member to be stopped. In a DB2 pureScale environment, valid values are from 0 to 127 inclusive.

#### **CF** *identifier*

This parameter option is valid only in a DB2 pureScale environment. Specifies the cluster caching facility (CF) to be stopped. Valid values are 128 and 129. If used outside of a DB2 pureScale, an SQL1695N error is returned.

#### **PROFILE** *profile*

This parameter option is valid only in a partitioned database environment. 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 `sql1ib` directory of the instance owner.

#### **DROP DBPARTITIONNUM** *identifier*

This parameter option is valid only in a partitioned database environment. Specifies the database partition to be dropped from the `db2nodes.cfg` configuration file.



**Note:** Before using the **DROP DBPARTITIONNUM** parameter, run the **DROP DBPARTITIONNUM VERIFY** command to ensure that there is no user data on this database partition environment.

**FORCE** Specifies to use **FORCE APPLICATION ALL** when stopping the database manager at each database partition.

**QUIESCE** *minutes*

This parameter option is valid only in a DB2 pureScale environment and valid only for the **MEMBER** parameter clause. It is used to stop active workload on a given member before the member is shut down (it cannot be used against a cluster caching facility). As opposed to a **db2stop FORCE** command where active applications are interrupted immediately, quiesce affords them a graceful exit by waiting until all applications on the member have completed their active transactions. The optional *minutes* parameter value specifies how long the command waits before it tells applications to disconnect from the given member. Once this timeout is reached, any active units of work remaining at the time are interrupted. If no timeout is specified the command waits indefinitely until all active applications have ended their unit of work before proceeding with shutting down the member.

**Important:** To bring the member back online, you need to issue the **db2start** command against the member.

Valid values for the *minutes* parameter value are from -1 to 1440 inclusive. The default value is -1 which means the specified member waits indefinitely until all active workload has ended. If a value of 0 is specified then active applications are interrupted immediately and, once they end, the member is shut down.

**Note:** A **db2stop QUIESCE** command is not interruptible. After you issue the command, no further DB2 commands may be run directly against that member. If you want to query information about the quiesced member, you need to issue the **LIST APPLICATIONS GLOBAL SHOW DETAIL** command from another active member. If you want to stop an application that is running on a quiesced member, you need to issue the **FORCE APPLICATION** '*<app handle>*' command from another active member.

### Examples

The following example is sample output from **db2stop** issued against a DB2 instance with members 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

On Microsoft Windows Vista or later versions, you must execute this command from a DB2 command window running with full administrator privileges.

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.

## STOP DATABASE MANAGER

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.

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 environment since the previous **STOP DATABASE MANAGER** command was issued are updated in the `db2nodes.cfg` file.
- On UNIX operating systems, 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:** Do not use the UNIX **kill** command to terminate the database manager because the command abruptly ends database manager processes without controlled termination and cleanup processing.

## STOP HADR

Stops HADR operations for a database.

### Authorization

One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT

### 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 <b>hadr_db_role</b> 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 <b>START HADR</b> command with the <b>AS PRIMARY</b> option is issued. Open sessions and transactions are not affected by the <b>STOP HADR</b> command. You can repeatedly issue <b>STOP HADR</b> and <b>START HADR</b> commands while the database remains online. These commands take effect dynamically. |
| Inactive standby database | Database role changes to standard. Database configuration parameter <b>hadr_db_role</b> 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: SQL1767N, SQL1769N, or SQL1770N 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 information:

## 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 authorities:

- SYSADM
- SYSCTRL
- SYSMANT

### 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
  |
  |-----|
  | PEER WINDOW ONLY
  |
  <----->

```

### 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.

#### PEER WINDOW ONLY

When this option is specified, there will not be any committed transaction loss if the command succeeds and the primary database is brought down before the end of the peer window period (set the database configuration parameter **hadr\_peer\_window** to a non-zero value). Not bringing down the primary database, before the peer window expires, will result in *split brain*. If the **TAKEOVER BY FORCE PEER WINDOW ONLY** command is executed when the HADR pair is not in a peer or disconnected peer state (the peer window has expired), an error is returned.

You cannot use the **PEER WINDOW ONLY** option when the synchronization mode is set to ASYNC or SUPERASYNC.

## TAKEOVER HADR

**Note:** The takeover operation with the **PEER WINDOW ONLY** option can behave incorrectly if the primary database clock and the standby database clock are not synchronized to within 5 seconds of each other. That is, the operation may succeed when it should fail, or fail when it should succeed. You should use a time synchronization service (for example, NTP) to keep the clocks synchronized to the same source.

### Usage notes

The following table shows the behavior of the **TAKEOVER HADR** command when issued on an active standby database 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                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Disconnected peer | No                                    | NOT ALLOWED (SQL1770N)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Disconnected peer | Yes, just <b>BY FORCE</b>             | ALLOWED - NO ASSURANCE OF DATA CONSISTENCY<br><b>Note:</b> A "no transaction loss" takeover is also possible using the <b>TAKEOVER BY FORCE</b> command without the <b>PEER WINDOW ONLY</b> option, that is, unconditional failover, as long as the necessary conditions hold. Such a failover can be executed even long after the expiration of the peer window that was in effect when the primary failed.                                                                                                                                                                                                                                                                                                                         |
| Disconnected peer | Yes, <b>BY FORCE PEER WINDOW ONLY</b> | ALLOWED - GREATER DEGREE OF DATA CONSISTENCY<br><br>There are situations in which data loss can still happen: <ul style="list-style-type: none"> <li>• If the primary database remains active past the time when the peer window expires, and if the primary database still has no connection to the standby database, the primary database will move out of disconnected peer state and resume processing transactions independently.</li> <li>• In NEARSYNC mode, if the standby database fails after acknowledging receipt of transaction logs from the primary database but before writing that transaction log information to disk, then that transaction log information, in the log receive buffer, might be lost.</li> </ul> |
| Local catchup     | No                                    | An error message is returned.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Local catchup     | Yes                                   | An error message is returned.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

| Standby state  | BY FORCE option used | Takeover behavior                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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"> <li>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 <b>hadr_db_role</b>.</li> <li>2. If the intended new primary is still in standby role, and takeover is still required, re-issue the <b>TAKEOVER HADR</b> command (see the next guideline regarding the <b>BY FORCE</b> option).</li> <li>3. It is possible to end up with both databases in standby role. In that case, the <b>TAKEOVER HADR</b> command with the <b>BY FORCE</b> option can be issued at whichever node should now become the primary. The <b>BY FORCE</b> option is required in this case because the two standbys cannot establish the usual HADR primary-standby connection.</li> </ol> |
| Peer           | Yes                  | <p>Old primary marked as invalid, preventing it from writing any more logs or committing any more transactions.</p> <p>The next log write attempt brings down the database. However, if sessions on the old primary only run read-only queries, the old primary might stay up indefinitely. Existing client connections stay open as long as they perform read-only operations and new client connections might be accepted.</p> <p>To avoid a situation of dual-primary, it is recommended that you stop transaction processing on old primary first before issuing a <b>TAKEOVER HADR</b> command with the <b>BY FORCE</b> option.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Remote catchup | No                   | <p>In SUPERASYNC mode, you can perform non-forced takeover operations in remote catchup state. An error is returned in other synchronization modes. Before starting a non-forced takeover operation, check the log gap between the primary and standby databases. Because the standby database must retrieve the logs in the gap and replay them, a large gap will cause a long elapsed time for the takeover operation. It is recommended that you perform non-forced takeover operations only when the log gap is small. To reduce the log gap between the primary and the standby databases, consider stopping or reducing the workload on the primary database.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

## TAKEOVER HADR

| Standby state          | BY FORCE option used | Takeover behavior                                                                                                                                 |
|------------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| Remote catchup         | Yes                  | In SUPERASYNC mode, you can perform forced takeover operations in remote catchup state. An error will be returned in other synchronization modes. |
| Remote catchup pending | No                   | An error message is returned.                                                                                                                     |
| Remote catchup pending | Yes                  | The standby database becomes the primary database.                                                                                                |

When issuing the **TAKEOVER HADR** command, the corresponding error codes might be generated: SQL1767N, SQL1769N, or SQL1770N 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.

When you issue the **TAKEOVER BY FORCE PEER WINDOW ONLY** command, and it succeeds (you called it while the primary was disconnected from the standby, but still within the peer window), then there will not be any transaction information about the primary database that was not already copied to the standby database.

If you have reads on standby enabled, any user application currently connected to the standby will be disconnected to allow the takeover to proceed. Depending on the number of readers that are active on the standby, the takeover operation can take slightly longer to complete than it would if there were no readers on the standby. New connections will not be allowed during the role switch. Any attempt to connect to the HADR standby during the role switch on takeover will receive an error (SQL1776N).

**Related information:**



---

## 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 before 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 database partition configuration.

## UNCATALOG DATABASE

Deletes a database entry from the database directory.

### Authorization

One of the following authorities:

- SYSADM
- SYSCTRL

### Required connection

None. Directory operations affect the local directory only.

### Command syntax

```

▶▶—UNCATALOG—{ DATABASE | DB }—database-alias—▶▶

```

### 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 an earlier 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 the database manager'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.

**Note:** When you add a database partition to the system, all existing databases in the instance are expanded to the new database partition. However, any uncataloged database is not recognized when adding a new database partition. The uncataloged database will not be present on the new database partition. An attempt to connect to the database on the new database partition returns the error message SQL1013N.

## UNCATALOG DCS DATABASE

Deletes an entry from the Database Connection Services (DCS) directory.

### Authorization

One of the following authorities:

- 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.

---

## UNCATALOG LDAP DATABASE

Deregisters the database from Lightweight Directory Access Protocol (LDAP).

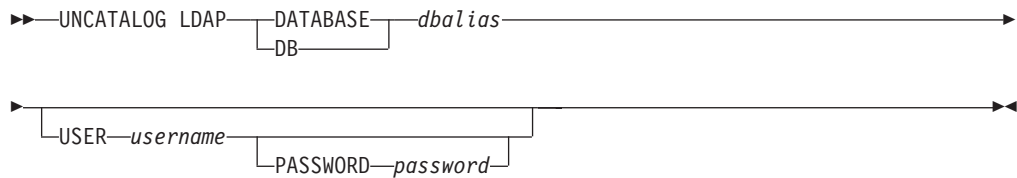
### Authorization

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.

---

## 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.

### UNCATALOG NODE

Deletes an entry from the node directory.

#### Authorization

One of the following authorities:

- 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. To see if directory caching is enabled, check the value for the **dir\_cache** directory cache support configuration parameter in the output from 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 the database manager'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.

## 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.

### Usage notes

On Microsoft Windows Vista or later versions, you must execute the **UNCATALOG SYSTEM ODBC DATA SOURCE** command from a DB2 command window running with full administrator privileges.





The following command will unquiesce the instance `instA` that had previously been quiesced.

```
db2 unquiesce instance instA
```

### Usage notes

- In a DB2 pureScale environment, after quiescing a database and restarting the instance, the database will remain quiesced across all members. An explicit **UNQUIESCE DATABASE** command is required to remove the quiesce state.

**Related information:**

---

## 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.

**Important:** The DB2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in DB2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see “ DB2 administration server (DAS) has been deprecated” at .

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**
- Diagnostic error capture level - **diaglevel**

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.

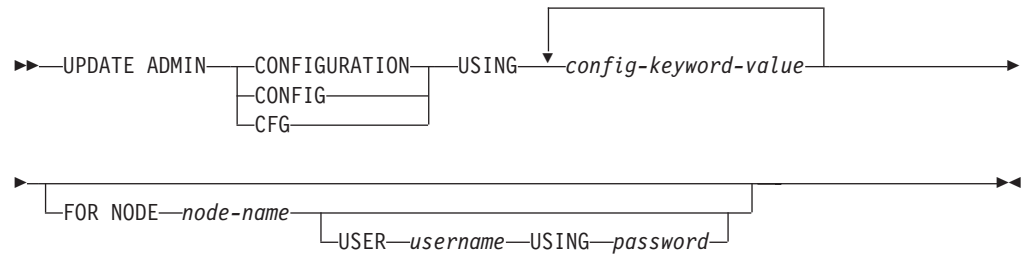
### Authorization

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.

## UPDATE ALERT CONFIGURATION

Updates the alert configuration settings for health indicators.

**Important:** This command or API has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in DB2 pureScale environments. For more information, see “Health monitor has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0055045.html>.

### Authorization

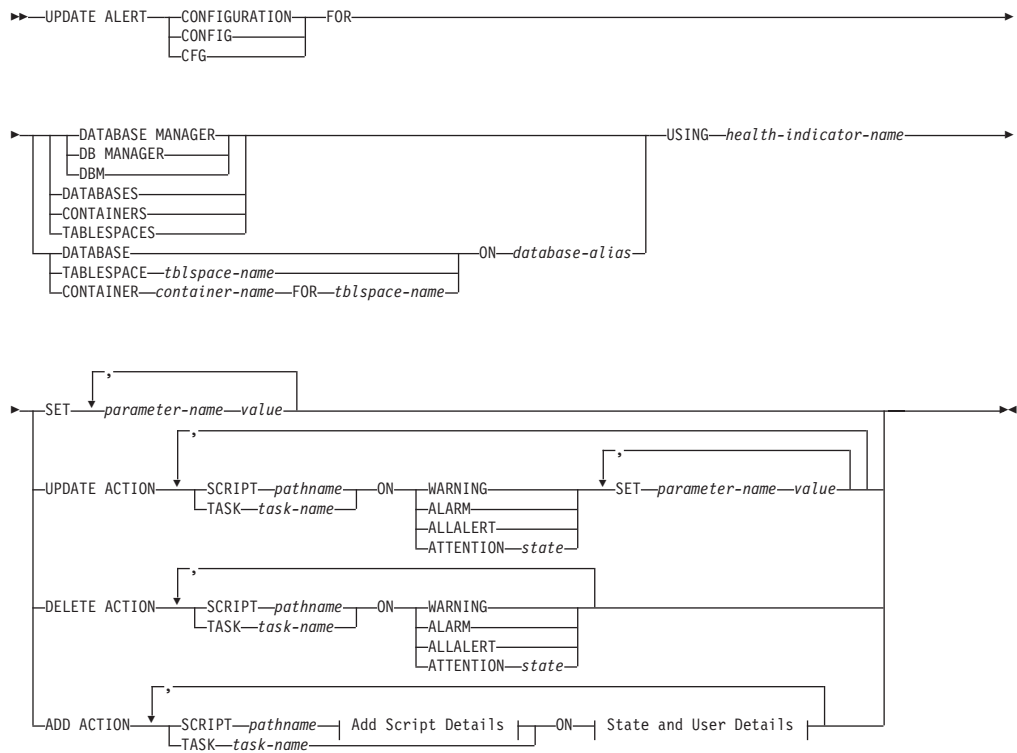
One of the following authorities:

- SYSADM
- SYSMANT
- SYSCTRL

### Required Connection

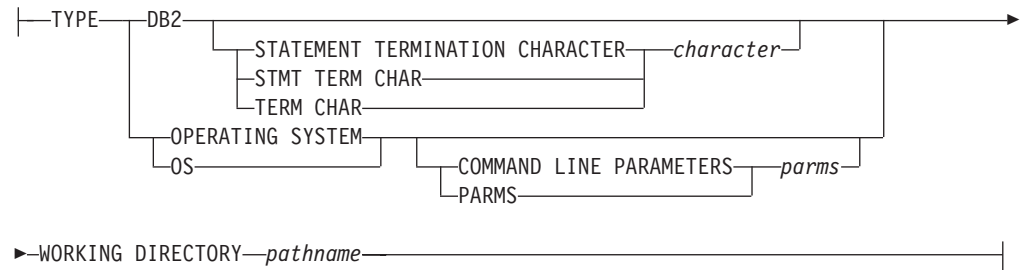
Instance. An explicit attachment is not required.

### Command Syntax

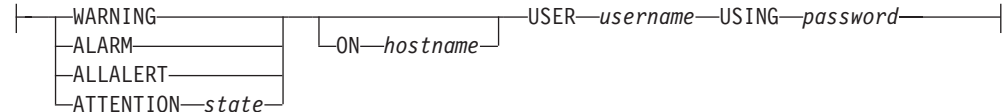


### Add Script Details:

## UPDATE ALERT CONFIGURATION



### State and User 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** *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** *tblspace-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** *container-name* **FOR** *tblspace-name* **ON** *database-alias*

Updates the alert settings for the table space container called *container-name*, for the table space specified using the **FOR** *tblspace-name* 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** *tblspace-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.

## UPDATE ALERT CONFIGURATION

### **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 values:

- 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 path name *pathname* will be updated according to the following clause:

#### **SET** *parameter-name value*

Updates the script attribute, *parameter-name*, to the specified value. *parameter-name* must be one of the following values:

- 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** *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 values:

- USERID
- PASSWORD
- SYSTEM

### **DELETE ACTION SCRIPT** *pathname* **ON** [WARNING | ALARM | ALLALERT | ATTENTION *state*]

Removes the action script with absolute path name *pathname* from the list of alert action scripts.

**DELETE ACTION TASK** *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 path name *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 ;

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 path name, *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 | ALLALERT | ATTENTION *state*]

Specifies the condition on which the action or task will run. For threshold-based health indicators (HIs), this is **WARNING** or **ALARM**. For state-based HIs, this will be a numeric state as documented for each state-based HI (for example, for the `ts.ts_op_status` health indicator, refer to the **tablespace\_state** monitor element for table space states). **ALLALERTS** handles any changes in the state for threshold-based HIs and state-based HIs (for example, the state changes from warning to normal).

**ATTENTION** *state*

Valid numeric values for some of the database health indicator states are given in the following section, as an example for the **ADD ACTION SCRIPT** CLP command option:

- 0 - Active; Normal (ACTIVE)
- 1 - Quiesce pending (QUIESCE\_PEND)
- 2 - Quiesced (QUIESCED)
- 3 - Rollforward (ROLLFWD)

Additional state-based health indicators are defined in the header files `sqlmon.h` and `sqlutil.h`.

## Usage notes

For the **ADD ACTION** option, the supplied *username* and *password* may be exposed in various places where SQL statement text is captured:

## UPDATE ALERT CONFIGURATION

- the network (username/password are passed over the wire unencrypted)
- **db2diag** log file
- trace files
- dump file
- snapshot monitor (dynamic SQL snapshot)
- system monitor snapshots
- a number of event monitors (statement, deadlock)
- explain tables
- **db2pd** output (package cache and lock timeout mechanisms, among others)
- DB2 audit records



---

## 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 authorities:

- 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.

## 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

```

▶—UPDATE ALTERNATE SERVER FOR LDAP—DATABASE—database-alias—▶
      |
      |—DB—
      |
▶—USING—NODE—node—▶
      |
      |—GWNODE—gwnode—
      |
      |—USER—username—
      |
      |—PASSWORD—password—▶
  
```

### 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 **db21dcfg**, 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 **db21dcfg**, the user name and password do not have to be specified here.

## 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 CLI and the applications using it. The file is divided into sections, each section corresponding to a database alias name.

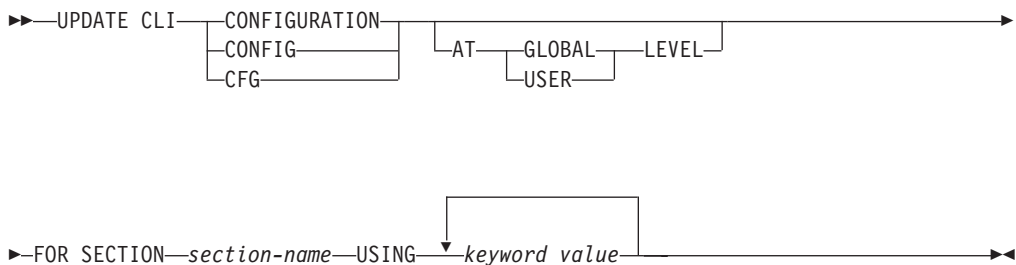
### Authorization

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. A user ID must have the "System" Active Directory authorization to update the CLI configuration parameter at the global level; otherwise, the update fails with error SQL3267N. 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), by using 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.

To delete CLI configuration parameters in an LDAP environment, your user ID must have DELETE authorization within the "System/IBM" Active Directory container. A user ID with WRITE and CREATE ALL CHILD OBJECTS authorizations can deactivate CLI configuration parameters, but not delete them.



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.

---

# 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

**UPDATE 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.



## 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.

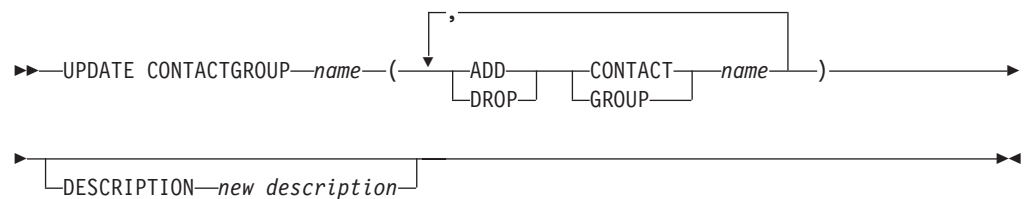
### Authorization

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.

## 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 updates all database partitions or members by default, except when the following optional clause is specified:

- **MEMBER** to update only one database member for a DB2 pureScale environment, or to update only one database partition in a partitioned database environment.

### Authorization

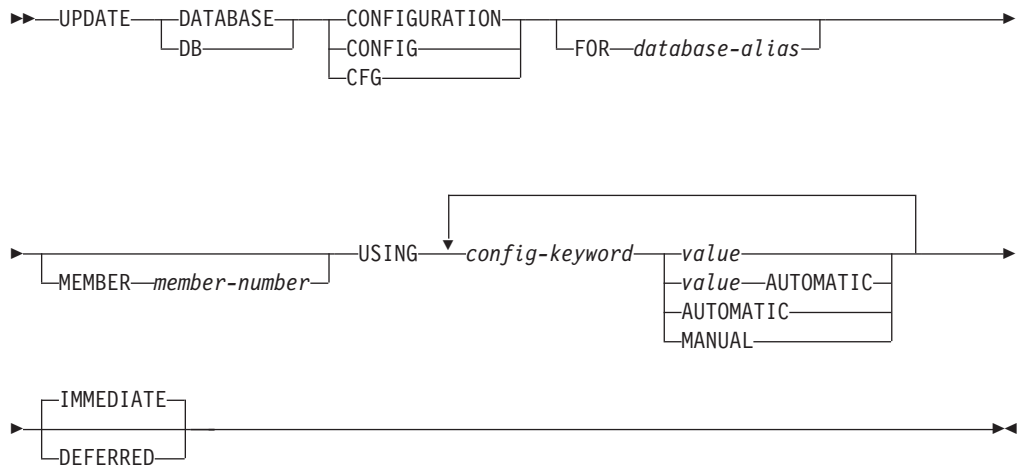
One of the following authorities:

- 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

#### AUTOMATIC

Some configuration parameters can be set to **AUTOMATIC**, allowing DB2 database systems to automatically adjust these parameters to reflect the current resource requirements. For a list of configuration parameters that support the **AUTOMATIC** keyword, refer to the configuration parameters summary. If a value is specified along with the **AUTOMATIC** keyword, it

might influence the automatic calculations. For specific details about this behavior, refer to the documentation for the configuration parameter.

### 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.
- 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.

### MEMBER *member-number*

The **MEMBER** clause specifies to which member the change should be applied. Omission of this clause results in the change being applied to all the members.

### IMMEDIATE

Make the changes immediately, while the database is running. **IMMEDIATE** is the default action, but it requires a database connection to be effective.

This is a default clause when operating in the CLPPlus interface as well. **IMMEDIATE** need not be called when using CLPPlus processor.

**MANUAL** Disables automatic tuning for the configuration parameter. The parameter is set to its current internal value and is no longer updated automatically.

### USING *config-keyword value*

*config-keyword* specifies the database configuration parameter to be updated. *value* specifies the value to be assigned to the parameter.

## Examples

### Update database configuration on a multi-partition instance

This example demonstrates how to update database configuration parameter **MAXAPPLS** from 10 to 50 for a database named SAMPLE.

A user has a multi-partition instance that has 4 partitions as defined in the `db2nodes.cfg`:

```
10 gilera 0
20 gilera 1
30 motobi 0
40 motobi 1
```

The user has created the SAMPLE database on the instance. The catalog partition for SAMPLE is on member 10. Let us assume the user is logged on to system motobi.

## UPDATE DATABASE CONFIGURATION

Since the default behavior for a multi-partition instance is to update the database configurations on all database partitions, the following command issued by users will result in the same value for **MAXAPPLS** across all database partitions:

```
db2 update db cfg for sample using maxappls 50
```

To update **MAXAPPLS** only on member 30, the following commands may be issued:

```
db2 update db cfg for sample member 30 using maxappls 50
```

### 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 logarchmeth1 logretain
```

To check that the **logarchmeth1** configuration parameter has changed, use:

```
db2 get db cfg for zellmart
```

For example, to change the logging mode to “archival logging” on all partitions (provided the registry variable **DB2\_UPDDBCFG\_SINGLE\_DBPARTITION** is set, by default, to NULL or FALSE) in a multiple-partitioned database environment containing a database called “zellmart”, use:

```
db2 update db cfg for zellmart using logarchmeth1 logretain
```

To check that the **logarchmeth1** configuration parameter has changed on all database partitions, use:

```
db2_all ";db2 get db cfg for zellmart"
```

Using the previous example, but to update the logging mode to only one specific partition (30), use:

```
db2 update db cfg for zellmart member 30 using logarchmeth1 logretain
```

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 **logarchmeth1** values:

```
db2_all ";db2 get db cfg for zellmart | grep -i logarchmeth1"
```

For more information about DB2 database 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.

Some changes to the database configuration file become effective only after they are loaded into memory. All applications must disconnect from the database before

## UPDATE DATABASE CONFIGURATION

this can occur. For more information about which parameters are configurable online 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.

### Compatibilities

For compatibility with previous versions:

- **DBPARTITIONNUM** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.

**Related information:**

## UPDATE DATABASE MANAGER CONFIGURATION

Modifies individual entries in the database manager configuration file.

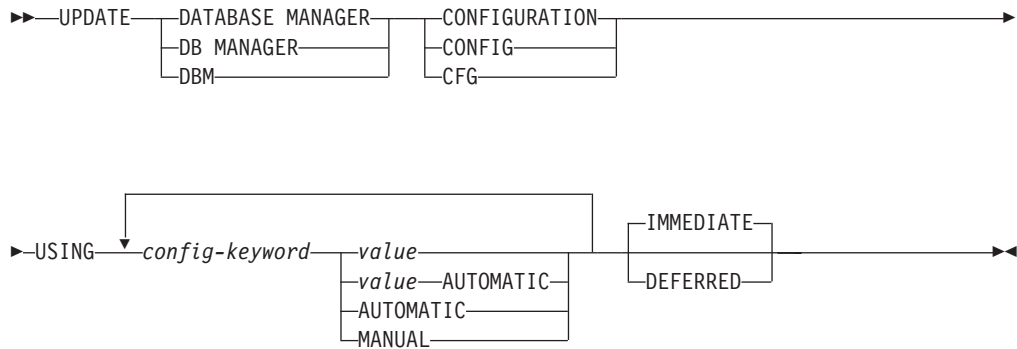
### Authorization

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 online, it is also necessary to first attach to the instance.

### Command syntax



### Command parameters

#### AUTOMATIC

Some configuration parameters can be set to **AUTOMATIC**, allowing DB2 to automatically adjust these parameters to reflect the current resource requirements. For a list of configuration parameters that support the **AUTOMATIC** keyword, refer to the configuration parameters summary. If a value is specified along with the **AUTOMATIC** keyword, it might influence the automatic calculations. For specific details about this behavior, refer to the documentation for the configuration parameter.

#### DEFERRED

Make the changes only in the configuration file, so that the changes take effect when the instance is restarted.

This is a default clause when operating in the CLPPlus interface. **DEFERRED** need not be called when using CLPPlus processor.

#### 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.

**MANUAL** Disables automatic tuning for the configuration parameter. The parameter is set to its current internal value and is no longer updated automatically.

**USING** *config-keyword value*

Specifies the database manager configuration parameter to be updated. For

## UPDATE DATABASE MANAGER CONFIGURATION

a list of configuration parameters, refer to the configuration parameters summary. *value* specifies the value to be assigned to the parameter.

### 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.

Some changes to the database manager configuration file become effective only after they are loaded into memory. For more information about which parameters are configurable online 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 online 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 information:**

---

## UPDATE HEALTH NOTIFICATION CONTACT LIST

Updates the contact list for notification about health alerts issued by an instance.

### Authorization

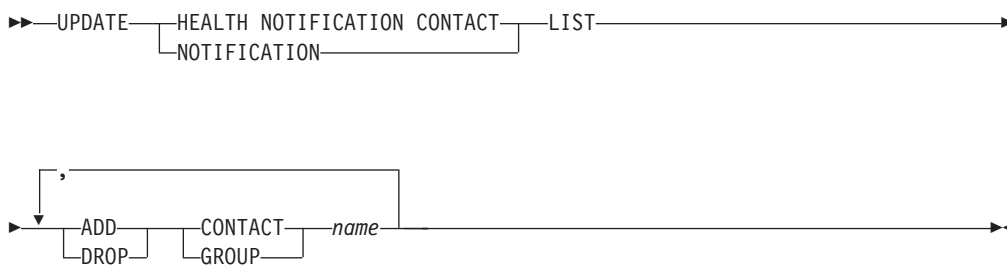
One of the following authorities:

- 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.



## UPDATE HISTORY

Updates the location, device type, comment, or status in a database history records entry.

### Authorization

One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- DBADM

### Required connection

Database

### Command syntax

```

▶ UPDATE HISTORY [FOR object-part | EID eid] WITH
[LOCATION new-location | DEVICE TYPE new-device-type |
COMMENT new-comment | STATUS new-status]

```

### 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:

|          |                        |
|----------|------------------------|
| <b>D</b> | Disk                   |
| <b>K</b> | Diskette               |
| <b>T</b> | Tape                   |
| <b>A</b> | Tivoli Storage Manager |
| <b>F</b> | Snapshot backup        |
| <b>U</b> | User exit              |
| <b>P</b> | Pipe                   |
| <b>N</b> | Null device            |
| <b>X</b> | XBSA                   |

## UPDATE HISTORY

**Q** SQL statement

**0** 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. The backup image is on the active log chain. Most entries are active.

**I** Inactive. Backup images that no longer correspond to the current log sequence, also called the current log chain, are flagged as 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.

**X** Do not delete. Recovery database history records file entries that are marked DB2HISTORY\_STATUS\_DO\_NOT\_DELETE will not be pruned by calls to the **PRUNE HISTORY** command, running the ADMIN\_CMD procedure with **PRUNE HISTORY**, calls to the db2Prune API, or automated recovery database history records pruning. You can use the DB2HISTORY\_STATUS\_DO\_NOT\_DELETE status to protect key recovery file entries from being pruned and the recovery objects associated with them from being deleted. Only log files, backup images, and load copy images can be marked as DB2HISTORY\_STATUS\_DO\_NOT\_DELETE.

### Example

To update the database history records 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 records 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.

## UPDATE LDAP NODE

Updates the protocol information associated with a node entry that represents the DB2 server in Lightweight Directory Access Protocol (LDAP).

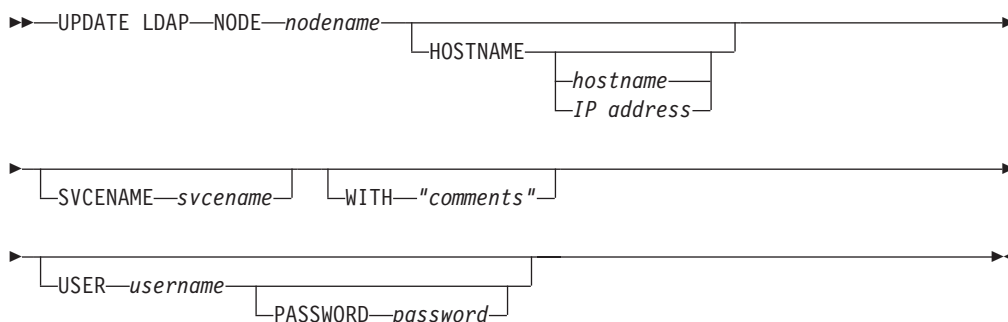
### Authorization

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* | *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.

## UPDATE LDAP NODE

**PASSWORD** *password*  
Account password.

# 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.

## Scope

This command is invoked on the currently attached member and only returns information for that member by default. The currently attached member defaults to the host on which the command is run. If there are multiple members per host, the currently attached member is the first member listed in the `db2nodes.cfg` file on that host.

To run the command on a member that is not the current attach member, specify the **AT MEMBER** option to run this command on a specific member, or specify the **GLOBAL** option to run this command on all members to receive an aggregated result.

To change the current attach member from which to run any subsequent commands, use the **ATTACH\_MEMBER** parameter of the **SET CLIENT** command. You must **DETACH** and then again **ATTACH** your application for this client setting to take effect.

## Authorization

One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON

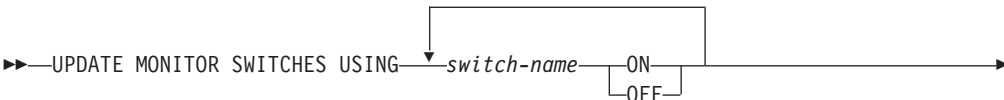
## Required connection

Instance or database:

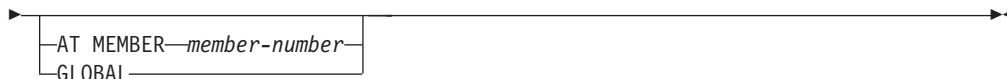
- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

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



## UPDATE MONITOR SWITCHES



### 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 MEMBER** *member-number*

Specifies the member for which the monitor switches are updated.

**GLOBAL**

Updates the monitor switches on all members.

### 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 previous versions:

- **DBPARTITIONNUM** or **NODE** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.

## UPDATE XMLSCHEMA

Updates one XML schema with another in the XML schema repository (XSR).

### Authorization

One of the following authorities:

- DBADM
- SELECT privilege on the catalog views SYSCAT.XSROBJECTS and SYSCAT.XSROBJECTCOMPONENTS and one of the following sets of privileges:
  - ALTERIN privilege on the XML schema to be updated and DROPIN privilege on the new XML schema, if the **DROP NEW SCHEMA** option is specified.
  - OWNER of the XML schema specified by `xmlschema1`.

### Required connection

Database

### Command syntax

```
►► UPDATE XMLSCHEMA xmlschema1 WITH xmlschema2 DROP NEW SCHEMA ◄◄
```

### Command parameters

**UPDATE XMLSCHEMA** *xmlschema1*

Specifies the SQL identifier for the original XML schema to be updated.

**WITH** *xmlschema2*

Specifies the SQL identifier for the new XML schema that will be used to update the original XML schema.

**DROP NEW SCHEMA**

Indicates that the new XML schema should be dropped after it is used to update the original XML schema.

### Example

```
UPDATE XMLSCHEMA JOHNDOE.OLDPROD
WITH JOHNDOE.NEWPROD
DROP NEW SCHEMA
```

The contents of the XML schema JOHNDOE.OLDPROD is updated with the contents of JOHNDOE.NEWPROD, and the XML schema JOHNDOE.NEWPROD is dropped.

### Usage notes

- The original and new XML schema must be compatible. For details about the compatibility requirements, see “Compatibility requirements for evolving an XML schema”.
- Before an XML schema can be updated, both the original and the new schema must be registered in the XML schema repository (XSR).

### UPGRADE DATABASE

Converts a DB2 database of the previous version to the formats corresponding to the release run by the instance.

The **db2ckupgrade** command must be issued before upgrading the instance to verify that your databases are ready for upgrade. The **db2iupgrade** command implicitly calls the **db2ckupgrade**. Backup all databases before upgrading, and before the installation of the current version of DB2 database product on Windows operating systems.

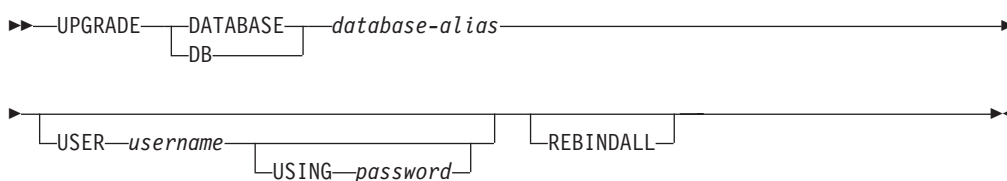
#### Authorization

SYSADM

#### Required connection

This command establishes a database connection.

#### Command syntax



#### Command parameters

**DATABASE** *database-alias*

Specifies the alias of the database to be upgraded to the currently installed version of the database manager.

**USER** *username*

Identifies the user name under which the database is to be upgraded.

**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.

**-REBINDALL**

Specifies that a REBIND of all packages is performed during upgrade. Performing the REBINDs automatically during the upgrade ensures that the step is not missed and will help ensure that other applications are not started before the REBINDs are complete.

#### Examples

The following example upgrades the database cataloged under the database alias sales:

```
db2 UPGRADE DATABASE sales
```

#### Usage notes

This command will only upgrade a database to a newer version, and cannot be used to convert an upgraded database to its previous version.



The database must be cataloged before upgrade.

If an error occurs during upgrade, it might be necessary to issue the **TERMINATE** command before attempting the suggested user response. For example, if a log full error occurs during upgrade (SQL1704: Database upgrade failed. Reason code "3"), it will be necessary to issue the **TERMINATE** command before increasing the values of the database configuration parameters **logprimary** and **logfilesiz**. The CLP must refresh its database directory cache if the upgrade failure occurs after the database has already been relocated (which is likely to be the case when a "log full" error returns).

## UPGRADE DATABASE

---

## Chapter 6. CLPPlus commands

The CLPPlus feature includes many commands which provide extensive user control, customization, and personalization.

**Note:** Unless otherwise specified, CLPPlus command names and parameters are not case-sensitive; you can specify uppercase or lowercase letters.

The `. CLPPlus` command is similar to a No Operation Performed (NOOP or NOP) machine language command. It is ignored when entered on its own with no other CLPPlus command.

The `. CLPPlus` command can also be used to skip the currently entered command and move to the next `SQL>` prompt. This can be done by entering the command on its own on a new line in the current block of code. This can help you cancel a command when you enter an incorrect entry. A cancelled command is available in the history and can be accessed and edited.

### Invocation

You must run this command from the CLPPlus interface or from within a CLPPlus script file.

### Authorization

None

### Required connection

None

### Command syntax

▶▶ . —————▶▶

### Example

In the following example, the `.` command is used to cancel the current command.

```
SQL> begin
      2 dbms_output.putline('wrong put_line');
      3 .
SQL>
```

As of V9.7 Fix Pack 3, the ! CLPPlus command is a synonym to the **HOST** CLPPlus command. It will run an operating system command.

## Invocation

You must run this command in the CLPPlus interface.

## Authorization

None

## Required connection

None

## Command syntax

►—!*os\_command*—◄

## Command parameters

*os\_command*

Specifies an operating system command.

## Example

In the following example, the ! command is issued with the **dir** operating system command.

```
SQL> ! dir
```

```
Volume in drive C has no label.  
Volume Serial Number is 6806-ABBD
```

```
Directory of C:\USER_Backup\Office\Project\FP3\FP3\src
```

```
06/05/2010 22:18 <DIR>  .  
06/05/2010 22:18 <DIR>  ..  
06/05/2010 22:35          405.classpath  
06/05/2010 17:20 <DIR>  com  
                1 File(s)          798 bytes  
                3 Dir(s) 33,397,190,656 bytes free
```





The @@ CLPPlus command is an alias for the **START** CLPPlus command. It can only be used from within a CLPPlus script file to call and run another CLPPlus script file.

## Invocation

You must run this command from within a CLPPlus script.

## Authorization

None

## Required connection

None

## Command syntax

▶▶ @@ path script-file ▶▶

## Command parameters

### *path*

Specifies the path, either absolute or relative, to the script file that contains SQL statements and commands to run. If no path is specified, the current directory is used.

### *script-file*

Specifies the script file name that contains SQL statements and commands to run.

## Example

A script called dept\_details.sql calls another script called employee\_count.sql. The contents of dept\_details.sql follows:

```
ACCEPT dept_id PROMPT "Enter Department ID code : "  
@@ employee_count &dept_id
```



## ACCEPT

The **ACCEPT** CLPPlus command creates a variable with a specified name. Values can be assigned to this variable interactively in the CLPPlus interface or via a parameter read as part of a script being run. The **ACCEPT** command is useful for storing values that you commonly use in SQL statements or in the SQL buffer.

Output of the command is by default displayed to the standard output of the CLPPlus interface.

### Invocation

You must run this command from the CLPPlus interface or from within a CLPPlus script file.

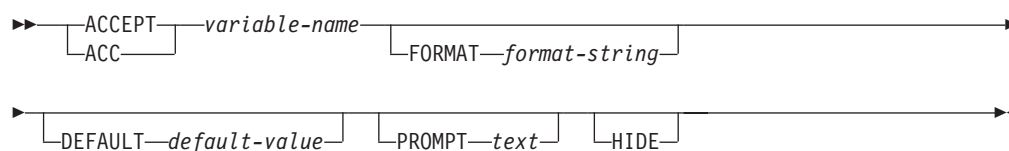
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### *variable-name*

Defines the variable name. You cannot use special symbols and characters such as the forward slash (/) or at sign (@).

When you issue the **ACCEPT** command, you are prompted for the value of *variable-name*.

#### **FORMAT** *format-string*

Defines the format assigned to the variable. The value you attempt to assign to the variable must follow the format outlined.

#### **DEFAULT** *default-value*

The default value defined with this option is assigned to the variable when a user hits the ENTER key and does not provide any value when prompted.

#### **PROMPT** *text*

The value defined with this option is displayed in the prompt when the **ACCEPT** command is entered.

#### **HIDE**

When **HIDE** is specified, the value entered by the user is not echoed on the console.



## APPEND

The **APPEND** CLPPlus command adds text to the end of the current line in the SQL buffer. You can use this command to build commands and statements in the SQL buffer.

### Invocation

This is a line editor command and can be used to build commands in the SQL buffer.

### Authorization

None

### Required connection

None

### Command syntax

### Command parameters

*text-string*

Specifies a string of characters to append. The string can include spaces and special characters. The case of the string is preserved.

### Examples

In the following example, the **APPEND** command appends the string this text is appended. to the end of the current line in the SQL buffer:

```
APPEND this text is appended.
```

The following example shows how you can use the **APPEND** command to build a **SELECT** statement in the SQL buffer. Two spaces are placed between the **APPEND** command and the **WHERE** clause to separate **DEPT** and **WHERE** by one space in the SQL buffer.

```
SQL> APPEND SELECT * FROM DEPT
SQL> LIST
1* SELECT * FROM DEPT
SQL> APPEND  WHERE DEPTNO = 10
SQL> LIST
1* SELECT * FROM DEPT WHERE DEPTNO = 10
```

The **LIST** command displays the contents of the SQL buffer as the SQL statement is being built.

## BREAK

The **BREAK** CLPPlus command inserts a page break or blank lines at the specified point in a result set.

### Invocation

You must run this command from the CLPPlus interface.

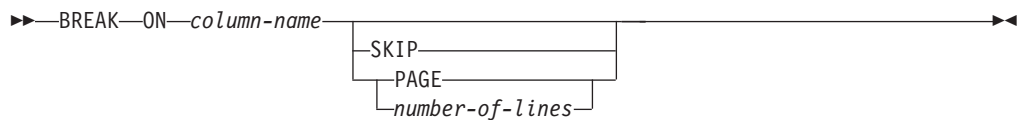
### Authorization

None

### Required connection

You must be connected to a database.

### Command syntax



### Command parameters

*column-name*

Specifies the column used to determine a break.

**SKIP PAGE** | *number-of-lines*

Where *number-of-lines* is an integer.

When **SKIP PAGE** is appended to the command the output breaks and continues on the next page. When **SKIP** *number-of-lines* is appended to the command, the output breaks and blanks lines equal to the *number-of-lines* specified are inserted in the result set.

### Example

In the following example, when the SELECT statement is invoked and the value of WORKDEPT changes from one row to another, the **BREAK** command is invoked and the action specified is performed. In this case, since **SKIP PAGE** was specified, the next row will be printed on the next page skipping the remainder of the current page.

```

SQL> BREAK ON WORKDEPT SKIP PAGE;
SQL> SELECT * FROM EMPLOYEE ORDER BY WORKDEPT;
  
```

In the following example, in addition to the behavior of the preceding example, every time the value in the JOB column changes, 2 blank lines are printed on the display.

```

SQL> BREAK ON WORKDEPT SKIP PAGE;
SQL> BREAK ON JOB SKIP 2;
SQL> SELECT * FROM EMPLOYEE ORDER BY WORKDEPT, JOB;
  
```

## BTITLE

The **BTITLE** CLPPlus command inserts text at the bottom of each page displayed.

### Invocation

You must run this command from the CLPPlus interface.

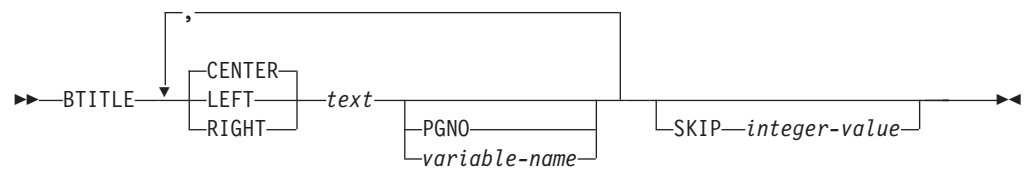
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

*text*

Specifies the text to be displayed.

**CENTER**

Specifies the display will center justify the text on each page. If neither **CENTER**, **LEFT**, or **RIGHT** is specified, center justification is the default behavior.

**LEFT**

Specifies the display will left justify the text on each page.

**RIGHT**

Specifies the display will right justify the text on each page.

**PGNO**

Specifies the current page number.

*variable-name*

Specifies a user defined variable that will follow the *text* field.

**SKIP** *integer-value*

The *integer-value* value specifies the number of blank lines displayed after the bottom title.

### Example

In the following example, the DEPT: (with the variable contents), CONFIDENTIAL, and Page No: (with the current page number) is displayed across the bottom of every page. Three blank lines follows the bottom title.

```

SQL> BREAK ON workdept SKIP PAGE;
SQL> COLUMN workdept OLD_VALUE old_dept;
SQL> BTITLE LEFT 'DEPT: ' old_dept, CENTER 'CONFIDENTIAL, RIGHT 'Page No: ' PGNO SKIP 3;

```

## BTITLE

In the following example, the Page No: title (with the current page number) is displayed across the bottom of every page with right justification. Two blank lines follow the bottom title.

```
SQL> BTITLE RIGHT 'Page No: ' PGNO SKIP 2;
```

## CHANGE

The **CHANGE** CLPPlus command modifies specified content in the SQL buffer. If you set the buffer reader to a specific line in the buffer, the command modifies only the specified content in that line.

As of V9.7 Fix Pack 3, the **CHANGE** or **C** token is optional for the complete command when specifying which line in the buffer you are changing. You can issue the command by specifying only the line number in the buffer you want to change along with the new text.

### Invocation

This is a line-editor command used to modify fields in the SQL buffer.

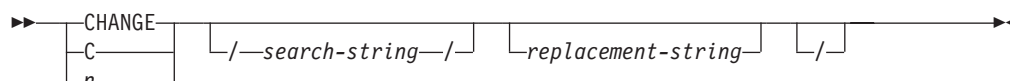
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

*n* Specifies the line number in the buffer that will be changed. The *n* command only takes the *replacement-string* variable.

#### *search-string*

Defines the text in the SQL buffer to be replaced or deleted. If the buffer contains more than one line of text, specify the line to be modified by entering the line number at the prompt before you run the **CHANGE** command.

If the text to search for contains an asterisk (\*), enclose the asterisk in single quotation marks.

#### *replacement-string*

Specifies the replacement text or specifies that text is to be removed. If you specify a value for *replacement-string*, the first occurrence of the value of *search-string* is replaced with the value of *replacement-string*. If you do not specify a value for *replacement-string*, the first occurrence of the value of *search-string* is removed.

If the replacement text contains an asterisk (\*), enclose the asterisk in single quotation marks. As of V9.7 Fix Pack 3, you do not need to enclose an asterisk (\*) in single quotation marks.

### Examples

In the following example, the **LIST** command displays the contents of the buffer. At the SQL prompt, 3 is entered to move the buffer reader to the start of the third line

## CHANGE

in the buffer. The third line becomes the new current line, as indicated by an asterisk. The **CHANGE** command then replaces the occurrence of the string 20 with the string 30. The LIST command then displays the modified text within the buffer.

```
SQL> LIST
 1 SELECT EMPNO, ENAME, JOB, SAL, COMM
 2 FROM EMP
 3 WHERE DEPTNO = 20
4* ORDER by EMPNO
SQL> 3
3* WHERE deptno = 20
SQL> CHANGE /20/30/
3* WHERE DEPTNO = 30
SQL> LIST
 1 SELECT EMPNO, ENAME, JOB, SAL, COMM
 2 FROM EMP
3* WHERE DEPTNO = 30
 4 ORDER by EMPNO
```

In the following example, the buffer contains the following single statement:

```
SQL> SELECT EMPNO FROM EMPLOYEE
```

To change the statement so that EMPNO is replaced with \*, 1 is entered to move the buffer reader to the start of the first line in the buffer. The following **CHANGE** command is issued:

```
SQL> CHANGE /empno/'*' /
```

The output of the command is as follows:

```
1* SELECT * FROM EMPLOYEE
```

The command output displays the line number followed by the new content for that line.

As of V9.7 Fix Pack 3, the following is now possible. You can use the enhanced **CHANGE** command to specify the line number in the buffer you want to change, and what value you want to change it to.

```
SQL> SELECT *
 2 FROM
 3 EMPLOKEE ;
ERROR near line 1:
SQL0204N "SCHEMA.EMPLOKEE" is an undefined name.
```

```
SQL> LIST
 1 SELECT *
 2 FROM
3* EMPLOKEE
```

```
SQL> 3 EMPLOYEE
3* EMPLOYEE
```

```
SQL> LIST
 1 SELECT *
 2 FROM
3* EMPLOYEE
```

```
SQL> /
```



## CLEAR

The **CLEAR** CLPPlus command removes the contents of the SQL buffer, deletes all column definitions set by the **COLUMN** command, or clears the screen.

### Invocation

You must run this command from the CLPPlus interface.

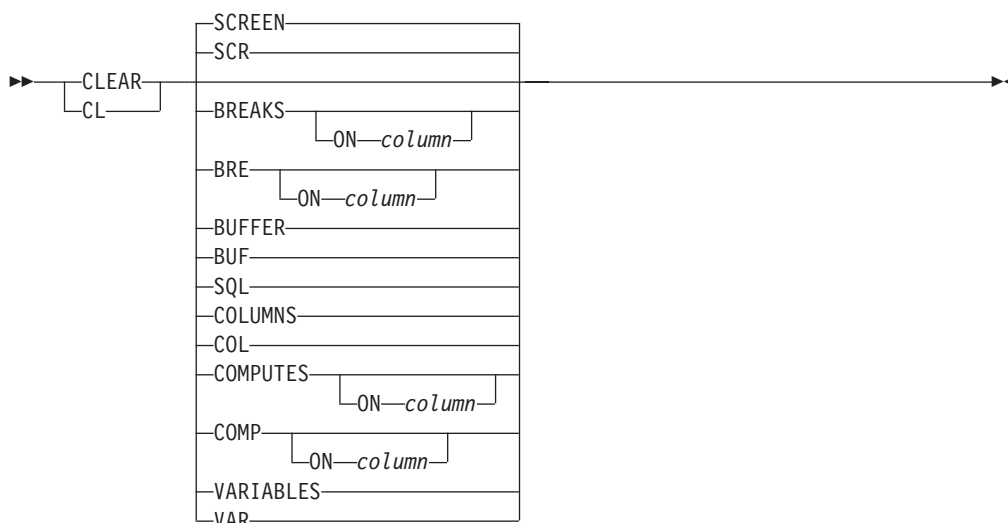
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### SCREEN | SCR

Removes all SQL commands, data currently displayed, and CLPPlus messages from the screen. When the **CLEAR** command is entered with no options, the default behavior clears the screen.

#### BREAKS | BRE ON *column*

Clears all breaks when no column is specified. When a column is specified, the break associated with that column is cleared, all other breaks are left intact.

#### BUFFER | BUF and SQL

Deletes all text in the SQL buffer. You must specify both the **BUFFER** parameter (or the **BUF** parameter) and the **SQL** parameter.

#### COLUMNS | COL

Removes column definitions in the SQL buffer.

#### COMPUTES | COMP ON *column*

Clears all compute definitions when no column is specified. When a column is

## **CLEAR**

specified, the compute definition associated with that column is cleared, all other compute definitions are left intact.

### **VARIABLES | VAR**

Clears all defined bind variables.

## COLUMN

The **COLUMN** CLPPlus command specifies character and numeric output formats for columns in a table. Formats set by the **COLUMN** command remain in effect only for the duration of the current session. You can change or clear format settings for the same column more than once in the current session.

When you issue **COLUMN** for a specified column in a database, format settings are by default displayed using the standard output of the CLPPlus interface.

### Invocation

This command must be executed from the CLPPlus interface.

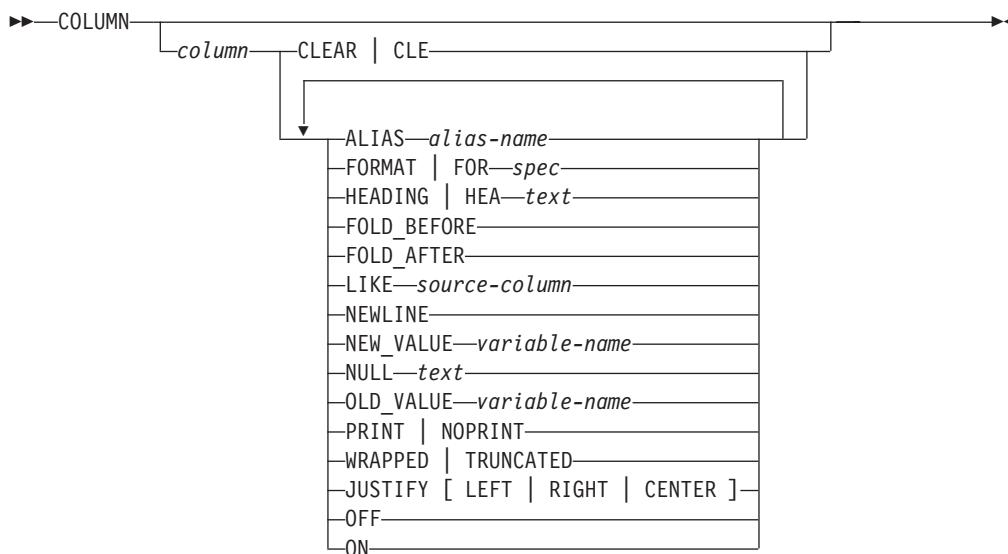
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### *column*

Specifies the name of a table column to which formats are applied. If you do not specify any parameters after *column*, default format settings apply. If you do not specify any parameters after *column*, and you have set parameters for the same column previously in this session, the last set of parameters assigned to the column are again used.

#### **ALIAS** *alias-name*

Specifies and alias name for *column*. The maximum length of *alias-name* is 255 characters. The alias can be used in other commands such as: **COMMAND**, **COMPUTE**, and **COLUMN**.

## COLUMN

### **CLEAR** | **CLE**

Changes all formatting settings for the specified column to their default values. If you specify **CLEAR**, you cannot specify any parameters other than *column*.

### **FORMAT** | **FOR** *spec*

Specifies the formats to apply to the specified column. There are two types of columns: character columns and numeric columns.

For a character column, the value of *spec* is *An*, where *n* specifies the number of characters that can be used to display the column. The data wraps if it is wider than the specified width.

For numeric columns, the value of *spec* can be one or more of the following characters:

- \$ Displays a leading dollar sign.
- ,
- .
- 0 Displays a zero at the indicated position.
- 9 Displays a significant digit at the indicated position.

If loss of significant digits occurs due to overflow of the format settings, the # character will be displayed.

### **HEADING** | **HEA** *text*

Specifies a heading for the specified column.

### **FOLD\_BEFORE**

Before printing the values for the specified column, new line feed and carriage return are provided for each row.

### **FOLD\_AFTER**

After printing the values for the specified column, new line feed and carriage return are provided for each row.

### **LIKE** *source-column*

The format and display attributes of *source-column* are applied to *column*.

### **NEWLINE**

A synonym of **FOLD\_AFTER**. After printing the values for the specified column, new line feed and carriage return are provided for each row.

### **NEW\_VALUE** *variable\_name*

Defines a variable that can hold the new value for a break column defined using the **BREAK** command. The variable can be used with page top title **TTITLE** command. The break column must be defined with the **SKIP PAGE** action.

As of V9.7 Fix Pack 3, **NEW\_VALUE** can also be used in all places within the current session. Similar to a substitution variable. Whenever you define a **NEW\_VALUE** variable for a column, CLPPlus creates a substitution variable with the specified variable name. This variable is updated with the column value on each column break.

### **NULL** *text*

When the value for the specified column is NULL, the value specified for *text* is printed. The maximum length of *text* is 255 characters.

### **OLD\_VALUE** *variable\_name*

Defines a variable that can hold the old value for a break column defined

using the **BREAK** command. The variable can be used with page bottom title **BTITLE** command. The break column must be defined with the **SKIP PAGE** action.

As of V9.7 Fix Pack 3, **OLD\_VALUE** can also be used in all places within the current session. Similar to a substitution variable. Whenever you define a **OLD\_VALUE** variable for a column, CLPPlus creates a substitution variable with the specified variable name. This variable is updated with the column value on each column break.

**PRINT | NOPRINT**

Specifies whether console printing of a specified column is enabled or disabled.

**WRAPPED | TRUNCATED**

Specifies if column data is wrapped or truncated in the CLPPlus output if it exceeds the specified format.

**JUSTIFY [LEFT | RIGHT | CENTER]**

Specifies column justification to be either LEFT, RIGHT, or CENTER.

**OFF**

Changes the formatting options to the default values. The values that you previously specified for the column in the session are saved and still available for use later in the session.

**ON**

Changes the formatting options to the values applied to the specified column the last time that you ran **COLUMN**.

**Examples**

In the following example, the **SET PAGESIZE** command sets the maximum page length to 9999, and the **COLUMN** command changes the display width of the JOB column to five characters. The SELECT statement then prints specified columns in the table.

```
SQL> SET PAGESIZE 9999
SQL> COLUMN JOB FORMAT A5
SQL> COLUMN JOB
COLUMN      JOB      ON
FORMAT      A5
WRAPPED
SQL> SELECT EMPNO, ENAME, JOB FROM EMP;
```

| EMPNO | ENAME   | JOB           |
|-------|---------|---------------|
| 7369  | SMITH   | CLERK         |
| 7499  | ALLEN   | SALES<br>MAN  |
| 7521  | WARD    | SALES<br>MAN  |
| 7566  | JONES   | MANAG<br>ER   |
| 7654  | MARTING | SALES<br>MAN  |
| 7698  | BLAKE   | MANAG<br>ER   |
| 7782  | CLARK   | MANAG<br>ER   |
| 7788  | SCOTT   | ANALY<br>ST   |
| 7839  | KING    | PRESI<br>DENT |
| 7844  | TURNER  | SALES         |

## COLUMN

```
7876 ADAMS      MAN
7900 JAMES      CLERK
7902 FORD       CLERK
              ANALY
              ST
7934 MILLER     CLERK
```

14 rows received.

In the following example, the **COLUMN** command applies a numeric format to the SAL column:

```
SQL> COLUMN SAL FORMAT $99,999.00
SQL> COLUMN
COLUMN  JOB  ON
FORMAT  A5
WRAPPED

COLUMN  SAL  ON
FORMAT  $99,999.00
WRAPPED
SQL> SELECT EMPNO, ENAME, JOB, SAL FROM EMP;
```

| EMPNO | ENAME  | JOB   | SAL        |
|-------|--------|-------|------------|
| 7369  | SMITH  | CLERK | \$800.00   |
| 7499  | ALLEN  | SALES | \$1,600.00 |
|       |        | MAN   |            |
| 7521  | WARD   | SALES | \$1,250.00 |
|       |        | MAN   |            |
| 7566  | JONES  | MANAG | \$2,975.00 |
|       |        | ER    |            |
| 7654  | MARTIN | SALES | \$1,250.00 |
|       |        | MAN   |            |
| 7698  | BLAKE  | MANAG | \$2,850.00 |
|       |        | ER    |            |
| 7782  | CLARK  | MANAG | \$2,450.00 |
|       |        | ER    |            |
| 7788  | SCOTT  | ANALY | \$3,000.00 |
|       |        | ST    |            |
| 7839  | KING   | PRESI | \$5,000.00 |
|       |        | DENT  |            |
| 7844  | TURNER | SALES | \$1,500.00 |
|       |        | MAN   |            |
| 7876  | ADAMS  | CLERK | \$1,100.00 |
| 7900  | JAMES  | CLERK | \$950.00   |
| 7902  | FORD   | ANALY | \$3,000.00 |
|       |        | ST    |            |
| 7934  | MILLER | CLERK | \$1,300.00 |

14 rows retrieved.

In the following example, the improved **NEW\_VALUE** parameter behavior is shown. The new **OLD\_VALUE** behavior is identical. This is only available as of V9.7 Fix Pack 3:

```

SQL> break on empno skip 1
SQL> column empno new_value highest_sal
SQL> select empno from employee order by salary;

EMPNO
-----
200340
*****

000290
*****

200330
*****

000310
*****

...
...

000070
*****

000030
*****

000010
*****

SQL>DEFINE
DEFINE HIGHEST_SAL = 000010

SQL> select EMPNO, FIRSTNME, MIDINIT, LASTNAME from employee where empno=&highest_sal;
EMPNO  FIRSTNME      MIDINIT LASTNAME
-----
000010 CHRISTINE     I      HAAS

```

## COMPUTE

The **COMPUTE** CLPPlus command executes a specified function on the aggregate values of a defined column. The command works in conjunction with the **BREAK** CLPPlus command.

### Invocation

You must run this command from the CLPPlus interface.

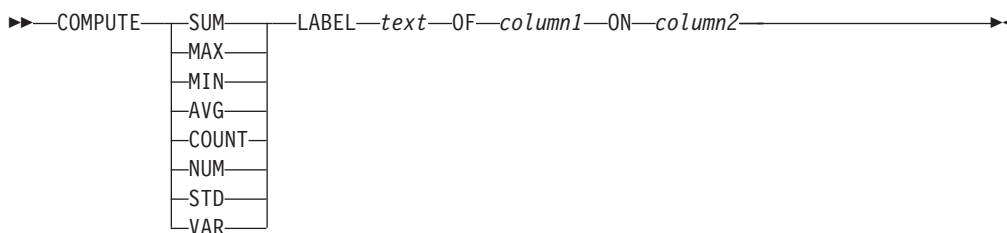
### Authorization

None

### Required connection

You must be connected to a database.

### Command syntax



### Command parameters

#### SUM

The SUM function adds the aggregate values on the column specified.

#### MIN

The MIN function returns the smallest of the aggregate values on the column specified.

#### MAX

The MAX function returns the largest of the aggregate values on the column specified.

#### AVG

The AVG function returns the average of the aggregate values on the column specified.

#### COUNT

The **COUNT** function counts the number of non-null values on the column specified.

#### NUM

The NUM function returns the number of aggregate rows processed on the column specified.

#### STD

The STD function returns the standard deviation of the aggregate values on the column specified.



**VAR**

The VAR function returns the variance of the aggregate values on the column specified.

**LABEL** *text*

Defines the text label that precedes the output of the function specified.

*column1*

Specifies the column on which the function is executed.

*column2*

Specifies the column on which the **BREAK** command is executed against.

**Example**

The following example highlights the usage of the **COMPUTE** command in conjunction with the **BREAK** command.

```
SQL> BREAK ON WORKDEPT SKIP 2;
SQL> COMPUTE AVG LABEL "Average" OF SALARY ON WORKDEPT;
SQL> COMPUTE MAX LABEL "Maximum" OF SALARY ON WORKDEPT;
SQL> SELECT WORKDEPT, EMPNO, SALARY FROM EMPLOYEE ORDER BY WORKDEPT;
```

Here is the output of the commands in the example.

| WORKDEPT | EMPNO | SALARY   |
|----------|-------|----------|
| -----    | ----- | -----    |
| A01      | 00100 | 75000.00 |
| A01      | 00101 | 80000.00 |
| A01      | 00102 | 70000.00 |
| *****    |       | -----    |
| Average  |       | 75000.00 |
| Maximum  |       | 80000.00 |
|          |       |          |
| A02      | 00103 | 80000.00 |
| A02      | 00104 | 90000.00 |
| A02      | 00105 | 85000.00 |
| *****    |       | -----    |
| Average  |       | 85000.00 |
| Maximum  |       | 90000.00 |



parameters are obtained interactively. As of V9.7 Fix Pack 5, if you configure an LDAP directory server in the specified `db2dsdriver.cfg` file, the following steps apply:

1. The database connection information is read from the `db2dsdriver.cfg` file from the `dsn` with alias name `dsn_alias`.
2. If `dsn_alias` is not found in the `db2dsdriver.cfg` file, the configured LDAP Directory server is searched for an entry with the name `dsn_alias`.
3. If `dsn_alias` is not found on the LDAP Directory server, `dsn_alias` is used as a database name and all other connection parameters are obtained interactively.

## Examples

In the following example, the database connection is changed to database DB2 on the local host at port 5445 with user name smith:

```
SQL> CONNECT smith/mypassword@localhost:5445/db2
Connected to CLPPlus 1.1.0.10 (localhost:5445/db2) AS smith
```

In the same session, the connection is changed to username CLPPlus. For this connection, localhost, 5444, and DB2 are maintained as the host, port, and database values.

```
SQL> CONNECT CLPPlus/password
Connected to CLPPlus 1.1.0.10 (localhost:5444/db2) AS CLPPlus
```

The following example attempts to connect to a database by first locating a `db2dsdriver.cfg` file. If one is found, the `default_dsn` is read for the host, port, and database values. The current log on ID is used in the connection attempt. If no `db2dsdriver.cfg` file is found, all required parameters are requested interactively.

```
SQL> CONNECT /
```

The following example attempts to connect to a database by fetching the parameters from the `data_dsn` alias in the `db2dsdriver.cfg` file. The `db2admin` user ID is used in the connection. Any parameters that cannot be read are requested interactively.

```
SQL> CONNECT db2admin@data_dsn
```

## COPY

The **COPY** CLPPlus command copies data from a source database and table to a target database and table.

### Invocation

You must run this command from the CLPPlus interface or from within a CLPPlus script file.

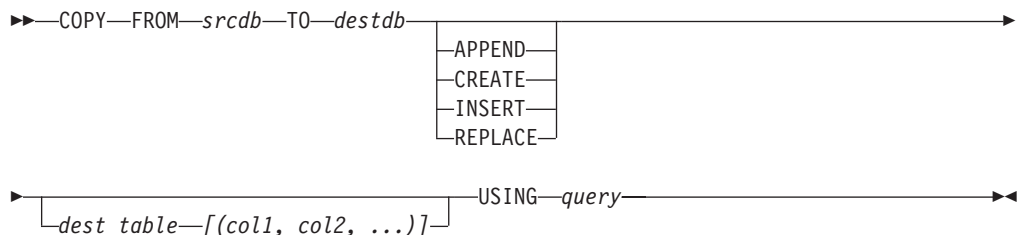
### Authorization

None

### Required connection

None

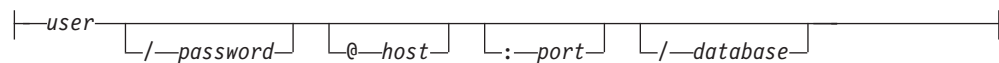
### Command Syntax



#### srcdb, destdb:



#### connection\_identifier:



### Command parameters

#### FROM *srcdb*

Defines the connection details and database name from which the data will be copied.

**Note:** Either one, or both, of the FROM or TO parameters must be specified in the **COPY** command. If FROM is not specified, and TO is specified, the database you are currently connected to, if a connection exists, will be used for the source database.

#### TO *destdb*

Defines the connection details and database name which the data will be copied into.

**Note:** Either one, or both, of the FROM or TO parameters must be specified in the **COPY** command. If TO is not specified, and FROM is specified, the database you are currently connected to, if a connection exists, will be used for the target database.

#### **APPEND**

Inserts data into the *dest\_table*. If *dest\_table* does not exist, the table is created and the data is inserted.

#### **CREATE**

Creates *dest\_table* and inserts the data. If *dest\_table* already exists, an error is returned.

#### **INSERT**

Inserts data into the *dest\_table*. If *dest\_table* does not exist, an error is returned.

#### **REPLACE**

If *dest\_table* exists, it will be dropped, recreated, and then data is inserted.

#### *dest\_table*

Target database table into which data is inserted.

#### *query*

The SQL query used to get the data from the source database.

#### *user*

Specifies the user ID to connect to the database.

#### *password*

Specifies the password that corresponds to the user ID.

#### *hostname*

Specifies the name of the computer on which the database is located. For example, for a computer named ascender, specify @ascender.

#### *port*

Specifies the port number receiving connections on the computer where the database server is installed. The default is 50000.

#### *database*

Specifies the database name to which the connection is made. The default is SAMPLE.

#### *dsn\_alias*

Specifies that the database connection information be picked up from the db2dsdriver.cfg file from the dsn with alias name *dsn\_alias*. If the specified *dsn\_alias* is not found in the db2dsdriver.cfg file, the string *dsn\_alias* is used as a database name and all other connection parameters are obtained interactively.

### **Example**

The following command copies the rows in the emp table in the db1 database into the emp table in the db2 database.

```
COPY FROM u1@db1 TO u2@db2 INSERT emp USING SELECT * FROM emp;
```

---

## DEFINE

The **DEFINE** CLPPlus command creates a user variable, also called a substitution variable, and specifies its value. The command also displays the value of one or more user variables.

### Invocation

This command must be executed from the CLPPlus interface.

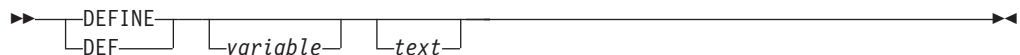
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### *variable*

Specifies the name of a variable. If you specify *variable* without *text*, the name of the variable and its value are displayed. If you do not specify *variable*, the names of all variables and their values are displayed.

#### *text*

Specifies text to assign to the variable specified by *variable*. If the text contains spaces, you must enclose it in double or single quotation marks. If the text does not contain spaces, quotation marks are optional.

### Example

In the following example, the **DEFINE** command defines the DB2, DEPT, and NAME variables and then displays the values of all variables:

```
SQL> DEFINE DEPT = 20
SQL> DEFINE NAME = 'John Smith'
SQL> DEFINE NAME = 'localhost:5445/edb'
SQL> DEFINE
DEFINE DB2 = "localhost:5445/edb"
DEFINE DEPT = "20"
DEFINE NAME = "John Smith"
```

## DEL

The **DEL** command deletes one or more lines from the SQL buffer.

### Invocation

This command must be executed from the CLPPlus interface.

### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

You can use the parameters to specify the start and end of a range of lines to delete from the SQL buffer. If you do not specify any parameters, the current line is deleted.

*n* Specifies a line number.

*n m*

Specifies two line numbers, where the value of *m* is greater than the value of *n*.

\* Indicates the current line.

**LAST | L**

Indicates the last line in the SQL buffer.

### Example

In the following example, the fifth line, containing column **SAL**, and the sixth line, containing column **COMM**, are deleted from the **SELECT** statement in the SQL buffer:

```

SQL> LIST
 1 SELECT
 2   EMPNO
 3   ,ENAME
 4   ,JOB
 5   ,SAL
 6   ,COMM
 7   ,DEPTNO
 8* FROM EMP
  
```

## DEL

```
SQL> DEL 5 6
SQL> LIST
 1 SELECT
 2     EMPNO
 3     ,ENAME
 4     ,JOB
 5     ,DEPTNO
6* FROM EMP
```

The contents of line 7 becomes the contents of line 5, and the contents of line 8 becomes the contents of line 6. The contents of the current line have not changed, but the current line, marked with an asterisk, has changed from line 8 to line 6.



## DESCRIBE

The **DESCRIBE** CLPPlus command displays a list of columns and their data types and lengths for a table view; a list of parameters for a procedure or function; or a list of procedures and functions and their parameters for a package.

As of V9.7 Fix Pack 3, the functionality of the **DESCRIBE** CLPPlus command is extended so that you can specify the type of database object you want to describe. If you do not specify the type of object you want to describe, then all objects found with the given name and schema will be described. The default schema is *CURRENT SCHEMA*.

As of V9.7 Fix Pack 4, the functionality of the **DESCRIBE** CLPPlus command is extended to support temporal tables. Temporal tables are new for DB2 z/OS Version 10.

### Invocation

You can run this command from the CLPPlus interface.

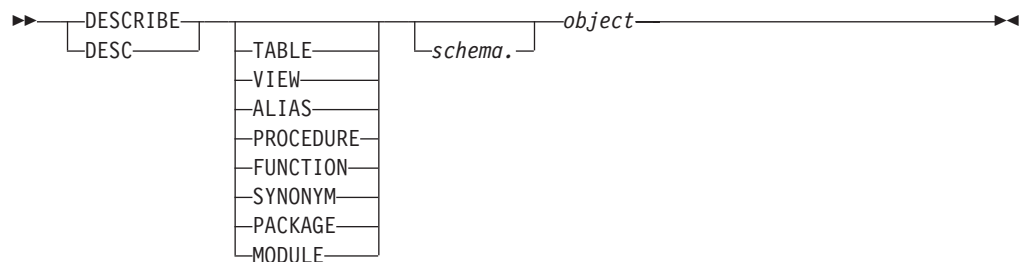
### Authorization

None

### Required connection

You must be connected to a database.

### Command syntax



### Command parameters

#### TABLE

Specifies that the type of database object to be described is a table. Available as of V9.7 Fix Pack 3.

#### VIEW

Specifies that the type of database object to be described is a view. Available as of V9.7 Fix Pack 3.

#### ALIAS

Specifies that the type of database object to be described is an alias. Available as of V9.7 Fix Pack 3.

#### PROCEDURE

Specifies that the type of database object to be described is a procedure. Available as of V9.7 Fix Pack 3.

## DESCRIBE

### FUNCTION

Specifies that the type of database object to be described is a function.  
Available as of V9.7 Fix Pack 3.

### SYNONYM

Specifies that the type of database object to be described is a synonym.  
Available as of V9.7 Fix Pack 3.

### PACKAGE

Specifies that the type of database object to be described is a package.  
Available as of V9.7 Fix Pack 3.

### MODULE

Specifies that the type of database object to be described is a module. Available  
as of V9.7 Fix Pack 3.

### *schema*

Specifies the name of the schema containing an object to be described. The  
default schema is *CURRENT SCHEMA*.

### *object*

Specifies the name of a table, view, procedure, function, or package to be  
described.

## Example

In the following example, the **DESCRIBE** command is run to get details on the table  
named **ABCD**.

```
SQL> DESCRIBE TABLE ABCD ;
```

```
TABLE – ABCD
```

| Name | Data Type | Type schema | Length | Scale | Nulls | Hidden |
|------|-----------|-------------|--------|-------|-------|--------|
| ROLL | INTEGER   | SYSIBM      | 4      | 0     | Y     | N      |
| NAME | CHARACTER | SYSIBM      | 10     | 0     | Y     | P      |

In the following example, the **DESCRIBE** command is run to get details on a  
bitemporal table.

```
SQL > create table policy
(
  policy_id int NOT NULL,
  coverage int NOT NULL IMPLICITLY HIDDEN,
  bus_start date NOT NULL,
  bus_end date NOT NULL,
  system_start TIMESTAMP(12) generated always as row begin NOT NULL,
  system_end TIMESTAMP(12) generated always as row end NOT NULL,
  trans_start generated always as transaction start ID,
  period BUSINESS_TIME(bus_start, bus_end),
  period SYSTEM_TIME (system_start, system_end)
);
```

```
DB250000I: The command completed successfully.
```

```
SQL> create table policy_hist LIKE policy;
```

```
DB250000I: The command completed successfully.
```

```
SQL> ALTER TABLE policy ADD VERSIONING USE HISTORY TABLE policy_hist;
```

```
DB250000I: The command completed successfully.
```

```
SQL> describe policy
```

TABLE - POLICY

\*\*\*\*\*

| Name         | Data Type | Type schema | Length | Scale | Nulls | Hidden   |
|--------------|-----------|-------------|--------|-------|-------|----------|
| POLICY_ID    | INTEGER   | SYSIBM      |        | 4     | 0 N   | Not      |
| COVERAGE     | INTEGER   | SYSIBM      |        | 4     | 0 N   | Implicit |
| BUS_START    | TIMESTAMP | SYSIBM      |        | 7     | 0 N   | Not      |
| BUS_END      | TIMESTAMP | SYSIBM      |        | 7     | 0 N   | Not      |
| SYSTEM_START | TIMESTAMP | SYSIBM      |        | 13    | 12 N  | Not      |
| SYSTEM_END   | TIMESTAMP | SYSIBM      |        | 13    | 12 N  | Not      |
| TRANS_START  | TIMESTAMP | SYSIBM      |        | 13    | 12 Y  | Not      |

Temporal Type : Bitemporal

Table is versioned and has the following periods

| Name          | Type | Begin Column | End Column |
|---------------|------|--------------|------------|
| SYSTEM_TIME   | S    | SYSTEM_START | SYSTEM_END |
| BUSINESS_TIME | A    | BUS_START    | BUS_END    |

\*\*\*\*\*

In the following example, the **DESCRIBE** command is run to get details on a table with system period, but not versioned.

```
SQL> create table demo_nontemp
(
  policy_id int NOT NULL,
  coverage int NOT NULL IMPLICITLY HIDDEN,
  system_start TIMESTAMP(12) generated always as row begin NOT NULL,
  system_end TIMESTAMP(12) generated always as row end NOT NULL,
  trans_start generated always as transaction start ID,
  period SYSTEM_TIME (system_start, system_end)
);
```

DB250000I: The command completed successfully.

SQL&gt; describe demo\_nontemp

TABLE - TEMPTAB

\*\*\*\*\*

| Name         | Data Type | Type schema | Length | Scale | Nulls | Hidden   |
|--------------|-----------|-------------|--------|-------|-------|----------|
| POLICY_ID    | INTEGER   | SYSIBM      |        | 4     | 0 N   | Not      |
| COVERAGE     | INTEGER   | SYSIBM      |        | 4     | 0 N   | Implicit |
| SYSTEM_START | TIMESTAMP | SYSIBM      |        | 13    | 12 N  | Not      |
| SYSTEM_END   | TIMESTAMP | SYSIBM      |        | 13    | 12 N  | Not      |
| TRANS_START  | TIMESTAMP | SYSIBM      |        | 13    | 12 Y  | Not      |

Table has the following periods

| Name        | Type | Begin Column | End Column |
|-------------|------|--------------|------------|
| SYSTEM_TIME | S    | SYSTEM_START | SYSTEM_END |

\*\*\*\*\*

In the following example, the **DESCRIBE** command is run to get details on a application period temporal table.

## DESCRIBE

```
SQL> create table demo_app
(
  policy_id int NOT NULL,
  coverage int NOT NULL IMPLICITLY HIDDEN,
  bus_start date NOT NULL,
  bus_end date NOT NULL,
  period BUSINESS_TIME(bus_start, bus_end));
```

DB250000I: The command completed successfully.

```
SQL> describe demo_app
```

TABLE - DEMO\_APP

\*\*\*\*\*

| Name      | Data Type | Type schema | Length | Scale | Nulls | Hidden   |
|-----------|-----------|-------------|--------|-------|-------|----------|
| POLICY_ID | INTEGER   | SYSIBM      | 4      | 0     | N     | Not      |
| COVERAGE  | INTEGER   | SYSIBM      | 4      | 0     | N     | Implicit |
| BUS_START | TIMESTAMP | SYSIBM      | 7      | 0     | N     | Not      |
| BUS_END   | TIMESTAMP | SYSIBM      | 7      | 0     | N     | Not      |

Temporal Type : Application period temporal

Table has the following periods

```
-----
Name                Type Begin Column  End Column
-----
BUSINESS_TIME       A    BUS_START      BUS_END
-----
```

\*\*\*\*\*

In the following example, the **DESCRIBE** command is run to get details on a system period temporal table.

```
SQL> create table demo_sys
(
  policy_id int NOT NULL,
  coverage int NOT NULL IMPLICITLY HIDDEN,
  system_start TIMESTAMP(12) generated always as row begin NOT NULL,
  system_end TIMESTAMP(12) generated always as row end NOT NULL,
  trans_start generated always as transaction start ID,
  period SYSTEM_TIME (system_start, system_end)
);
```

DB250000I: The command completed successfully.

```
SQL> create table demo_sys_history like demo_sys ;
```

DB250000I: The command completed successfully.

```
SQL> ALTER TABLE DEMO_SYS ADD VERSIONING USE HISTORY TABLE DEMO_SYS_HISTORY;
```

DB250000I: The command completed successfully.

```
SQL> desc demo_sys
```

TABLE - DEMO\_SYS

\*\*\*\*\*

| Name      | Data Type | Type schema | Length | Scale | Nulls | Hidden   |
|-----------|-----------|-------------|--------|-------|-------|----------|
| POLICY_ID | INTEGER   | SYSIBM      | 4      | 0     | N     | Not      |
| COVERAGE  | INTEGER   | SYSIBM      | 4      | 0     | N     | Implicit |

**DESCRIBE**

|              |           |        |    |      |     |
|--------------|-----------|--------|----|------|-----|
| SYSTEM_START | TIMESTAMP | SYSIBM | 13 | 12 N | Not |
| SYSTEM_END   | TIMESTAMP | SYSIBM | 13 | 12 N | Not |
| TRANS_START  | TIMESTAMP | SYSIBM | 13 | 12 Y | Not |

Temporal Type : System period temporal

Table is versioned and has the following periods

-----

| Name        | Type | Begin Column | End Column |
|-------------|------|--------------|------------|
| SYSTEM_TIME | S    | SYSTEM_START | SYSTEM_END |

-----

\*\*\*\*\*

## DISCONNECT

---

### DISCONNECT

The **DISCONNECT** CLPPlus command closes the current database connection but does not exit the CLPPlus session.

#### Invocation

You can run this command from the CLPPlus interface.

#### Authorization

None

#### Required connection

You must be connected to a database.

#### Command syntax

```
▶▶ — DISCONNECT —————▶▶  
    | DISC —————▶▶
```

## EDIT

The **EDIT** CLPPlus command starts an external editor to make large changes to the contents of a file or the SQL buffer.

CLPPlus reads the **EDITOR** and **PATH** system environment variables to establish which external editor is used when the **EDIT** command is started. Any editor of your choice can be specified in the **EDITOR** system environment variable provided it is installed on the system. The location of the binary file for the external editor specified in the **EDITOR** system environment variable location must be included in your **PATH** system environment variable. If these variables are not set or not set properly, the default editor used on Windows operating systems is Notepad. On UNIX and Linux operating systems, it is vi.

### Invocation

You must run this command from the CLPPlus interface.

### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### *path*

Specifies the path, either absolute or relative, to the file specified by the *filename* variable. If no path is specified, the current directory is used.

#### *filename*

Specifies the name of the file to open. If you do not specify a file extension, the .sql extension is used. If you do not specify the *filename* parameter, the contents of the SQL buffer are brought into the editor.

## EXECUTE

The **EXECUTE** CPPPlus command runs a procedure defined in the database to which a connection is currently established. It is also used to define variables and run single-line PL/SQL statements. For a DB2 database connection, when you run this command, a DB2 CALL statement is issued.

### Invocation

You must run this command from the CLPPlus interface.

### Authorization

The privileges held by the authorization ID of the statement must include at least one of the following:

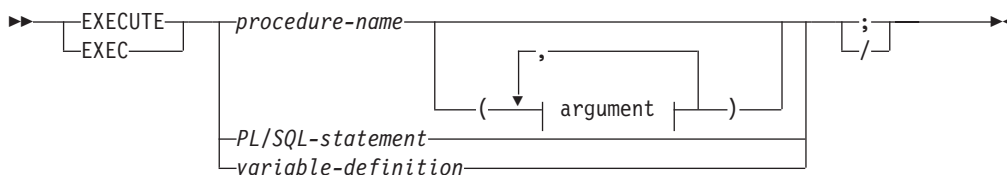
- EXECUTE privilege on the procedure
- DATAACCESS authority

If a matching procedure exists that the authorization ID of the statement is not authorized to run, an error is returned (SQLSTATE 42501).

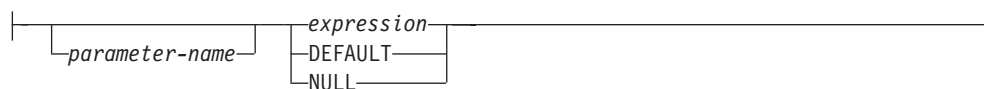
### Required connection

You must be connected to a database.

### Command syntax



### argument:



### Command parameters

#### *procedure-name*

Specifies the procedure to call. The procedure must be cataloged. For DB2 data servers, you can qualify the name with a schema name, a module name, or both a schema name and a module name. The procedure to start is chosen using the procedure resolution. For DB2 databases, the execution of the **EXECUTE** command, including procedure resolution, is the same as the DB2 CALL statement.

#### *PL/SQL-statement*

Specifies the PL/SQL statement to run.

#### *variable-definition*

Specifies the definition of a variable.

#### *argument*



*parameter-name*

For DB2 data servers only, specifies the name of the parameter to which a value is assigned. If you assign a value to a parameter by name, all subsequent value assignments must also be by name.

All named parameters that you use to run the procedure must exist in the DB2 procedure definition.

Parameter names must be unique.

You cannot use named parameters to run uncataloged DB2 procedures.

*value*

Specifies the value associated with a parameter. The *n*th unnamed value corresponds to the *n*th parameter defined in the **CREATE PROCEDURE** statement for the procedure. Named values correspond to the same named parameter, regardless of the order in which you specify them.

*expression*

Passes a user-specified list of parameter values to the procedure being called.

**NULL**

Passes NULL as the parameter value.

**DEFAULT**

If a default value is defined in the **CREATE PROCEDURE** statement, passes that default value as the parameter value; otherwise, passes NULL.

For DB2 databases, you must specify a value for each parameter that is not defined to have a default value (SQLSTATE 428HF). Also, for DB2 databases, each value must be compatible with the corresponding parameter in the procedure definition, as follows:

- IN parameter
  - The value must be assignable to the parameter.
  - The assignment of a string argument uses the storage assignment rules.
- OUT parameter
  - The value must be a single variable or parameter marker (SQLSTATE 42886).
  - The value must be assignable to the parameter.
  - The assignment of a string value uses the retrieval assignment rules.

**Note:** You cannot display the following output data type values in the CLPPlus interface: row, array, associative array, and Boolean.

- INOUT parameter
  - The value must be a single variable or parameter marker (SQLSTATE 42886).
  - The value must be assignable to the parameter.
  - The assignment of a string value uses the storage assignment rules on invocation and the retrieval assignment rules on return.

**Examples**

1. The **CREATE PROCEDURE** statement creates a procedure called `save_tester_details_PROC`. The **EXECUTE** command runs this procedure.

## EXECUTE

```
> SQL> CREATE PROCEDURE save_tester_details_PROC
      (tno, IN integer, tname IN varchar, tadd IN varchar)
      AS
      BEGIN
      INSERT INTO tester1 VALUES
      (tno, tname, tadd);
      END;
/
```

> The SQL command completed successfully.

```
> SQL> EXECUTE save_tester_details_PROC(1, 'John Smith', 'Address1');
> DB250000I: The SQL command completed successfully.
```

2. The **EXECUTE** command spans multiple lines and the block terminator / is used to submit the command for processing. The block terminator / must be used at the end of a command which spans multiple lines.

```
SQL> exec dbms_output.put_line('test serveroutput')
2 /
test serveroutput
DB250000I: The command completed successfully.
```

3. The **EXECUTE** command runs a single PL/SQL statement.

```
SQL> Exec BEGIN dbms_output.put_line('TEST EXEC'); END
2
/
DB250000I: The command completed successfully.
```

4. The **EXECUTE** command defines a variable.

```
SQL> Variable bindvar varchar(20)
SQL> Execute :bindvar := 'value' ;
```

## EXIT

The **EXIT** CLPPlus command ends the CLPPlus session and returns control to the operating system. This command is synonymous with the **QUIT** command.

### Invocation

You must run this command from the CLPPlus interface.

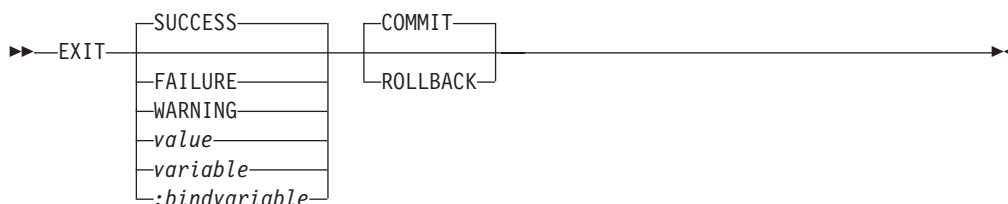
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### SUCCESS

Returns an operating system–dependant return code indicating success.

#### FAILURE

Returns an operating system–dependant return code indicating failure.

#### WARNING

Returns an operating system–dependant return code indicating a warning.

#### *value*

Specifies a variable created by the **DEFINE** command whose value is returned as the return code.

#### *variable*

Specifies a substitution variable value created by the **DEFINE** command whose value is returned as the return code. Available as of V9.7 Fix Pack 3.

#### *:bindvariable*

Specifies a Bind variable value created by the **DEFINE** command whose value is returned as the return code. Available as of V9.7 Fix Pack 3.

#### COMMIT

Specifies that uncommitted updates are committed when the CLPPlus session ends.

#### ROLLBACK

Specifies that uncommitted updates are rolled back when the CLPPlus session ends.

### Examples

## EXIT

In the following example, the **EXIT** command is issued with the **SUCCESS** parameter.

```
SQL> exit success
C:\>echo %errorlevel%
0
```

-----

In the following example, the **EXIT** command is issued with the **WARNING** and **COMMIT** parameters.

```
SQL> exit warning commit
C:\>echo %errorlevel%
2
```

-----

In the following examples, the substitution variable *exit\_value* is defined, set, and used in the **EXIT** command. This functionality is available as of V9.7 Fix Pack 3.

```
SQL> variable exit_value integer
DB250000I: The command completed successfully.
SQL> exec :exit_value:=5 ;
DB250000I: The command completed successfully.
SQL> exit :exit_value rollback
C:\>echo %errorlevel%
5
```

-----

```
SQL> define exit_value = 4
SQL> exit exit_value commit
C:\>echo %errorlevel%
4
```

## EXPLAIN PLAN

The **EXPLAIN PLAN** CLPPlus command retrieves explain plan information for any single SQL statement.

As of V9.7 Fix Pack 5, **EXPLAIN PLAN** is supported on DB2 for z/OS and IBM Informix.

### Invocation

You must run this command from the CLPPlus interface.

### Authorization

None

### Required connection

You must be connected to a database.

### Restrictions

Support on IBM Informix has these restrictions:

- Only SELECT statements are supported.
- You must create and specify a default sbspace name for the **SBSPACENAME** configuration parameter in the ONCONFIG file. This sbspace is used for creating BLOB objects when an explain plan is created.
- To retrieve statistics data from an Informix server, you must have the **Informix**, or equivalent, privilege. Only users with this privilege have access to statistics data.

### Syntax diagram

►►—EXPLAIN—PLAN—FOR—*SQL-statement*—◄◄

### Command parameters

*SQL-statement*

The SQL statement on which explain information is retrieved. For IBM Informix only **SELECT** statements are supported.

### Example

SQL> explain plan for select \* from emp where bonus > 1000 and salary>10000;

| ID | TYPE   | OBJECT_SCHEMA | OBJECT_NAME | PREDICATE_TEXT      |
|----|--------|---------------|-------------|---------------------|
| 1  | RETURN |               |             |                     |
| 2  | TBSCAN | MANSHANB      | EMPLOYEE    | (10000 < Q1.SALARY) |
| 2  | TBSCAN | MANSHANB      | EMPLOYEE    | (1000 < Q1.BONUS)   |

---

## GET

The **GET** CLPPlus command loads the contents of a text file into the CLPPlus SQL buffer.

### Invocation

You must run this command from the CLPPlus interface.

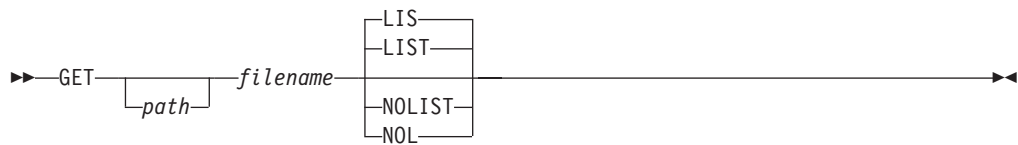
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### *path*

Specifies the path, either absolute or relative, to the file specified by the *filename* variable. If no path is specified, the current directory is used.

#### *filename*

Specifies the name of the file to load into the SQL buffer. If you do not specify a file extension, the .sql extension is used.

#### **LIST | LIS**

Displays the contents of the SQL buffer after the file is loaded.

#### **NOLIST | NOL**

Prevents the contents of the SQL buffer from being displayed after the file is loaded.

---

## HELP

The **HELP** command displays an index of topics for CLPPlus or it displays help for a specific CLPPlus topic.

### Invocation

You must run this command from the CLPPlus interface.

### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### **INDEX**

Displays an index all CLPPlus help topics.

#### *topic*

Displays help for a specific CLPPlus subject, for example, **ACCEPT**.

## HOST

The **HOST** CLPPlus command runs an operating-system command in the CLPPlus interface.

### Invocation

You must run this command in the CLPPlus interface.

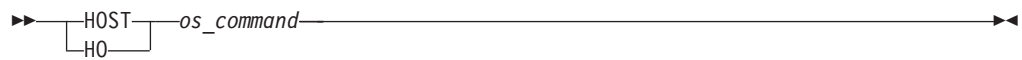
### Authorization

None

### Required connection

None.

### Command syntax



### Command parameters

*os\_command*  
Specifies an operating-system command.



## IMPORT CLPPlus command

As of V9.7 Fix Pack 5, **IMPORT** is supported from a remote CLPPlus client where the import file is processed on the same client.

### Invocation

You must run this command from the CLPPlus interface.

### Authorization

None

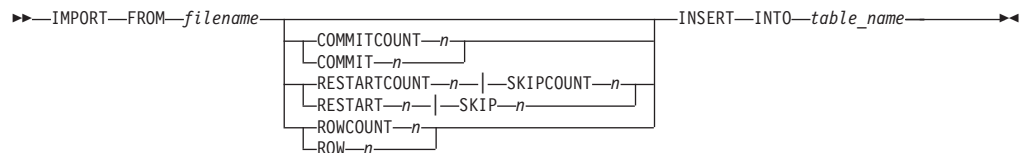
### Required connection

You must be connected to a database.

### Restrictions

- The parameters that are available for the **IMPORT** command in the CLPPlus interface are a subset of the parameters that are available for the **IMPORT** command in the CLP interface.
- Data can be imported only from a delimited file. The , character is the default delimiter. You can set the delimiter to another character by using the **SET CLPPlus** command.
- The target database table cannot be a system table, a declared temporary table, or a summary table.

### Syntax diagram



### Command parameters

#### filename

Specifies the file that contains the import data. You can specify a path as part of the *filename* variable. The path can be absolute or relative to the current directory. If the path is omitted, the current directory is searched.

#### COMMITCOUNT | COMMIT *n*

When specified, **IMPORT** commits data after every *n* records are read and imported.

#### RESTARTCOUNT | RESTART *n*

When specified, **IMPORT** starts at record *n+1*. The first *n* records are skipped. This option is functionally equivalent to **SKIPCOUNT**. **RESTARTCOUNT** and **SKIPCOUNT** are mutually exclusive.

#### SKIPCOUNT | SKIP *n*

When specified, **IMPORT** starts at record *n+1*. The first *n* records are skipped. This option is functionally equivalent to **RESTARTCOUNT**. **RESTARTCOUNT** and **SKIPCOUNT** are mutually exclusive.

## IMPORT CLPPlus command

### **ROWCOUNT** | **ROW** *n*

When specified, *n* physical records from the beginning of *filename* are imported. When **ROWCOUNT** is specified with **RESTARTCOUNT** or **SKIPCOUNT**, **IMPORT** reads *n* rows from *filename* starting from the record defined by **RESTARTCOUNT** or **SKIPCOUNT**.

### *table\_name*

Specifies the target database table for the **IMPORT** operation. This table cannot be a system table, a declared temporary table or a summary table. If not fully qualified with a schema, the default schema is the current ID.

## Examples

The following **IMPORT** command reads the first 100 rows of the `c:\data.txt` file and inserts the data into the `db2admin.emptab` table:

```
import from c:\data.txt rowcount 100 insert into db2admin.emptab;
```

The following **IMPORT** command reads data starting at row 11 of the `data.txt` file and inserts the data into the `emptab` table:

```
import from data.txt skip 10 insert into emptab;
```

The following **IMPORT** command reads data starting at row 11 of the `data.txt` file, one directory up in the directory tree relative to the current directory. The command inserts the data into the `emptab` table.

```
import from ../data.txt restart 10 insert into emptab;
```

## INPUT

The **INPUT** line-editor command adds a line of text to the SQL buffer after the current line.

### Invocation

You must run this command from the CLPPlus interface.

### Authorization

None

### Required connection

None

### Command syntax

### Command parameters

*text*

Specifies the text to be inserted into the SQL buffer.

### Example

In the following example, the sequence of **INPUT** commands constructs a **SELECT** statement, which is displayed by the **LIST** command:

```
SQL> INPUT SELECT empno, ename, job, sal, comm
SQL> INPUT FROM emp
SQL> INPUT WHERE deptno = 20
SQL> INPUT ORDER BY empno
SQL> LIST
 1 SELECT empno, ename, job, sal, comm
 2 FROM emp
 3 WHERE deptno = 20
 4* ORDER BY empno
```

## LIST

The **LIST** line-editor command displays all of the lines or a range of lines in the SQL buffer.

### Invocation

You must run this command from the CLPPlus interface.

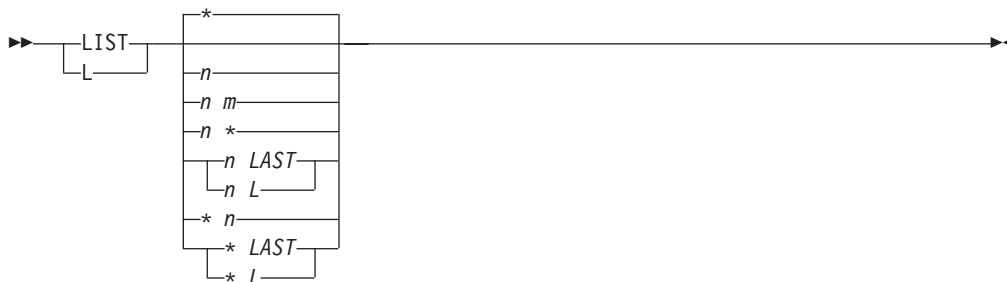
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

*n* Displays the specified line in the SQL buffer.

*n m* Displays the specified range of lines in the SQL buffer, where the value of *m* is greater than the value of *n*.

\*

Displays the current line in the SQL buffer.

**LAST | L**

Displays the last line in the SQL buffer.

### Example

In the following example, the **L** command is used to show line buffer contents.

```
SQL> LIST
 1 SELECT
 2 *
 3 FROM
 4* EMPLOYEE
SQL> L2
 2* *
SQL> L4
 4* EMPLOYEE
SQL> L5
DB250400E: The buffer line number is invalid. Valid values can be between '1' and '4'.
SQL>
```

In the following example, a variant of the **LIST** command is used to show line buffer contents.

```
SQL> begin
2 dbms_output.put_line('list command ');
3 end;
4 /
```

DB250000I: The command completed successfully.

```
SQL> list
1 begin
2 dbms_output.put_line('list command ');
3* end
```

```
SQL> 2
2* dbms_output.put_line('list command ');
```

```
SQL>
```

Note in the previous example how the number 2 is entered on its own in. The result is the contents of line buffer 2 is displayed. You can enter any valid line number in the buffer to display the contents of the line.

---

## PAUSE

The **PAUSE** CLPPlus command suspends CLPPlus processing, optionally displays a message, and waits to resume processing until the ENTER key is pressed.

### Invocation

You must run this command in the CLPPlus interface.

### Authorization

None

### Required connection

You must be connected to a database.

### Command syntax

▶▶ `PAUSE` *optional-text* ▶▶  
    └─ PAU ─┘

### Command parameters

*optional-text*

Specifies a message. The message is a string of characters that can include spaces and special characters. If you use quotation marks around the message, it is included in the command output or statement output. The character case specified as **optional-text** is maintained as entered.

---

## PRINT

The **PRINT** CLPPlus command displays the value of a bind variable. Bind variables are used in place of literal values. If you issue a **SELECT** statement multiple times, you can use bind variables to reduce the number of literal values.

### Invocation

You can run this command in the CLPPlus interface.

### Authorization

None

### Required connection

You must be connected to a database.

### Command syntax

►► `PRINT` `bind-variable-name` ►►  
    └─PRI─┘

### Command parameters

*bind-variable-name*

Specifies a string of characters that can include spaces and special characters. If you do not specify *bind-variable-name*, the values of all bind variables are printed.

If *bind-variable-name* is of the datatype **REFCURSOR**, the result set pointed to by *bind-variable-name* will be read in its entirety and all the rows will be printed following the report formatting specified in the current CLPPlus session.

## PROMPT

---

### PROMPT

The **PROMPT** CLPPlus command prints a line to the display.

Output of the **PROMPT** command is displayed in CLPPlus output.

#### Invocation

You must run this command from the CLPPlus interface.

#### Authorization

None

#### Required connection

None

#### Command syntax



#### Command parameters

*optional-text*

Specifies a message. The message is a string of characters that can include spaces and special characters. If you use quotation marks around the message, it is included in the command output or statement output. The case of letters is maintained.



## QUIT

The **QUIT** CLPPlus command ends the CLPPlus session and returns control to the operating system. This command is synonymous with the **EXIT** command.

### Invocation

You must run this command from the CLPPlus interface.

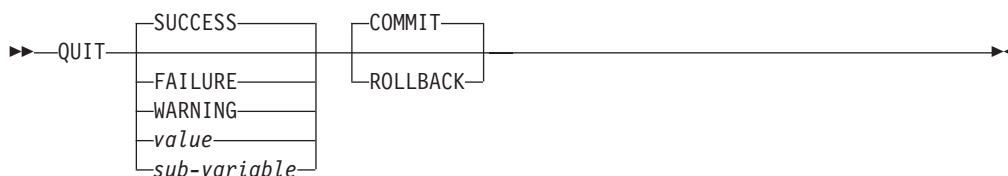
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### **SUCCESS**

Returns an operating system–dependant return code indicating success.

#### **FAILURE**

Returns an operating system–dependant return code indicating failure.

#### **WARNING**

Returns an operating system–dependant return code indicating a warning.

#### *value*

Specifies a variable created by the **DEFINE** command whose value is returned as the return code.

#### *sub-variable*

Specifies a substitution variable that can be used to return information.

#### **COMMIT**

Specifies that uncommitted updates are committed when the CLPPlus session ends.

#### **ROLLBACK**

Specifies that uncommitted updates are rolled back when the CLPPlus session ends.

## REMARK

---

## REMARK

The **REMARK** CLPPlus command allows for the coding of a comment in a script. This is similar to a comment line coded with two dashes.

### Invocation

You must run this command from the CLPPlus interface. It can only be included in CLPPlus scripts.

### Authorization

None

### Required connection

None

### Command syntax

►► REMARK  
REM *optional-text* ◄◄

### Command parameters

*optional-text*

Specifies a string of characters that can include spaces and special characters. You can use the convention shown in the following example.

```
/*  
 * This is a three-line comment  
*/
```



## REORGCHK

```
SQL> reorgchk on table user
```

5. This example performs **REORGCHK** on schema named manshanb.

```
SQL> reorgchk on schema manshanb
```

## REPFOOTER

The **REPFOOTER** CLPPlus command prints a report footer at the end of a report.

### Invocation

You must run this command from the CLPPlus interface.

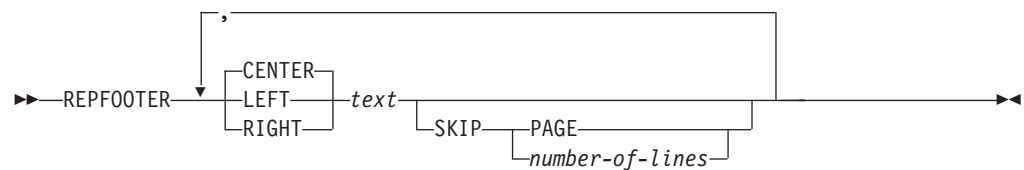
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

*text*

Specifies the text displayed at the end of a report.

#### **CENTER**

Specifies the display will center justify the report footer. If neither **CENTER**, **LEFT**, or **RIGHT** is specified, center justification is the default behavior.

#### **LEFT**

Specifies the display will left justify the text for the report footer.

#### **RIGHT**

Specifies the display will right justify the text for the report footer.

#### **SKIP**

##### **PAGE**

Specifies the report footer is displayed on the next new page.

*number-of-lines*

Specifies the number of lines to skip.

### Example

In the following example, the report footer END SALARY REPORT is displayed with center justification at the end of the report on the next new page.

```
SQL> REPFOOTER CENTER 'END SALARY REPORT' SKIP PAGE;
```

In the following example, the report footer Company Name is displayed, two lines are skipped, then End of Report is displayed all with center justification on the next new page.

```
SQL> REPFOOTER CENTER "Company Name" SKIP 2, CENTER "End of Report" SKIP PAGE
```

## REPHEADER

The **REPHEADER** CLPPlus command prints a report heading once at the beginning of a report.

### Invocation

You must run this command from the CLPPlus interface.

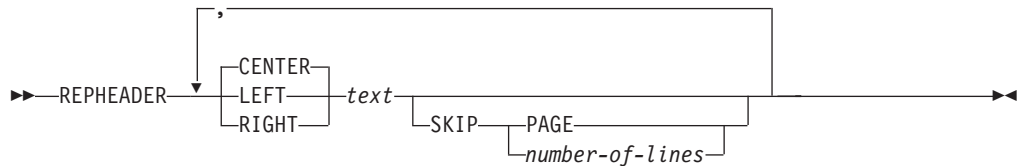
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

*text*

Specifies the text displayed for the report header.

#### CENTER

Specifies the display will center justify the report header. If neither **CENTER**, **LEFT**, or **RIGHT** is specified, center justification is the default behavior.

#### LEFT

Specifies the display will left justify the text for the report header.

#### RIGHT

Specifies the display will right justify the text for the report header.

#### SKIP

##### PAGE

Specifies the report header is displayed and the report data starts on the next new page.

*number-of-lines*

Specifies the number of lines to skip.

### Example

In the following example, the report header SALARY REPORT is displayed with left justification and the report data starts on the next new page.

```
SQL> REPHEADER LEFT 'SALARY REPORT' SKIP PAGE;
```

In the following example, the report header COMPANY NAME is displayed, two lines are skipped, and then SALARY REPORT is displayed all with center justification. The report data starts on the next new page.

```
SQL> REPHEADER CENTER 'COMPANY NAME' SKIP 2, CENTER 'SALARY REPORT' SKIP PAGE;
```

---

## RUN

The **RUN** CLPPlus command runs queries stored in the SQL buffer. Because an SQL buffer can store commands and statements on multiple lines, the command or statement at the current line in the buffer is run.

### Invocation

You must run this command from the CLPPlus interface.

### Authorization

None

### Required connection

You must be connected to a database.

### Command syntax

▶—RUN—◀

### Example

In the following example, the contents of the current line in the SQL buffer is `SELECT EMPNO FROM EMP`. The **RUN** command issues the statement to select the values of the `EMPNO` column from the `EMP` table:

```
SQL> run
1* SELECT EMPNO FROM EMP
```

The output is as follows:

```
EMPNO
-----
000010
000020
...
...
```



## SAVE

The **SAVE** line-editor command stores the contents of the SQL buffer in a new or existing file.

### Invocation

You must run this command from the CLPPlus interface.

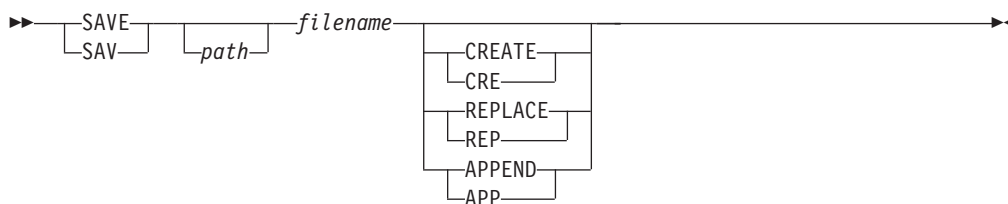
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### *path*

Specifies the path, either absolute or relative, to the file used. If no path is specified, the current directory is used.

#### *filename*

Specifies the name of a file to which the buffer contents are written. If you do not provide a file extension, the `.sql` extension is used.

#### **CREATE | CRE**

Creates the specified file if it does not exist.

#### **REPLACE | REP**

Overwrites the specified file.

#### **APPEND | APP**

Adds the contents of the SQL buffer to the end of the specified file.

---

## SET

The **SET** CLPPlus command controls a session level variable.

**Important:**

- For each invocation of the **SET** command, you can specify only one parameter.
- In a batch script, you can issue several **SET** commands in series.

**Invocation**

You must run this command from the CLPPlus interface.

**Authorization**

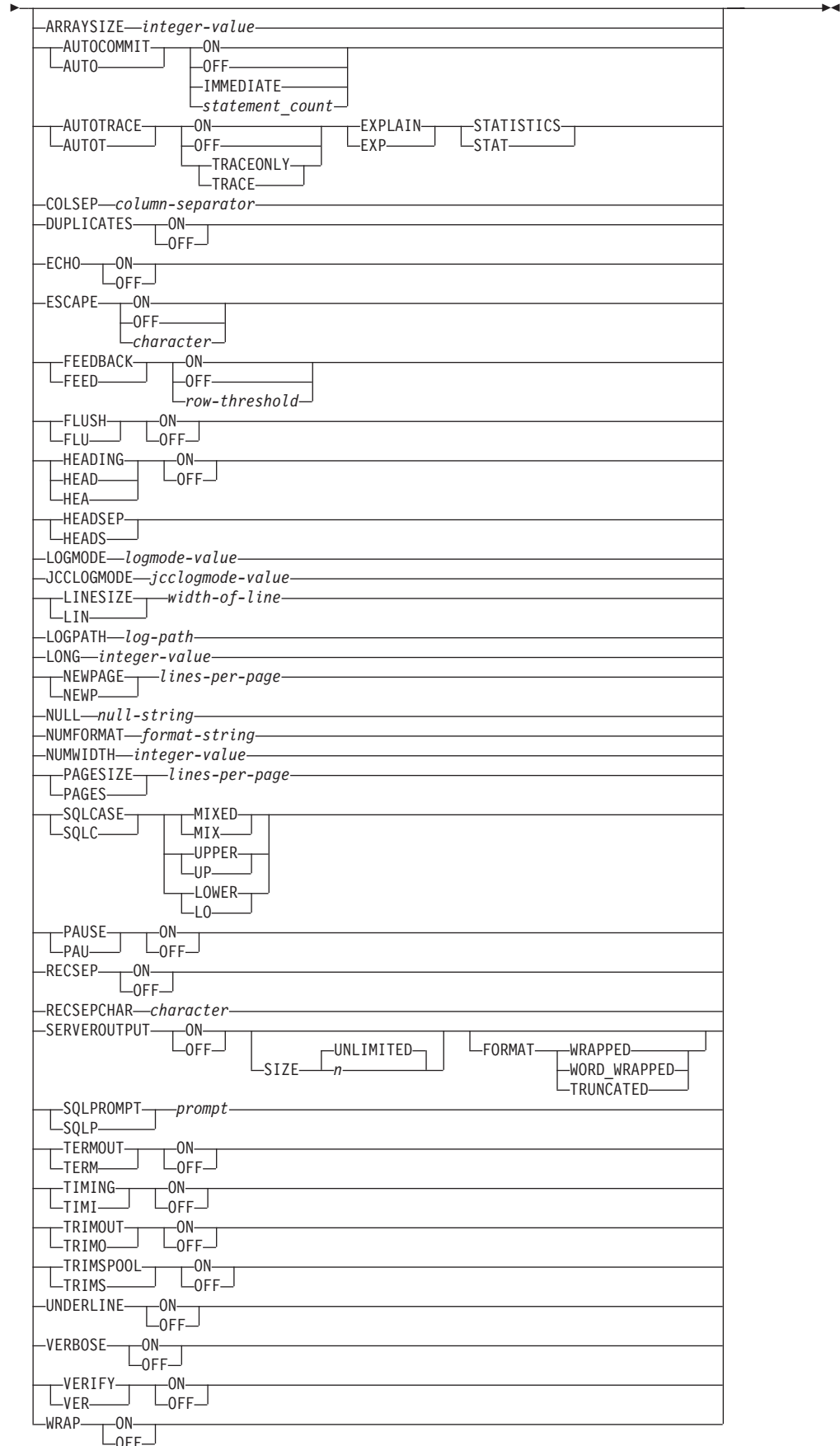
None

**Required connection**

None

**Syntax Diagram**

▶▶—SET—▶▶



## Command parameters

### **ARRAYSIZE** *integer-value*

Defines the number of rows fetched at a time from the server. This option can be used to tune query performance. The default value is 10. Valid values range from 1 to 10,000.

### **AUTOCOMMIT** | **AUTO**

Controls the commit behavior of SQL statements in CLPPlus. CLPPlus always automatically commits DDL statements.

#### **ON** | **IMMEDIATE**

Enables automatic commitment of SQL statements.

#### **OFF**

Disables automatic commitment of SQL statements except for DDL statements.

### **AUTOTRACE** | **AUTOT**

Enables the display of explain plans and statistics information for SQL statements in a CLPPlus session. Once AUTOTRACE is set ON, CLPPlus continues to display the explain information until the session ends or until AUTOTRACE is turned OFF.

As of V9.7 Fix Pack 5, the **AUTOTRACE** parameter is supported on DB2 for z/OS. As of V9.7 Fix Pack 5, the **AUTOTRACE** parameter is also supported on IBM Informix with these restrictions:

- The EXPLAIN option is supported only for SELECT statements.
- If you specify the EXPLAIN, option, you must create and specify a default sbspace name for the **SBSpaceNAME** configuration parameter in the ONCONFIG file. This sbspace name is used for creating BLOBs when an explain plan is created.
- To retrieve statistics data from an Informix server, you must have the Informix privilege or an equivalent privilege.

**ON** Enables AUTOTRACE.

#### **OFF**

Disables AUTOTRACE.

### **TRACEONLY** | **TRACE**

Disables the display of query execution output.

### **EXPLAIN** | **EXP**

Enables the display of the explain plan.

### **STATISTICS** | **STAT**

Enables the display of the statistics information of statements.

### **COLSEP** *column-separator*

Places the specified delimiter between columns in a table. The delimiter must be a character, including a special character or a space (the default value).

### **DUPLICATES**

Controls the printing of duplicate column values for the break columns specified when using the **BREAK** command.

**ON** Enables the printing of duplicate column values.

#### **OFF**

Disables the printing of duplicate column values.

**ECHO**

Controls whether all commands and the associated output are displayed in the standard output of the CLPPlus interface.

**ON** Enables the display of commands and their output.

**OFF**

Disables the display of commands and their output.

**ESCAPE**

Controls whether an escape character is set for use in the CLPPlus interface.

**ON** Enables the default escape character "\".

**OFF**

Disables the currently defined escape character.

*character*

Enables the escape character with the value given to *character*.

**FEEDBACK | FEED**

Controls the display of interactive information after you issue an SQL statement.

**ON** Enables the display of interactive information. This is the default action.

**OFF**

Disables the display of interactive information.

*row-threshold*

Specifies the minimum number of rows required to enable feedback.

**FLUSH | FLU**

Controls whether the output buffer is accessible to external programs. **FLUSH** is still active while the output buffer is being appended to. The process creates extra processing requirements when you call statements or run commands, however.

**ON** Makes the buffer accessible to external programs.

**OFF**

Prevents the buffer from being available to external programs. .

**HEADING | HEA**

Determines whether column headings are displayed for SELECT statements.

**ON** Enables the display of column headings.

**OFF**

Disables the display of column headings.

**HEADSEP | HEADS**

Sets the heading separator character used by the **COLUMN HEADING** command. The default character is |.

**LOGMODE** *logmode-value*

Controls tracing and logging for the CLPPlus client layer and JDBC driver layer (JCC).

**CLPPLUS**

Performs tracing and logging for the CLPPlus client layer only.

**JCC**

Performs tracing and logging for the JDBC client layer only.

## SET

### **BOTH**

Performs tracing and logging for the CLPPlus client layer and JDBC client layer.

### **NONE**

Disables all tracing and logging.

### **JCCLOGMODE** *jcclogmode-value*

Specifies what JCC client layer features should be traced, logged, or both. To use *jcclogmode-value*, you must set **LOGMODE** to 1 or 2.

**0** (TRACE\_NONE)

**1** (TRACE\_CONNECTION\_CALLS)

**2** (TRACE\_STATEMENT\_CALLS)

**4** (TRACE\_RESULT\_SET\_CALLS)

**16** (TRACE\_DRIVER\_CONFIGURATION)

**32** (TRACE\_CONNECTS)

**64** (TRACE\_DRDA\_FLOWS)

**128**

(TRACE\_RESULT\_SET\_META\_DATA)

**256**

(TRACE\_PARAMETER\_META\_DATA)

**512**

(TRACE\_DIAGNOSTICS)

**1024**

(TRACE\_SQLJ)

**2048**

(TRACE\_XA\_CALLS)

**-1** (TRACE\_ALL). By default, the -1 setting is used, meaning that all layers are traced.

### **LINESIZE** | **LIN** *width-of-line*

The keyword **LINESIZE** and its abbreviation **LIN** can be used interchangeably. **LINESIZE** specifies the width of a line in characters.

**Its value is the width of a line in characters with a valid range of 1 to 32767. By default, it is set to 80.**

### **LOGPATH**

**LOGPATH** sets a path name for the file where log records are to be written on the database server.

*log-path*

The path of a file name used to keep logging records of traces using **LOGMODE** and **JCCLOGMODE**.

### **LONG** *integer-value*

Defines the number of characters displayed for large text objects such as CLOB and XML. The default value is 50. The valid range is 1 to 2,147,483,647 characters.

### **NEWPAGE** | **NEWP**

**NEWPAGE** controls how many blank lines are printed after a page break.

*lines-per-page*

An integer value that does not include negative numbers or values greater than 100. By default it is set to 1 to indicate one blank line is printed after a page break. The command `SET NEWPAGE 0` prints a form-feed at the start of each new page.

**NULL**

The keyword NULL is the string that is displayed when showing a null value for a displayed column value in the output buffer.

*null-string*

A string of characters that can include spaces and special characters. By default it is set to a space. The use of quotation marks around *null-string* has no effect on its value. The case of letters is maintained.

**NUMFORMAT** *format-string*

Sets the default format string used for displaying numbers. The supported formats are the same as those for **COLUMN FORMAT** *format-string*.

**NUMWIDTH** *integer-value*

Sets the default width used to display numbers. The default value is 10.

**PAGESIZE** | **PAGES** *lines-per-page*

Sets the number of printed lines that fit on a page. By default it is set to 14. Valid values can be 0, or between 2 and 50000.

**SQLCASE** | **SQLC**

Controls whether the characters in SQL statements that are transmitted to the server are converted to uppercase or lowercase letters.

**MIXED** | **MIX**

Specifies that a string of characters can contain uppercase and lowercase letters.

**UPPER** | **UP**

Specifies that a string of characters can contain only uppercase letters.

**LOWER** | **LO**

Specifies that a string of characters can contain only lowercase letters.

**PAUSE** | **PAU**

Determines whether to stop a process before each page break. If the output cannot be displayed in one page, you are prompted with the message Hit ENTER to continue before each page break.

**ON** Pauses the display of output.

**OFF**

Displays the output without pausing.

**RECSEP**

Specifies whether the record separating character set with the **RECSEPCHAR** option is displayed after printing each record in the result set.

**ON** The record separating character is printed following each record in the result set.

**OFF**

The record separating character is not printed.

**RECSEPCHAR** *character*

Specifies the record separating character used in conjunction with the **RECSEP** option. A single character is accepted as the value for this option.

**SERVEROUTPUT**

Specifies whether output messages from server-side procedures are retrieved and displayed on the client console. As of V9.7 Fix Pack 3, SERVEROUTPUT can take SIZE and FORMAT as options.

**ON** Server-side procedure output messages are retrieved and displayed.

**OFF**

Server-side procedure output messages are not retrieved and displayed.

**SIZE**

Specifies the number of characters displayed on the screen. The default value is UNLIMITED. *n* can be any positive integer.

**FORMAT**

Specifies the format style used to display server output on the console.

WRAPPED enables text to overflow to the next line as needed.

WORD\_WRAPPED enables text to overflow to the next line and will not split words across lines. TRUNCATED truncates text that exceeds the linesize. You must specify one of these options when using FORMAT.

**SQLPROMPT | SQLP *prompt***

Specifies the prompt in the CLPPlus interface. By default, the prompt is SQL>. The prompt must be a string, which can include special characters and spaces. The use of quotation marks around the string has no affect on its value; the case of letters is maintained.

**TERMOUT | TERM**

Determines if output is displayed on the standard output of the CLPPlus interface.

**ON** Displays the output on the screen.

**OFF**

No output is displayed.

**TIMING | TIMI**

Controls whether elapsed time is displayed for each SQL statement after it is issued.

**ON** Specifies that elapsed time is displayed.

**OFF**

Specifies that elapsed time is not displayed.

**TRIMOUT | TRIMO**

Controls whether trailing blank spaces are removed from the output before it is written to the console. By default, trailing blank spaces are not removed.

**ON** Specifies that trailing blank spaces are removed.

**OFF**

Specifies that trailing blank spaces are not removed.

**TRIMSPool | TRIMS**

Controls whether trailing blank spaces are removed from the spool output before it is written to the spool file. By default, trailing blank spaces are not removed.

**ON** Specifies that trailing blank spaces are removed.

**OFF**

Specifies that trailing blank spaces are not removed.



**UNDERLINE**

Specifies if column headings are underlined.

**ON** Column headings are underlined.

**OFF**

Column headings are not underlined.

**VERBOSE**

Determines whether all CLPPlus messages are printed to the console. The default value is OFF.

**ON** Specifies that all CLPPlus messages are printed to the console.

**OFF**

Specifies that only a subset of messages are printed to the console.

**VERIFY | VER**

Determines whether the old and new values of an SQL statement are displayed when a substitution variable is encountered.

**ON** Specifies that the old and new values are displayed.

**OFF**

Specifies that the old and new values are not displayed.

**WRAP**

Sets the default alignment used when displaying column values.

**ON** Column values exceeding the column width are wrapped.

**OFF**

Column values exceeding the column width are truncated.

---

## SPOOL

The **SPOOL** CLPPlus command echoes and logs CLPPlus output to a file.

### Invocation

You must run this command from the CLPPlus interface.

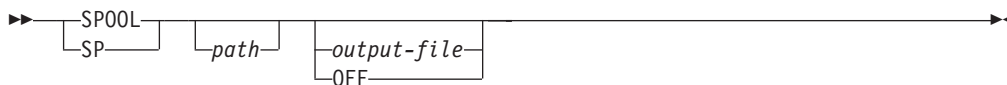
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### *path*

Specifies the path, either absolute or relative, to the output file. If no path is specified, the current directory is used.

#### *output-file*

The variable *output-file*, when set, turns on SPOOL. The standard output of the CLPPlus interface displays command output which is now echoed and logged in the specified *output-file*.

#### **OFF**

The keyword option OFF turns SPOOL off and restores the default which sends all output to the standard output of the CLPPlus interface.

## SHOW

The **SHOW** CLPPlus command displays the current settings of session-level variables in the CLPPlus interface or errors returned from server-side procedures. The settings are controlled using the **SET** command.

### Invocation

You must run this command from the CLPPlus interface.

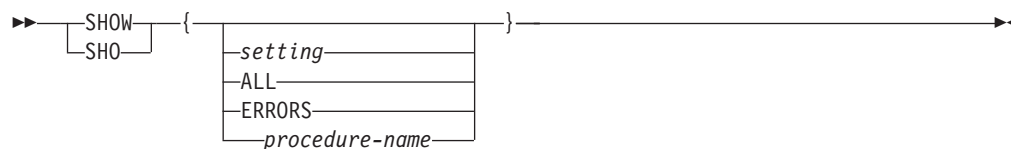
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

*setting*

Displays the name and setting of the specified session-level variable.

**ALL**

Displays the names and settings of all session-level variables.

**ERRORS**

Displays the errors for all server-side procedures run in the current CLPPlus session.

*procedure-name*

When appended to the **SHOW ERRORS** command, shows only the errors for *procedure-name*.

## START

---

## START

The **START** CLPPlus command runs a CLPPlus script file.

### Invocation

You must run this command from the CLPPlus interface.

### Authorization

None

### Required connection

None

### Command syntax

▶▶ `START` `script-file` ▶▶  
    └─STA─┘     └─path─┘

### Command parameters

*path*

Specifies the path, either absolute or relative, to the script file that contains SQL statements and commands to run. If no path is specified, the current directory is used.

*script-file*

Specifies the script file name that contains SQL statements and commands to run.

---

**TTITLE**

The **TTITLE** CLPPlus command inserts text at the top of each page displayed.

### Invocation

You must run this command from the CLPPlus interface.

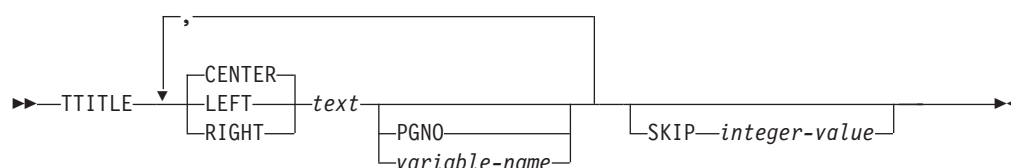
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

*text*

Specifies the text to be displayed.

#### **CENTER**

Specifies the display will center justify the text on each page. If neither **CENTER**, **LEFT**, or **RIGHT** is specified, center justification is the default behavior.

#### **LEFT**

Specifies the display will left justify the text on each page.

#### **RIGHT**

Specifies the display will right justify the text on each page.

#### **PGNO**

Specifies the current page number.

*variable-name*

Specifies a user defined variable that will follow the *text* field.

#### **SKIP integer-value**

The *integer-value* value specifies the number of blank lines displayed after the bottom title.

### Example

In the following example, the DEPT: (with the variable contents), CONFIDENTIAL, and Page No: (with the current page number) is displayed across the top of every page. One blank line follows the top title.

```

SQL> BREAK ON workdept SKIP PAGE;
SQL> COLUMN workdept NEW_VALUE new_dept;
SQL> TTITLE LEFT 'DEPT: ' new_dept, CENTER 'CONFIDENTIAL', RIGHT 'Page No: ' PGNO SKIP 1;

```

## TTITLE

In the following example, the Page No: title (with the current page number) is displayed across the top of every page with right justification. Two blank lines follow the top title.

```
SQL> TTITLE RIGHT 'Page No: ' PGNO SKIP 2;
```

---

## UNDEFINE

The **UNDEFINE** CLPPlus command clears and deletes a variable created by the **DEFINE** CLPPlus command.

### Invocation

You must run this command from the CLPPlus interface.

### Authorization

None

### Required connection

You must be connected to a database.

### Command syntax



### Command parameters

*variable-name*

Specifies the name of the variable to clear and delete.

## WHENEVER OSERROR

As of V9.7 Fix Pack 3, the **WHENEVER OSERROR** CLPPlus command specifies the action CLPPlus performs when an operating system error occurs. This command allows you to trap errors and control CLPPlus behavior by performing specified actions like **EXIT** or **CONTINUE**.

### Invocation

This command must be executed from the CLPPlus interface.

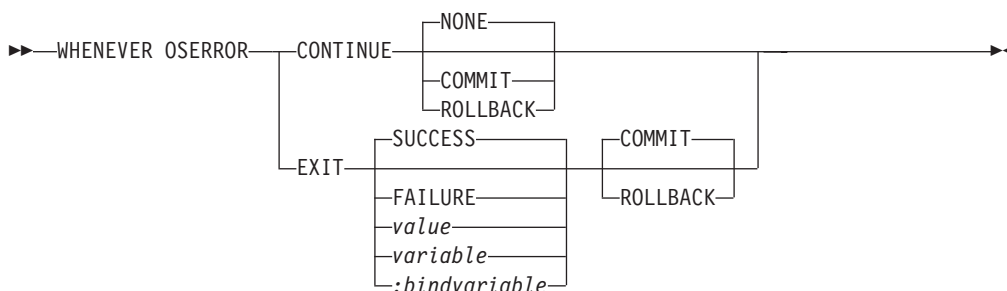
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### CONTINUE

Directs CLPPlus to continue with a specified action when an SQL or PL/SQL error is encountered.

#### NONE

The default value used in the **WHENEVER OSERROR CONTINUE** command. No action on the block of SQL generating an error is taken.

#### COMMIT

When **COMMIT** is specified in the **WHENEVER OSERROR CONTINUE** command, any possible work done by the current SQL block is committed.

#### ROLLBACK

When **ROLLBACK** is specified in the **WHENEVER OSERROR CONTINUE** command, all work in the current SQL block is rolled back.

#### EXIT

Directs CLPPlus to exit once an operating system error is encountered. The functionality of this option is the same as the stand-alone **EXIT** command.

#### SUCCESS

Returns an operating system–dependant return code indicating success. This is the first default **EXIT** parameter.



**FAILURE**

Returns an operating system–dependant return code indicating failure.

*value*

Specifies a variable created by the **DEFINE** command whose value is returned as the return code.

*variable*

Specifies a substitution variable value created by the **DEFINE** command whose value is returned as the return code.

*:bindvariable*

Specifies a Bind variable value created by the **DEFINE** command whose value is returned as the return code.

**COMMIT**

Specifies that uncommitted updates are committed when the CLPPlus session ends. This is the second default **EXIT** parameter.

**ROLLBACK**

Specifies that uncommitted updates are rolled back when the CLPPlus session ends.

**Examples**

The following example shows the command behavior when **EXIT** and an exit error value are specified.

```
SQL> whenever oserror exit -1
SQL> get c:\nonexistingfile.sql
DB250204E: An attempt to locate a file 'c:\nonexistingfile.sql' failed. The co
mmand cannot be processed.
c:\>echo %errorlevel%
-1
```

The following example shows the command behavior when **CONTINUE** is specified.

```
SQL> whenever oserror continue
SQL> get c:\nonexistingfile.sql
DB250204E: An attempt to locate a file 'c:\nonexistingfile.sql' failed. The co
mmand cannot be processed.
SQL>
```

## WHENEVER SQLERROR

As of V9.7 Fix Pack 3, the **WHENEVER SQLERROR** CLPPlus command specifies the action CLPPlus performs when an SQL error occurs in SQL or PL/SQL. This command allows you to trap errors and control CLPPlus behavior by performing specified actions like **EXIT** or **CONTINUE**.

### Invocation

This command must be executed from the CLPPlus interface.

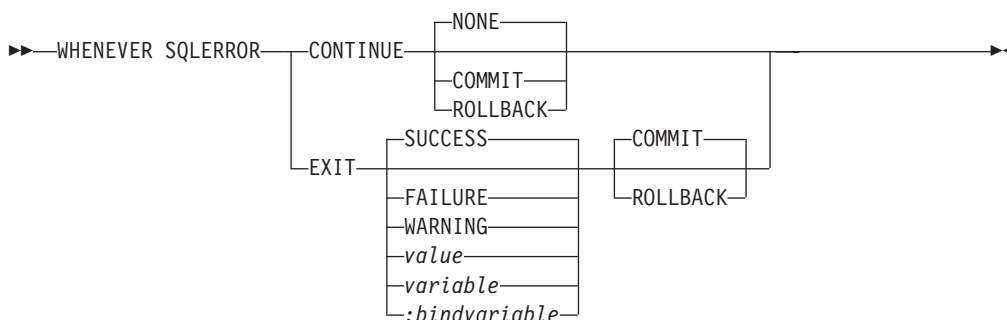
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### CONTINUE

Directs CLPPlus to continue with a specified action when an SQL or PL/SQL error is encountered.

#### NONE

The default value used in the **WHENEVER SQLERROR CONTINUE** command. No action on the block of SQL generating an error is taken.

#### COMMIT

When **COMMIT** is specified in the **WHENEVER SQLERROR CONTINUE** command, any possible work done by the current SQL block is committed.

#### ROLLBACK

When **ROLLBACK** is specified in the **WHENEVER SQLERROR CONTINUE** command, all work in the current SQL block is rolled back.

#### EXIT

Directs CLPPlus to exit once an SQL or PL/SQL error is encountered. The functionality of this option is the same as the stand-alone **EXIT** command.

#### SUCCESS

Returns an operating system–dependant return code indicating success. This is the first default **EXIT** parameter.

### **FAILURE**

Returns an operating system–dependant return code indicating failure.

### **WARNING**

Returns an operating system–dependant return code indicating a warning.

### *value*

Specifies a variable created by the **DEFINE** command whose value is returned as the return code.

### *variable*

Specifies a substitution variable value created by the **DEFINE** command whose value is returned as the return code.

### *:bindvariable*

Specifies a Bind variable value created by the **DEFINE** command whose value is returned as the return code.

### **COMMIT**

Specifies that uncommitted updates are committed when the CLPPlus session ends. This is the second default **EXIT** parameter.

### **ROLLBACK**

Specifies that uncommitted updates are rolled back when the CLPPlus session ends.

## **Examples**

The following example shows the **WHENEVER SQLERROR CONTINUE** command behavior. The CLPPlus prompt is returned and CLPPlus is still available for use.

```
SQL> whenever sqlerror continue
SQL> select * from nonexistingtable;
SQL0204N "SCHEMA.NONEXISTINGTABLE" is an undefined name.
SQL>
```

You can also commit, rollback, or take no action whenever an SQL error occurs.

```
SQL> whenever sqlerror continue commit
SQL>
SQL> whenever sqlerror continue commit
SQL>
SQL> whenever sqlerror continue none
SQL>
```

The following examples use the **EXIT** option to exit the CLPPlus application.

```
SQL> whenever sqlerror exit
SQL> select * from nonexistingtable;
SQL0204N "SCHEMA.NONEXISTINGTABLE" is an undefined name.
C:\>
```

The following specify the error code returned during exit. This behavior is identical to the **EXIT** CLPPlus command.

```
SQL> whenever sqlerror exit success
SQL> whenever sqlerror exit failure
SQL> select * from nonexistingtable;
SQL0204N "SCHEMA.NONEXISTINGTABLE" is an undefined name.
```

```
C:\>echo %errorlevel%
1
```

## WHENEVER SQLERROR

```
SQL> define exit_value=6
SQL> whenever sqlerror exit exit_value
SQL> select * from nonexistingtable;
SQL0204N "SCHEMA.NONEXISTINGTABLE" is an undefined name.
```

```
C:\echo %errorlevel%
6
```

Similar to the EXIT CLPPlus command, you can specify whether to commit or rollback while exiting the CLPPlus application.

```
SQL> whenever sqlerror exit 2 commit
```

```
SQL> whenever sqlerror exit 2 rollback
```

---

## Chapter 7. System commands

---

## dasauto - Autostart DB2 administration server

Enables or disables autostarting of the DB2 administration server.

**Important:** This command has been deprecated and might be removed in a future release because the DB2 administration server (DAS) has been deprecated in Version 9.7. For more information, see “DB2 administration server (DAS) has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html>.

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.

## dasCRT - Create a DB2 administration server

The DB2 Administration Server provides support services for remote administration and DB2 administration tools such as the Replication Center. If a system does not have a DAS, you can use this command to manually generate it.

**Important:** This command has been deprecated and might be removed in a future release because the DB2 administration server (DAS) has been deprecated in Version 9.7. For more information, see “DB2 administration server (DAS) has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html>.

The **dasCRT** 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 operating systems only. On Windows operating systems, you can use the **db2admin create** command for the same purpose.

**Note:** The DB2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in DB2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see “DB2 administration server (DAS) has been deprecated” at .

### Authorization

Root user authority

### Required connection

None

### Command syntax

```
▶▶ dasCRT -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.

The following restrictions apply:

- If existing IDs are used to create DB2 DAS, make sure that the IDs are not locked and do not have expired passwords.

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

### Usage notes

- On AIX 6.1 (or higher), when running this command from a shared DB2 copy in a system workload partition (WPAR) global environment, this command must be run as the root user. WPAR is not supported in a DB2 pureScale environment.

### dasdrop - Remove a DB2 administration server

On Linux and UNIX operating systems only, removes the DB2 Administration Server (DAS).

**Important:** This command has been deprecated and might be removed in a future release because the DB2 administration server (DAS) has been deprecated in Version 9.7. For more information, see “DB2 administration server (DAS) has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html>.

The DB2 Administration Server provides support services for remote administration and DB2 administration tools such as the Replication Center. On Windows operating systems, you can use the **db2admin drop** command for the same purpose.

**Note:** The DB2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in DB2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see “DB2 administration server (DAS) has been deprecated” at .

#### Authorization

Root user 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.
- On AIX 6.1 (or higher), when running this command from a shared DB2 copy in a system workload partition (WPAR) global environment, this command must be run as the root user. WPAR is not supported in a DB2 pureScale environment.



## 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.

**Important:** This command has been deprecated and might be removed in a future release because the DB2 administration server (DAS) has been deprecated in Version 9.7. For more information, see “DB2 administration server (DAS) has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html>.

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 operating 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 user authority on UNIX operating systems or Local Administrator authority on Windows operating systems

### Required connection

None

### Command syntax

For Linux and UNIX operating systems

```
▶▶ dasmigr [ -d ]
```

For Windows operating systems

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

### Command parameters

For Linux and UNIX operating systems:

**-d** Enters debug mode, for use with DB2 database support.

For Windows operating systems:

**-h** Displays usage information.

## dasmigr - Migrate the DB2 administration server

**-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.

### Example

On Linux and UNIX operating systems:

```
DB2DIR/instance/dasmigr
```

On Windows operating systems:

```
DB2DIR\bin\dasmigr
```

### Usage notes

- On AIX 6.1 (or higher), when running this command from a shared DB2 copy in a system workload partition (WPAR) global environment, this command must be run as the root user. WPAR is not supported in a DB2 pureScale environment.

## dasupdt - Update DAS

On Linux and UNIX operating systems, updates the DB2 Administration Server (DAS) if the related DB2 database system installation is updated. On Linux, UNIX, and Windows operating systems, 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.

**Important:** This command has been deprecated and might be removed in a future release because the DB2 administration server (DAS) has been deprecated in Version 9.7. For more information, see “DB2 administration server (DAS) has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html>.

This utility is located in the *DB2DIR*/instance directory on Linux and UNIX operating systems, where *DB2DIR* is the location where the current version of the DB2 database product is installed. On Windows operating systems, the **dasupdt** command is located in the *DB2DIR*\bin directory.

On Windows operating systems, this command updates the DAS from one DB2 copy to another within the same DB2 database version. To upgrade a DAS from an older version, use the **dasmigr** command. With **dasupdt**, the DAS will be updated to the DB2 copy that the **dasupdt** command is executed from.

After the installation of a fix pack, the **dasupdt** command is executed automatically, if the DAS on the system is related to the DB2 database product installation path updated by **installFixPack**.

### Authorization

Root user authority on Linux and UNIX operating systems or Local Administrator authority on Windows operating systems

### Required connection

None

### Command syntax

For Linux and UNIX operating systems

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

For Windows operating systems

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

### Command parameters

For Linux and UNIX operating systems

**-d** Sets the debug mode, which is used for problem analysis.

## dasupdt - Update DAS

- D** Moves the DAS from a higher code level on one path to a lower code level installed on another path.
- h | -?** 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.

### Example

If a DAS is running in one DB2 database product 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
```

### Usage notes

- On AIX 6.1 (or higher), when running this command from a shared DB2 copy in a system workload partition (WPAR) global environment, this command must be run as the root user. WPAR is not supported in a DB2 pureScale environment.

## db2\_deinstall - Uninstall DB2 database products, features, or languages

Uninstalls DB2 database products, features, or languages, depending on the command parameters and the location where the **db2\_deinstall** command is run. The command is only available on Linux and UNIX operating 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 is also available on DB2 database product media. The **db2\_deinstall** command can be used to uninstall only the DB2 database products related to the installation path.

- If **db2\_deinstall -a** is run from a particular DB2 database installation path, then it can uninstall everything, or a particular feature or language, from the same path on the local host.
- If **db2\_deinstall -a** is run from the DB2 database product media, then you need to specify a path using the **-b** option. It can then uninstall everything, or a particular feature or language, from that install path on that local host.

### Authorization

Root installations require root user authority.

### Required Connection

None

### Command syntax

```

▶▶ db2_deinstall --F feature-name -a
                                   [-s GPFS] [-s TSAMP]
                                   [-r response-file]
▶ [-f sqllib] [-b installpath] [-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**.

Cannot be used in combination with **-a**, except in one case. When the feature being removed is the IBM Tivoli System Automation for Multiplatforms, Tivoli SA MP, and you have root user authority, you can use **-F TSAMP** and **-a** in combination, which removes both Tivoli SA MP and DB2 database products together.

Can be used in combination with **-r**, except in one case. When the feature being removed is the IBM Tivoli System Automation for Multiplatforms, Tivoli SA MP, you cannot use **-F TSAMP** and **-r** in combination.

## db2\_deinstall - Uninstall DB2 database products, features, or languages

The DB2 database uninstaller will automatically update the related DB2 instances after it removed some DB2 features. If instance update failed as reported in the log file, you need to manually update the related DB2 instances with the **db2iupdt** (root instances) or **db2nrupdt** (non-root instance) command.

### **-f sqllib**

This parameter is only valid for non-root install. When it is used with **-a**, the instance top directory and anything underneath will be removed.

### **-a**

This parameter is mandatory when running the **db2\_deinstall** command on a IBM DB2 pureScale Feature copy. Removes all installed DB2 database products in the current location. Cannot be used in combination with **-F**, except in one case. When the feature being removed is the IBM Tivoli System Automation for Multiplatforms, Tivoli SA MP, and you have root user authority, you can use **-F TSAMP** and **-a** in combination, which removes both Tivoli SA MP and DB2 database products together.

Cannot be combined with the **-r** parameter.

In a non-root install, **-a** used with **-f sqllib** will also remove the non-root instance, which includes removal of the `$HOME/sqllib` directory.

### **-s GPFS**

If IBM General Parallel File System (GPFS™) was installed as part of a DB2 pureScale Feature installation, GPFS is also uninstalled automatically. Specifying this parameter skips the removal of GPFS. You may want to use this parameter to keep the GPFS cluster and its file systems, but remove the DB2 copy. This parameter can only be combined with the **-a** parameter.

### **-s TSAMP**

If IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) was installed as part of a DB2 pureScale Feature installation, Tivoli SA MP is also uninstalled automatically. Specifying this parameter skips the removal of Tivoli SA MP. You may want to use this parameter to keep the RSCT peer domain, but remove the DB2 copy. This parameter can only be combined with **-a** or **-F** parameters.

### **-r response-file**

Performs an uninstallation of products, features, or languages based on what is specified in the response file. For example, `db2_deinstall -r db2un.rsp`. The DB2 product image includes ready-to-use sample response files with default entries. The sample response file to uninstall products, features, or languages is `db2un.rsp`. The `db2un.rsp` file can be found in `DB2DIR/install` path.

Cannot be combined with the **-a** parameter, or **-s** parameter.

Can be combined with the **-F** parameter, except in one case. When the feature being removed is the IBM Tivoli System Automation for Multiplatforms, Tivoli SA MP, you cannot use **-F TSAMP** and **-r** in combination.

If both the **-r** and **-F** parameters are specified, the DB2 features specified in the **-F** parameter override the **REMOVE\_COMP** keywords in the response file.

The DB2 database uninstaller will automatically update the related DB2 instances after it removed some DB2 features. If instance update failed as

## db2\_deinstall - Uninstall DB2 database products, features, or languages

reported in the log file, you need to manually update the related DB2 instances with the **db2iupdt** (root instances) or **db2nrupdt** (non-root instance) command.

### **-b**

This parameter is valid if the command is run from the DB2 database product media. It specifies the absolute path where the DB2 database product was installed and will be uninstalled. The command will prompt for the path if the parameter is not specified.

### **-l** *log-file*

Specifies the log file. For root installations, the default log file is `/tmp/db2_deinstall.log$$`, where `$$` represents the process ID.

For non-root installations, the default log file is `/tmp/db2_deinstall_<userID>.log`, where *userID* represents the user ID that owns the non-root installation. When the feature being removed is the IBM Tivoli System Automation for Multiplatforms, Tivoli SA MP, the install log file for SA MP will be located in the same directory as DB2 database log files.

### **-t** *trace-file*

Turns on the debug mode. The debug information is written to the file name specified as *trace-file*.

### **-h | -?**

Displays help information.

## Example

To uninstall all the DB2 database products that are installed in a location (*DB2DIR*), issue the **db2\_deinstall** command located in the *DB2DIR/install* directory:

```
DB2DIR/install/db2_deinstall -a -l /tmp/db2_deinstall.log  
-t /tmp/db2_deinstall.trc
```

## Usage notes

- If you run **db2\_deinstall -a**, only the components and features installed by the DB2 database installer are removed.
- Before running the **db2\_deinstall** command, you must manually remove the IBM General Parallel File System (GPFS) cluster and file system. Otherwise, you must run the **db2\_deinstall** command with the **-s** parameter to skip the removal of GPFS binary files. You can also run the **db2\_deinstall** command with the **-s** parameter to skip the removal of Tivoli SA MP.

## db2\_install - Install DB2 database product

Installs all features of a DB2 database product to the given path. This command is available only on UNIX operating systems.

**Important:** This command is deprecated and might be removed in a future release. Use the **db2setup** command instead.

### Authorization

Root installations require root user authority. For non-root installations, you must log on with the user ID that owns the non-root installation.

### Required Connection

None

### Command syntax

```

▶▶ db2_install [-b install-path] [-f NOTSAMP
                PURECLUSTER
                PURESACLE]
                [-f nobackup] [-f ignoreType1] [-p productShortName]
                [-c NLPACK_location] [-n] [-m] [-L language] [-l log-file]
                [-t trace-file] [-h] [-?]

```

### Command parameters

#### **-b** *install-path*

Specifies the path where the DB2 database 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.

The **-b** option is not required for a non-root installation of the DB2 database product, but it is still mandatory for a root installation, if the **-n** option is used. If **-b** is used in a non-root install, the path must be valid, which has to be the user's \$HOME/sql11ib. For both root and non-root install, the length of the absolute installation path is limited to 128 bytes.

#### **-f** NOTSAMP, **-f** PURECLUSTER or **-f** PURESACLE

The **-f** PURECLUSTER, **-f** PURESACLE and **-f** NOTSAMP parameters are mutually exclusive. The PURECLUSTER or PURESACLE option applies to products that support the IBM DB2 pureScale Feature only. The PURECLUSTER or PURESACLE option is ignored for products that do not support the DB2 pureScale Feature.



### NOTSAMP

Specifies that the IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) should not be either installed or updated.

### PURECLUSTER or PURESACLE

Specifies that the DB2 pureScale Feature will be installed.

### **-f** *nobackup*

This applies to the non-root upgrade only. Forces **db2\_install** to not backup installation files when the components are updated. If you choose not to backup the files, the space requirement of the installation directory is reduced. However, choosing not to backup the files also means that if any errors occur, the DB2 database installer will not be able to perform a rollback operation. In this case, you will need to manually clean up the files and reinstall the product.

### **-f** *ignoreType1*

This applies to the non-root upgrade only. Forces **db2\_install** to ignore type-1 indexes when checking the database status.

### **-p** *productShortName*

Specifies the DB2 database product to be installed. This parameter is case insensitive and is mandatory when the **-n** parameter is specified. The product short name (*productShortName*) can be found in the file *ComponentList.htm* (under the product full name) located in the *db2/plat* subdirectory on your media where *plat* is the platform name that you are installing on. You can only install one product at a time.

### **-c** *NLPACK\_location*

Specifies the absolute path location of the related DB2 National Language Pack (NLPACK). This parameter is mandatory when **-n** is specified. The DB2 NLPACK location needs to be provided explicitly if all of the following conditions are met:

- The **-n** option is specified.
- The installation requires National Language (non-English) support.
- The DB2 NLPACK is neither inside the DB2 database product image nor in the same subdirectory as the DB2 database product image.

### **-n** Specifies noninteractive mode. When specified, you must also specify **-b**, **-p**, and/or **-c**.

### **-m** This option applies to non-root installation only. Specifies upgrade of a non-root copy. During upgrade, all preexisting DB2 database products in the current path are removed. Upgrade installs the specified product. Following the upgrade, other DB2 database products need to be installed separately.

### **-L** *language*

Specifies national language support. You can install a non-English version of a DB2 database 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.

### **-l** *log-file*

Specifies the log file. For root installations, the default log file is

## db2\_install - Install DB2 database product

/tmp/db2\_install.log\$\$, where \$\$ represents the process ID. For non-root installations, the default log file is /tmp/db2\_install\_<userID>.log, where <userID> represents the user ID that owns the non-root installation. If the IBM Tivoli System Automation for Multiplatforms is being installed or updated with the **db2\_install** command, the corresponding log file will be located in the same directory as DB2 database log files.

### -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.

## Example

- To install from an image in /mnt/cdrom, and to be prompted for all needed input, or 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
```

- To install DB2 Enterprise Server Edition with the DB2 pureScale Feature to /db2/newlevel, from an image in /mnt/cdrom, non-interactively in English, issue:

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

## Usage notes

If a IBM PowerHA® SystemMirror for AIX cluster is running, you can not perform a Tivoli SA MP installation, upgrade or update because Tivoli SA MP bundles Reliable Scalable Cluster Technology (RSCT) filesets that are dependent on PowerHA SystemMirror. To skip Tivoli SA MP installation use the **db2\_install** command or the **installFixPack** command. For information on installing or upgrading Tivoli SA MP using a PowerHA SystemMirror cluster, see the white paper entitled "Upgrade guide for DB2 Servers in HACMP™ Environments", which is available from the "IBM Support and downloads" website (<http://www.ibm.com/support/docview.wss?uid=swg21045033>).

## db2\_local\_ps - DB2 process status for Linux/UNIX

On Linux and UNIX systems, all of the DB2 processes running under an instance can be displayed using the `db2_local_ps` command.

### Authorization

None

### Required connection

None

### Command syntax

▶—`db2_local_ps`—▶

### Command parameters

#### `db2_local_ps`

Outputs all of the DB2 processes running under an instance.

### Examples

```
[db2inst1@bower1 ~]$ db2_local_ps
```

```
Node 0
  UID      PID      PPID     C      STIME    TTY      TIME CMD
db2inst1  3254     3253     0      14:04   pts/1    00:00:00 db2sysc 0
  root     3255     3254     0      14:04   pts/1    00:00:00 db2ckpwd 0
  root     3256     3254     0      14:04   pts/1    00:00:00 db2ckpwd 0
  root     3257     3254     0      14:04   pts/1    00:00:00 db2ckpwd 0
  root     3266     3254     0      14:04   pts/1    00:00:00 db2gds 0
db2inst1  3267     3254     0      14:04   pts/1    00:00:00 db2licc 0
db2inst1  3268     3254     0      14:04   pts/1    00:00:00 db2ipccm 0
db2inst1  3269     3254     0      14:04   pts/1    00:00:00 db2tccpm 0
db2inst1  3271     3254     0      14:04   pts/1    00:00:00 db2resync 0
db2inst1  3273     3254     0      14:04   pts/1    00:00:00 db2acd ,0,0,0,1,0,0,897b50,...
db2inst1  3297     3266     0      14:04   pts/1    00:00:00 db2loggr (SAMPLE) 0
db2inst1  3299     3266     0      14:04   pts/1    00:00:00 db2loggw (SAMPLE) 0
db2inst1  3300     3266     0      14:04   pts/1    00:00:00 db2lfr (SAMPLE) 0
db2inst1  3301     3266     0      14:04   pts/1    00:00:00 db2dlock (SAMPLE) 0
db2inst1  3303     3266     0      14:04   pts/1    00:00:00 db2pclnr 0
db2inst1  3313     3266     0      14:05   pts/1    00:00:00 db2pfchr 0
db2inst1  3314     3266     0      14:05   pts/1    00:00:00 db2pfchr 0
db2inst1  3315     3266     0      14:05   pts/1    00:00:00 db2pfchr 0
db2inst1  3316     3266     0      14:05   pts/1    00:00:00 db2stmm (SAMPLE) 0
db2inst1  3317     3266     0      14:05   pts/1    00:00:00 db2taskd (TOOLSDB) 0
db2inst1  3318     3266     0      14:05   pts/1    00:00:00 db2taskd (SAMPLE) 0
db2inst1  3319     3266     0      14:05   pts/1    00:00:00 db2stmm (TOOLSDB) 0
db2inst1  3320     3266     0      14:05   pts/1    00:00:00 db2evmgi (DB2DETAILDEADLOCK) 0
db2inst1  3321     3266     0      14:05   pts/1    00:00:00 db2evmgi (DB2DETAILDEADLOCK) 0
db2inst1  3341     3266     0      14:05   pts/1    00:00:00 db2loggr (TOOLSDB) 0
db2inst1  3343     3266     0      14:05   pts/1    00:00:00 db2loggw (TOOLSDB) 0
db2inst1  3344     3266     0      14:05   pts/1    00:00:00 db2lfr (TOOLSDB) 0
db2inst1  3345     3266     0      14:05   pts/1    00:00:00 db2dlock (TOOLSDB) 0
db2inst1  3346     3266     0      14:05   pts/1    00:00:00 db2pclnr 0
db2inst1  3347     3266     0      14:05   pts/1    00:00:00 db2pfchr 0
db2inst1  3348     3266     0      14:05   pts/1    00:00:00 db2pfchr 0
db2inst1  3349     3266     0      14:05   pts/1    00:00:00 db2pfchr 0
```

## db2\_local\_ps - DB2 process status for Linux/UNIX

```
db2inst1      3270      3268    2    14:04 pts/1 00:00:01 db2agent (TOOLSDB) 0
db2inst1      3285      3268    0    14:04 pts/1 00:00:00 db2agent (SAMPLE) 0
```

Node 1 ...

### Usage notes

Note that processes will not be shown if the instance is stopped. Run the **db2start** command if processes are not listed.

## db2acsutil - Manage DB2 snapshot backup objects

Lists, deletes and monitors DB2 snapshot backup objects.

You can use **db2acsutil** to manage DB2 snapshot backup objects in the following three ways:

- List the DB2 snapshot backups that you can use to restore your database
- Delete DB2 snapshot backups that were generated using the **BACKUP** command, the db2Backup API, or the ADMIN\_CMD stored procedure with the BACKUP DATABASE parameter
- Monitor the status of DB2 snapshot backups

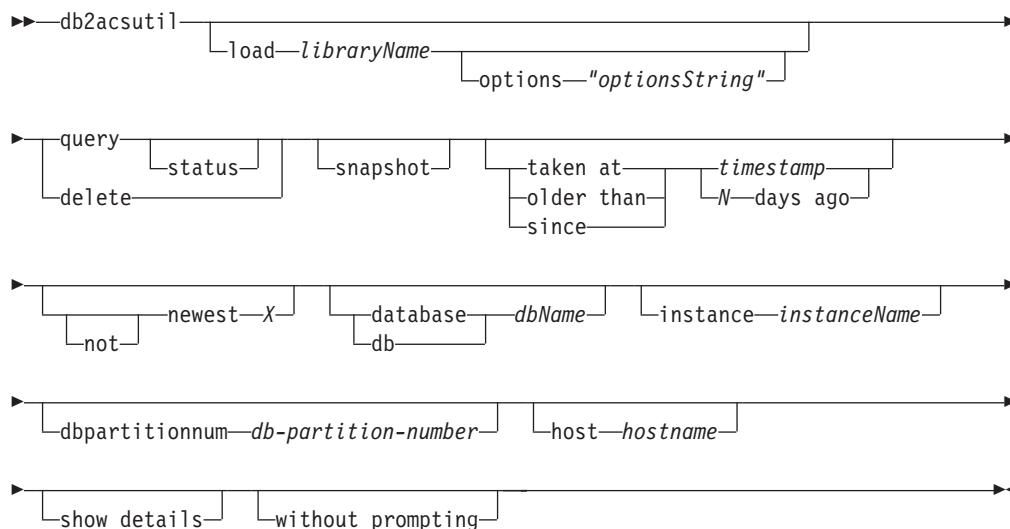
### Authorization

None

### Required connection

None

### Command syntax



### Command parameters

#### load *libraryName*

The name of the shared library containing the vendor fast copying technology used in the DB2 snapshot backup. This parameter can contain the full path. If the full path is not given, the default path is the same library as with the **BACKUP DB** and **RESTORE DB** commands (in ~/sql1lib/acs).

#### options "*optionsString*"

Specifies options to be used for this utility. The string will be passed to the vendor support library exactly as it was entered, without the double quotation marks.

#### query

Queries the ACS repository and returns a table of known objects.

## db2acsutil - Manage DB2 snapshot backup objects

### **status**

Queries the ACS repository and returns a table of known objects with their current status.

### **delete**

This deletes DB2 snapshot objects, and removes their record from the ACS repository once they have been deleted.

### **snapshot**

Filters the records returned or operated on to only snapshot objects.

### **taken at | older than | since**

These options filter the results of the utility to the specified time ranges.

#### *timestamp*

A timestamp of the form *YYYYMMDDhhmmss*.

#### *N days ago*

Number of days ago, where *N* is the number of days before the current date.

### **[not] newest X**

Filter the utility results such that only the newest (by timestamp) *X* records are considered. If the **NOT** keyword is specified, then all records except the newest *X* are considered.

### **database | db *dbName***

Considers only those objects associated with the specified database name.

### **instance *instanceName***

The name of the database manager instance associated with the DB2 snapshot backup objects you are managing.

### **dbpartitionnum *db-partition-number***

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

### **host *hostname***

Considers only those objects created by the specified *hostname*. For example, this would typically be the TCP/IP hostname of the DB2 server.

### **show details**

Displays detailed object information from the ACS repository. If this option is used, instead of a table with a single brief record per line, a detailed stanza will be produced for each ACS object.

### **without prompting**

Specifies that the utility will run unattended, and that any actions which normally require user intervention will return an error message.

## **Examples**

Sample output for a snapshot backup with an active background copy.

```
db2acsutil query status db f01 instance db2inst1 dbpartitionnum 0
```

| Instance | Database | Part | Image          | Time | Status                                                            |
|----------|----------|------|----------------|------|-------------------------------------------------------------------|
| keon14   | F01      | 0    | 20070719120848 |      | Remotely mountable + Background monitor<br>pending (16 / 1024 MB) |

Sample output for a snapshot backup with a completed background copy.

## db2acsutil - Manage DB2 snapshot backup objects

```
db2acsutil query status db f01 instance db2inst1 dbpartitionnum 0 show details
```

```
Instance : keon14
Database : F01
Partition : 0
Image timestamp : 20070719120848
Host : machine1
Owner :
DB2 Version : 9.5.0
Creation time : Thu Jul 19 12:08:50 2007
First active log (chain:file) : 0:0
Metadata bytes : 6196
Progress state : Successful
Usability state : Remotely mountable + Repetitively restorable + Swap restorable
                  + Physical protection + Full copy
Bytes completed : 0
Bytes total : 0
```

### Usage notes

Using **db2acsutil** is the only way to delete DB2 snapshot backups created using the **BACKUP** command, the db2Backup API, or the ADMIN\_CMD stored procedure with the BACKUP DATABASE parameter. You cannot use automated recovery object deletion or the **PRUNE HISTORY** command with the **AND DELETE** parameter to delete DB2 snapshot backups. You also cannot delete backups manually through the filer/storage system.

The usability state of a DB2 snapshot backup indicates what you can do with the DB2 snapshot. Table 1 lists and describes possible DB2 snapshot backup usability states.

Table 48. Usability states returned for DB2 snapshot backups

| Usability state          | Description                                                                                                                                                                          |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LOCALLY_MOUNTABLE        | You can mount the backed up data from the local machine.                                                                                                                             |
| REMOTEY_MOUNTABLE        | You can mount the backed up data from a remote machine.                                                                                                                              |
| REPETITIVELY_RESTORABLE  | You can use the DB2 snapshot backup image multiple times to restore your backed up data.                                                                                             |
| DESTRUCTIVELY_RESTORABLE | You can use the DB2 snapshot backup image to restore your backed up data once; after the backed up data is restored, this DB2 snapshot image, and potentially others, are destroyed. |
| SWAP_RESTORABLE          | You can access the volumes directly, but a <b>RESTORE DB</b> command cannot be executed and the backed up data cannot be copied back onto the source volumes.                        |
| PHYSICAL_PROTECTION      | The snapshot is protected against physical failures in the source volumes.                                                                                                           |
| FULL_COPY                | A full copy of the data has been created. You can use the DB2 snapshot backup image to restore the backed up data.                                                                   |

## db2acsutil - Manage DB2 snapshot backup objects

Table 48. Usability states returned for DB2 snapshot backups (continued)

| Usability state            | Description                                                                                                                                                                                                                                                     |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DELETED                    | Indicates that a backup has been marked for deletion. The snapshot storage associated with a DELETED backup will be withdrawn via a maintenance process running in the background. Once this has completed, the backup will be removed from the ACS repository. |
| FORCED_MOUNT               | Awaiting verification of filesystem consistency by mounting an AIX JFS filesystem.                                                                                                                                                                              |
| BACKGROUND_MONITOR_PENDING | Status is being monitored by the ACS background progress monitor.                                                                                                                                                                                               |
| TAPE_BACKUP_PENDING        | Awaiting an offloaded tape backup.                                                                                                                                                                                                                              |
| TAPE_BACKUP_IN_PROGRESS    | An offloaded tape backup is currently in progress.                                                                                                                                                                                                              |
| TAPE_BACKUP_COMPLETE       | Offloaded tape backup has completed.                                                                                                                                                                                                                            |



---

## db2addicons - Create main menu entries for DB2 tools

Creates main menu entries for DB2 tools.

On Linux operating systems, the **db2addicons** command creates main menu entries for DB2 tools for the current user. The main menu entries for DB2 tools are created by manually running the **db2addicons** command. For the DB2 instance owner, the menu entries are created automatically by instance utilities when the DB2 instance was created, updated or upgraded. If the main menu entries are needed on the desktop of another user, the **db2addicons** command can be run as that specific user, but the instance environment must first be set within that user's environment before running the command.

### Authorization

None

### Command syntax

►—db2addicons -h►

### Command parameters

**-h**      Displays usage information.

## 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.

**Important:** This command has been deprecated and might be removed in a future release because the DB2 administration server (DAS) has been deprecated in Version 9.7. For more information, see “DB2 administration server (DAS) has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html>.

On Linux and UNIX operating 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

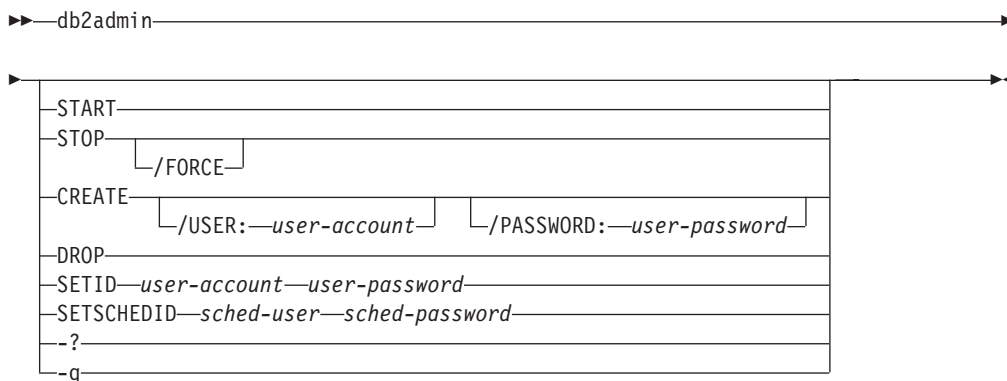
DASADM on UNIX operating systems but not associated with a 64-bit instance.

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.

**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.

## db2adutl - Managing DB2 objects within TSM

Allows users to query, extract, verify, and delete backup images, logs, and load copy images that are 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

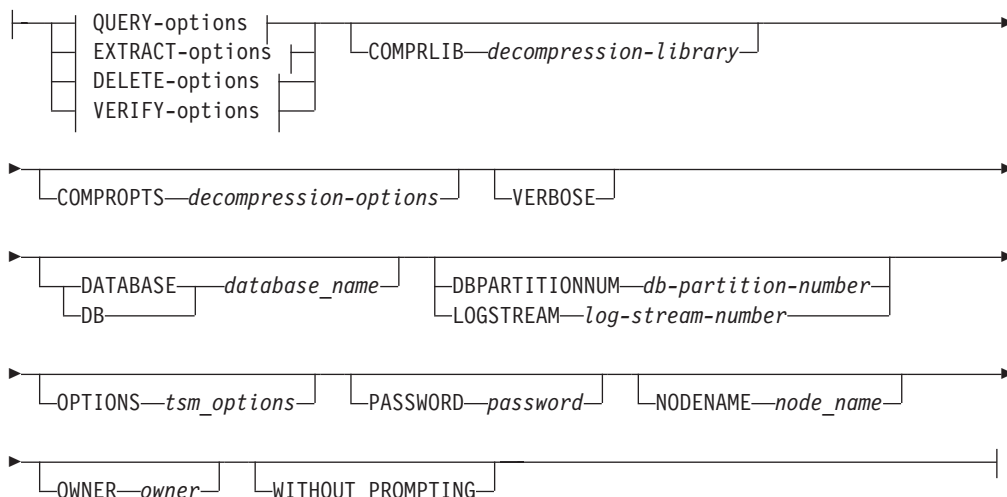
### Required connection

None

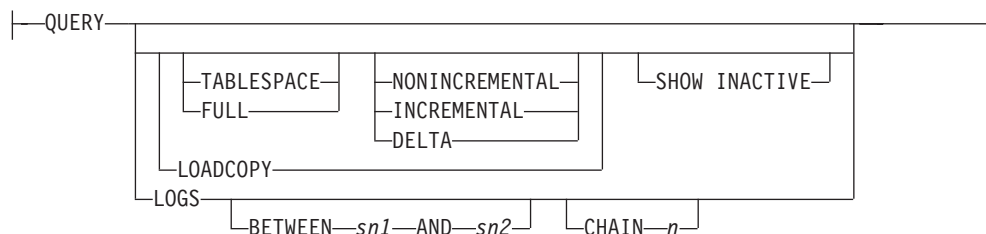
### Command syntax



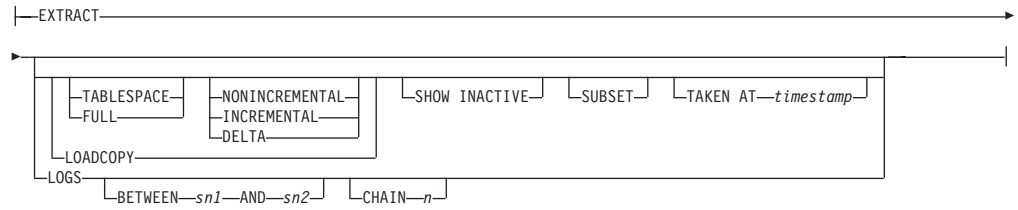
#### db2-object-options:



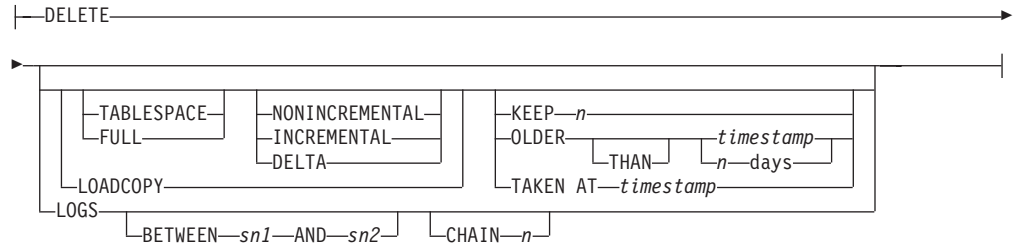
#### QUERY-options:



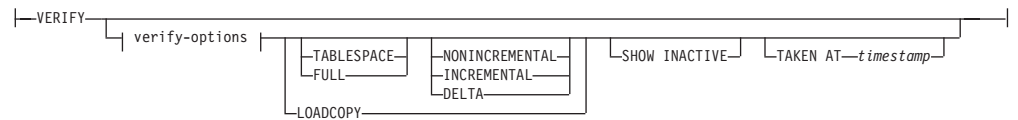
**EXTRACT-options:**



**DELETE-options:**



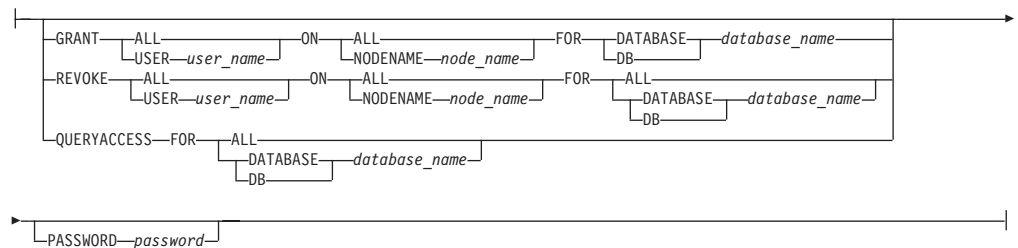
**VERIFY-options:**



**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.

#### **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>
```

**Note:**

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

For DMS table spaces:

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

**Note:**

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.

**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

## db2adutl - Managing DB2 objects within TSM

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.

### **LOGSTREAM** *log-stream-number*

Considers only those objects created by the specified logstream number.

### **OPTIONS** "*tsm\_options*"

Specifies options to be passed to the TSM server during the initialization of the TSM session. **OPTIONS** is passed to the TSM server exactly as it was entered, without the double quotation marks. When you use the **OPTIONS** parameter, the **db2adutl** command returns any errors generated by the TSM server.

### **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.

**Important:** The **NODENAME** parameter and the **OPTIONS** parameter with the **-asnodename** value are not compatible and cannot be used at the same time. You should use the **OPTIONS "-asnodename"** parameter for TSM environments supporting proxy nodes configurations, and use the **NODENAME** parameter for other types of TSM configurations.

### **OWNER** *owner*

Considers only those objects created by the specified owner.

**Important:** The **OWNER** parameter and the **OPTIONS** parameter with the **-asnodename** value are not compatible and cannot be used at the same time. You should use the **OPTIONS "-asnodename"** parameter for TSM environments supporting proxy nodes configurations, and use the **OWNER** parameter for other types of TSM configurations.

### **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 example is sample output from the command `db2 backup database rawsampl use tsm`

Backup successful. The timestamp for this backup is : 20031209184503

The following example 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, Log stream: 0,

    Taken at 2003-12-09-18.46.13

Log file: S0000051.LOG, Chain Num: 0, Log stream: 0,

    Taken at 2003-12-09-18.46.43

Log file: S0000052.LOG, Chain Num: 0, Log stream: 0,

    Taken at 2003-12-09-18.47.12

Log file: S0000053.LOG, Chain Num: 0, Log stream: 0,

    Taken at 2003-12-09-18.50.14

Log file: S0000054.LOG, Chain Num: 0, Log stream: 0,

    Taken at 2003-12-09-18.50.56

Log file: S0000055.LOG, Chain Num: 0, Log stream: 0,

    Taken at 2003-12-09-18.52.39

2. The following example 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 Log stream: 0 Sessions: 1

## db2adutl - Managing DB2 objects within TSM

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 example 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

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, Log stream: 0,  
Taken at 2003-12-09-18.46.13  
Log file: S0000051.LOG, Chain Num: 0, Log stream: 0,  
Taken at 2003-12-09-18.46.43  
Log file: S0000052.LOG, Chain Num: 0, Log stream: 0,  
Taken at 2003-12-09-18.47.12  
Log file: S0000053.LOG, Chain Num: 0, Log stream: 0,  
Taken at 2003-12-09-18.50.14  
Log file: S0000054.LOG, Chain Num: 0, Log stream: 0,  
Taken at 2003-12-09-18.50.56  
Log file: S0000055.LOG, Chain Num: 0, Log stream: 0,  
Taken at 2003-12-09-18.52.39

3. The following example 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

4. The following example is sample output that is displayed from a backup image of 3 members in a DB2 pureScale environment.

## db2adutl - Managing DB2 objects within TSM

```
BufAddr  MemberNum PoolID Token Type Offset FileSize ObjectSize OrigSize Object Name
-----
00000000:      0      0      0  19      0      268      268      0 "BACKUP.START.RECORD.MARKER"

numTbspsInDB : 3
numTbspsInImg : 3

Total members : 3
Member numbers: 0,1,2
```

### Usage notes

One parameter from each of the following groups 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.

When using proxy nodes in TSM environments, in order to see the backup images or the log archives taken when the proxy node was used, you must specify the **OPTIONS** parameter with the shared TSM proxy node value using the `asnodename` option (for example `OPTIONS "-asnodename=cluster1"`). The **OPTIONS** parameter is available starting in Version 9.8 Fix Pack 3 and later fix packs.

TSM grants delete access to the owner of the object or to the root user. It might restrict delete access to other users.

Each log file name has the following format:

```
S0*****.LOG
```

Before Version 9.8 Fix Pack 3, the log files on the TSM server were written to the `./NODE0***/TESTLOG/C0*****/` directory. In Version 9.8 Fix Pack 3 and later fix packs, the log files on the TSM server are written to the `./NODE****/LOGSTREAM****/C0*****/` directory.

## db2advls - 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.

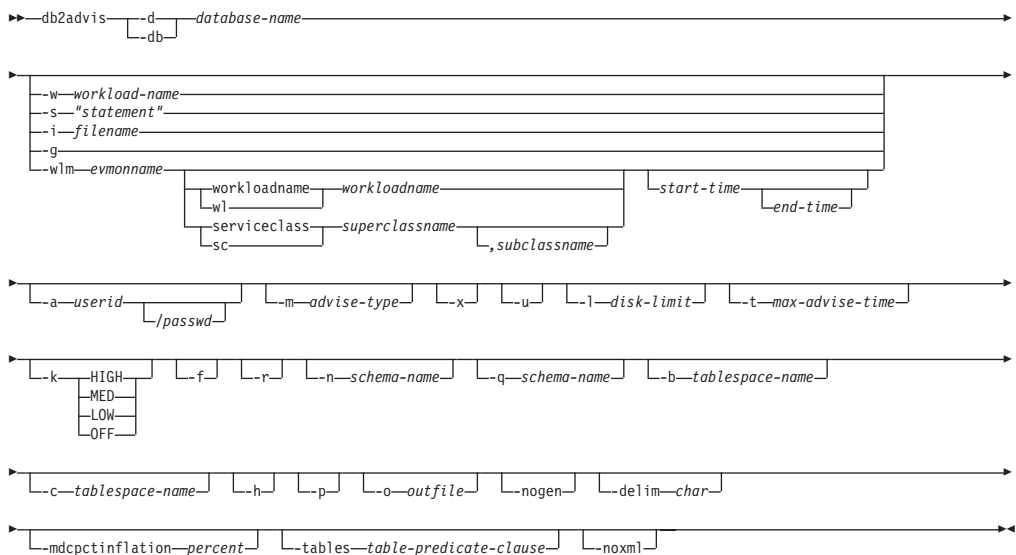
### Authorization

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

**-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.

**-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.

**-d** *database-name*

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

**-delim** *char*

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

**-f** Drops previously existing simulated catalog tables.

**-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**, or **-w** options.

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

**-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**, or **-w** options.

**-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 and every query is considered. The default is MED.

**-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.

### **-m** *advise-type*

Specifies the type of recommendation the advisor will return. Any combination of I, M, C, and P (in upper- or lowercase) can be specified. For example, **db2advls -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.

### **-mdcpctinflation** *percent*

Specifies the maximum percentage that the table disk size can increase in an MDC recommendation. For example, it indicates that a table is allowed to increase to  $1 + \text{percent}/100$  times its original size when it is converted to a MDC table. *percent* is a floating point number with a default value of 10.

### **-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.

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

**-noxml** Indicates that the detailed XML output following the recommendation text is not to be written to the console.

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

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

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

### **-q** *schema-name*

Specifies the qualifying name of unqualified names in the workload. It serves as the schema name to use for CURRENT SCHEMA when **db2advls** executes. The default schema name is the user ID of the person executing the command.

**-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 **db2advls** 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.

**-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**, or **-w** options.

**-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.

**-tables** *table-predicate-clause*

Indicates that only a subset of all existing tables should be considered. The *table-predicate-clause* must be a predicate that can be used in the WHERE clause of a query on SYSCAT.TABLES. The tables considered by **db2advis** will be the intersection of the tables from this query and the tables in the workload.

This command parameter does not apply to recommendations about new MQTs.

**-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.

**-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**, or **-s** options.

**-wlm** *evmonname*

Specifies to get the table names corresponding to the ACTIVITY and ACTIVITYSTMT logical data groups from SYSCAT.EVENTTABLES for event name *evmonname*, and joins them together on ACTIVATE\_TIMESTAMP, ACTIVITY\_ID and ACTIVITY\_SECONDARY\_ID for records that have PARTIAL\_RECORD = 0 (completed transactions). An optional *start-time* and *end-time* timestamp can be added to get statements on or after the *start-time* and, optionally, on or before the *end-time*. *start-time* and *end-time* are with respect to the TIME\_COMPLETED column from the ACTIVITY tables.

**workloadname | wl** *workloadname*

Specifies the *workloadname* that is searched for in SYSCAT.WORKLOADS. The ACTIVITY event monitor table is joined with SYSCAT.WORKLOADS on the workload id to obtain these statements.

**serviceclass | sc** *superclassname*

Specifies the service class information which comes from SYSCAT.SERVICECLASSES. When no subclass is given, all statements for a service superclass is retrieved, which is basically the PARENTSERVICECLASS in SYSCAT.SERVICECLASSES. The ACTIVITY event monitor table is joined with SYSCAT.SERVICECLASSES on the service class id to obtain these statements.

*,subclassname*

Specifies the *subclassname* if a *superclassname* is specified; separated by a comma. This parameter is optional.

*start-time*

Specifies the start timestamp.

*end-time*

Specifies the end timestamp. This parameter is optional.

- x Specifies that the password will be read from the terminal or through user input.

## 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 53 MB 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 -i 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 -1 -1 -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 -1 -1 -m IMCP -b space1 -c space2 -k
  HIGH -q schema1 -n schema2 -r
```

5. In the following example, the utility connects to database SAMPLE, and recommends MDC for tables for EMPLOYEE and DEPT where MDC candidates are allowed to grow by 30.5% of their original size.



```
db2advis -d sample -type C -disklimit 100 -i db2advis.in
-tables "TABNAME IN ('EMPLOYEE','DEPT')" -mdcpctinflation 30.5
```

## 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 statement is restricted.

Starting with Version 9.7, the Design Advisor will not recommend partitioned indexes. All indexes will be recommended with the NOT PARTITIONED clause. With this recommendation, it is your choice whether to use PARTITIONED (the default) or NOT PARTITIONED to create indexes based on their application scenarios and on the benefit that partitioned index can provide.

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

1. Issue the command:

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

Wait for an appropriate interval of time.

2. Issue the command:

```
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 file.

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 **db2advis** command results in an error saying "Cannot insert into DB2ADVISE\_INSTANCE", you will need to bind db2advis.bnd and run the **db2advis** command with the **-l** option. The bind operation can be performed by issuing the command:

```
db2 bind db2advis.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 **db2advis** process running, the advisor has terminated prematurely. If STATUS has an "EX", you are also shown an "SQLCODE" to determine how the advisor failed.

## db2advise - DB2 Design Advisor

If the `-1 disk-limit` option is not specified, you must have at least one of SYSADM, SYSCTRL, SYSMANT, or SYSMON authority to determine the maximum database size using the GET\_DBSIZE\_INFO stored procedure.

The *table-predicate-clause* in the `-tables` parameter is used to query SYSCAT.TABLES and determine the tables that the advisor will consider. Only base tables or existing MQTs can be considered, but aliases and logical views can be used in the *table-predicate-clause* to return the list of base table names or MQTs. For example, to specify the subset of tables that have views that start with 'TV', specify `-tables "(tablename, tabschema) in (SELECT bname, bschema FROM SYSCAT.TABDEP WHERE TABNAME LIKE 'TV%')"`.

As of Version 9.7, the query optimizer measures the cost of the I/O savings and the cost of decompressing key values and RIDs in the cost model. As such, the Index advisor is capable of estimating the compressed index size.

---

## db2audit - Audit facility administrator tool

DB2 database systems provide 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 audit log files. 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 both the instance and database levels, independently recording all activities in separate logs based on either the instance or the database.

DB2 database systems provide the ability to independently audit at both the instance and at the individual database level. The **db2audit** tool is used to configure audit at the instance level as well as control when such audit information is collected. The AUDIT SQL statement is used to configure and control the audit requirements for an individual database. The **db2audit** tool can be used to archive both instance and database audit logs as well as to extract from archived logs of either type.

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 about the coordinator partition and originating database partition identifiers.

The instance audit log (`db2audit.instance.log.node_number[.timestamp]`) is located in the instance's `security/auditdata` subdirectory, and the audit configuration file (`db2audit.cfg`) is located in the instance's `security` subdirectory. The database audit log is named `db2audit.db.dbname.log.node_number[.timestamp]`. 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. This does not include database level activities.
- Stop recording auditable events within the DB2 instance.
- Configure the behavior of the audit facility at the instance level only.
- Select the categories of the auditable events to be recorded at the instance level only.
- Request a description of the current audit configuration for the instance.
- Flush any pending audit records from the instance and write them to the audit log.
- Archive audit records from the current audit log for either the instance or a database under the instance.
- Extract audit records from an archived audit log by formatting and copying them to a flat file or ASCII delimited file. Extraction is done in preparation for analysis of log records.

# db2audit - Audit facility administrator tool

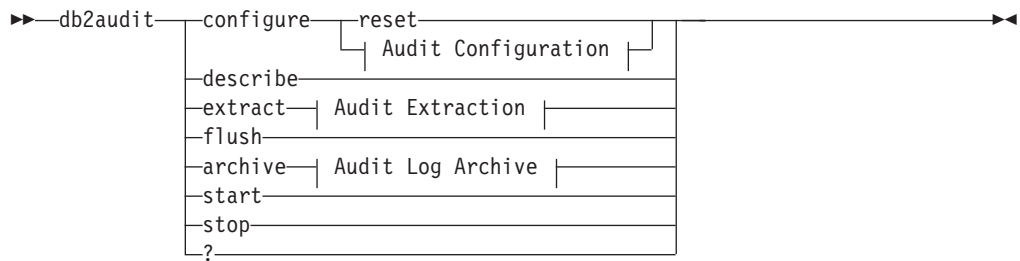
## Authorization

SYSADM

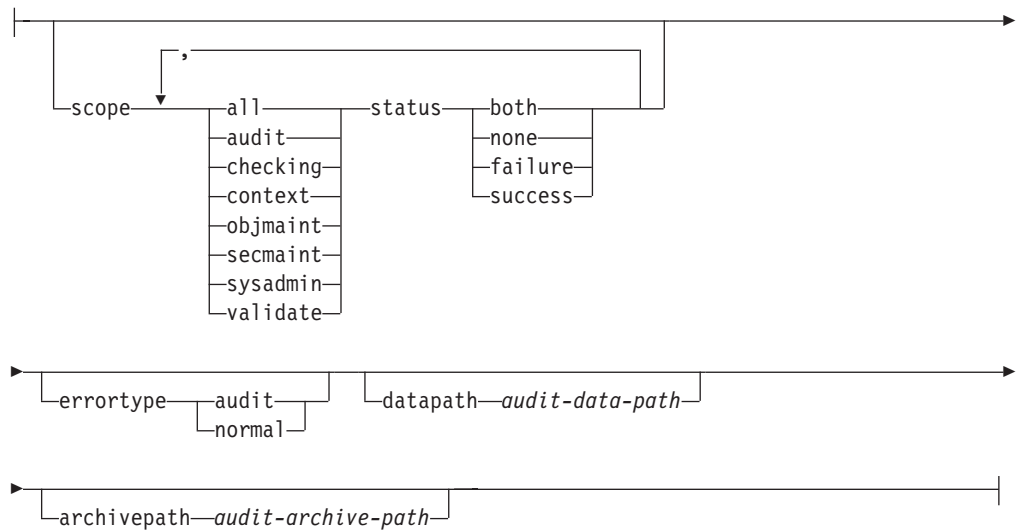
## Required Connection

None

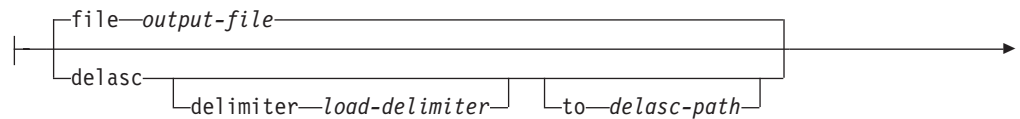
## Command syntax

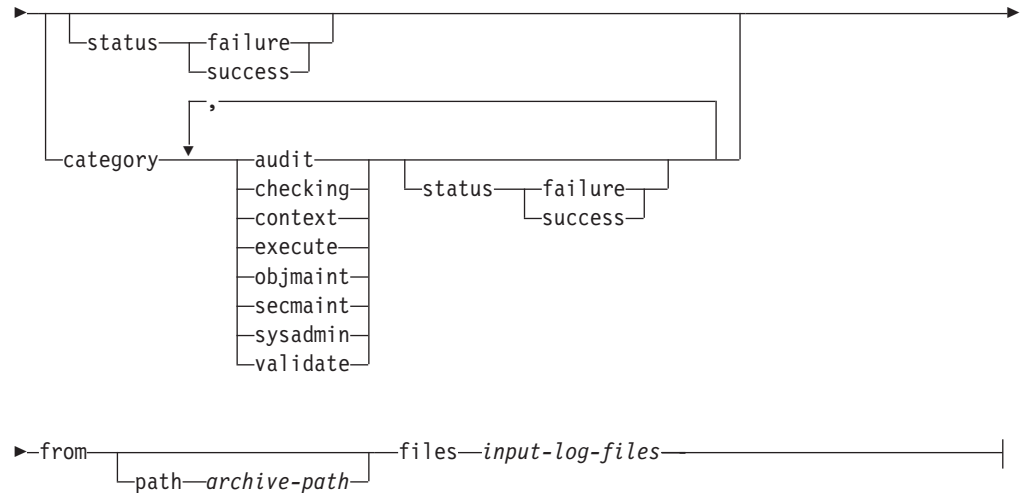


## Audit Configuration:

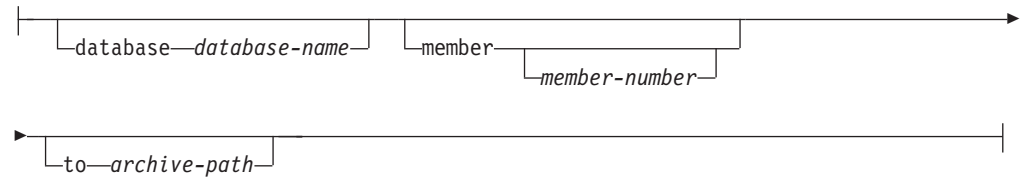


## Audit Extraction:





**Audit Log Archive:**



**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 stopped. Updates, occurring when the instance is active, dynamically affect the auditing being done by the DB2 instance. 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. All configure options, except the data path and archive path, only apply to instance level audit events, and not to database level audit events. The path options apply to the instance and all databases within the instance.

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** for each category is failure, **errortype** is normal, and the auditing of instance level events is off). This action will create a new audit configuration file if the original has been lost or damaged. The audit data path and archive path will be blank. This option does not reset any of the audit policies or use of those policies at the database level.
- scope** This action specifies which categories will be audited, and the status of each of those categories.
- status** This action specifies whether only successful or failing

## db2audit - Audit facility administrator tool

events, or both successful and failing events, should be logged. **status** has the following options:

**both** Successful and failing events will be audited.

**none** No events for this category will be audited.

**failure**  
Only failing events will be audited.

**success**  
Only successful events will be audited.

Only the categories specified on the configure statement will be modified. All other categories will have their status preserved.

### Note:

- 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. It is recommended that the number and type of events being logged be limited as much as possible, otherwise the size of the audit log will grow rapidly. This action also allows a particular focus for auditing and reduces the growth of the log.
- **context** events occur before the status of an operation is known. Therefore, such events are logged regardless of the value associated with this parameter, unless the **status** is none.
- If the same category is repeated, or categories are also specified with the **all** keyword, a syntax error will be returned.

### 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.

### datapath *audit-data-path*

This is the directory to which the audit logs produced by the DB2 database system will be written. The default is `sqllib/security/auditdata` (*instance path\instance\security\auditdata* on Windows). This parameter affects all auditing within an instance, including database level auditing. This must be a fully qualified path and not a relative path. The instance owner must have write permission on this directory. On Windows, the user issuing a local instance command, for example, **db2start**, **db2audit**, and **db2 update dbm cfg**, must also have write permission on this directory if the command is required to be audited. On a partitioned database environment, this directory does not need to be an NFS shared directory, although that is possible. A non-shared directory will result in increased performance as each member is

writing to a unique disk. The maximum length of the path is 971 bytes for UNIX or Linux and 208 bytes for Windows operating systems.

If the path is provided as "", then the path will be updated to be the default. **db2audit describe** will show no path as being set and the default path will be used. Note, to prevent the shell from interpreting the quotation marks, they will generally need to be escaped, for example

```
db2audit configure datapath "\\\"
```

The data path must exist. In a partitioned database environment, the same data path will be used on each member. There is no way to specify a unique set of data paths for a particular member unless database partition expressions are used as part of the data path name. Doing this allows the member number to be reflected in the storage path such that the resulting path name is different on each member.

### **archivepath** *audit-archive-path*

This is the default directory for the archive and extract options. In a partitioned database environment, it is recommended that this directory be an NFS shared directory accessible by all members. The default is sqllib/security/auditdata (sqllib\instance\security\auditdata on Windows). This must be a fully qualified path and not a relative path. The instance owner must have write permission on this directory. The maximum length of the path is 971 bytes for UNIX or Linux and 208 bytes for Windows operating systems.

The archive path must exist, and database partition expressions are NOT allowed for the archive path.

### **describe**

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

The following items are displayed:

- If audit is active.
- The status for each category.
- The error type in the form of whether or not an SQLCA is returned on errors.
- The data and archive paths.

This is an example of what the **describe** output looks like:

```
DB2 AUDIT SETTINGS:
```

```
Audit active: "FALSE "  
Log audit events: "SUCCESS"  
Log checking events: "FAILURE"  
Log object maintenance events: "BOTH"  
Log security maintenance events: "BOTH "  
Log system administrator events: "NONE"  
Log validate events: "FAILURE"  
Log context events: "NONE"  
Return SQLCA on audit error: "TRUE "  
Audit Data Path: "/auditdata"  
Audit Archive Path: "/auditarchive"
```

```
AUD0000I  Operation succeeded.
```

## db2audit - Audit facility administrator tool

### extract

This parameter allows the movement of audit records from the audit log to an indicated destination. The audit log will be created in the database code page. All of the fields will be converted to the current application code page when extract is run.

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

#### **file** *output-file*

The extracted audit records are placed in *output-file*. If the directory is not specified, *output-file* is written to the current working directory. If the file already exists the output will be appended to it. If a file name is not specified, records are written to the db2audit.out file in the archive path specified in the audit configuration file.

#### **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. In addition, the file auditlobs will also be created to hold any lobes that are included in the audit data. The filenames are:

- audit.del
- checking.del
- objmaint.del
- secmaint.del
- sysadmin.del
- validate.del
- context.del
- execute.del
- auditlobs

If the files already exist the output will be appended to them. The auditlobs file will be created if the **context** or **execute** categories are extracted. LOB Location Specifiers are included in the .del files to reference the LOBS in the auditlobs file.

#### **delimiter** *load-delimiter*

Allows you to override the default audit character string delimiter, which is the double quote ("), when extracting from the audit log. You would use **delimiter** followed by the new delimiter that you want to use in preparation for loading into a table that will hold the audit records. The new load delimiter can be either a single character (such as !) or a four-character string representing a hexadecimal number (such as 0xff).

#### **to** *delasc-path*

Allows you to specify the path to which the delimited files are written. If it is not specified, then the files are written to the directory indicated by the audit archive path option specified in the audit configuration file.

#### **category**

The audit records for the specified categories of audit events are to be extracted. If not specified, all categories 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.



**path**

The path to the location of the archived audit logs. If this is not specified, the archive path in the audit configuration will be used. The path is not used if the filename contains a fully qualified path.

**files**

The list of audit log files that will be extracted. This may be a single file or a list of files. These files are not altered during an extract. The filenames will be combined with **path** to get the fully qualified filenames if they are not already fully qualified. The list may include standard shell wild cards to specify multiple files.

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

**archive**

This parameter moves the current audit log for either an individual database or the instance to a new location for archiving and later extraction. The current timestamp will be appended to the filename. All records that are currently being written to the audit log will complete before the log is archived to ensure full records are not split apart. All records that are created while the archive is in progress will be written to the current audit log, and not the archived log, once the archive has finished.

The following are the options that can be used when archiving:

**database** *database-name*

The name of the database for which you would like to archive the audit log. If the database name is not supplied, then the instance level audit log is archived.

**member**

Indicates that the archive command is to only be run on the current member, and that the **node\_number** monitor element will indicate what the current member is.

**Note:** The use of current *member-number* is optional in a DB2 pureScale environment and in a partitioned database environment. If `db2audit archive .... node` command is passed and if **DB2NODE** is set, the node value will be used. If **DB2NODE** is not set, 0 will be used.

*member-number*

Informs the **db2audit** executable about which member it is currently running on.

**Note:** The use of current *member-number* is optional in a DB2 pureScale environment and in a partitioned database environment. If `db2audit archive .... node X` command is passed, regardless of whether **DB2NODE** is set or not, the node value (X) will be used.

**to** *archive-path*

The directory where the archived audit log should be created. The directory must exist and the instance owner must have create permission on this directory. If this is not provided, the archive path in the audit configuration will be used.

The format of the filename that is created is:

## db2audit - Audit facility administrator tool

- `db2audit.instance.log.member_number[.YYYYMMDDHHMMSS]` for the instance log
- `db2audit.db.dbname.log.member_number[.YYYYMMDDHHMMSS]` for the database log

where *YYYY* is the year, *MM* is the month, *DD* is the day, *HH* is the hour, *MM* is the minute, and *SS* is the seconds. The time will be the local time. The database name portion will not be present for instance audit logs. The member number in a non-partitioned database environment will be 0. If the file already exists, an append will be performed.

The timestamp will not reflect the last record in the log with 100% accuracy. The timestamp represents when the archive command was run. Entries that are currently being written to the log file must finish before it can be moved, and these entries may have timestamps that are later than the timestamp given to the filename.

If the **member** option is not specified, then the audit log on all members will be archived. The database server must be started in this case. If the database server has not been started, then archive must be run on each member, and the **member** option must be specified to indicate on which member **archive** is to be run (AUD0029).

The **archive** option will output the result and names of the files from each member that archive was run on.

- start** This parameter causes the audit facility to begin auditing events based on the contents of the `db2audit.cfg` file for the instance only. In a partitioned DB2 database instance, auditing will begin for instance and client level activities 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. This has no effect on database level auditing, which is controlled through the AUDIT DDL statement.
- stop** This parameter causes the audit facility to stop auditing events for the instance only. In a partitioned DB2 database instance, auditing will be stopped for instance and client level activities 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. This has no effect on database level auditing, which is controlled through the AUDIT DDL statement.
- ?** This parameter displays the help information for the **db2audit** command.

### Examples

This is a typical example of how to archive and extract a delimited ASCII file in a partitioned database environment. The UNIX remove (**rm**) command deletes the old delimited ASCII files.

```
rm /auditdelasc/*.del
db2audit flush
db2audit archive database mydb to /auditarchive
```

(files will be indicated for use in next step)

```
db2audit extract delasc to /auditdelasc from files /auditarchive
/db2audit.db.mydb.log.*.20070514102856
```

Load the .del files into a DB2 table.

## Usage notes

- Database level auditing is controlled with the **AUDIT** statement.
- The instance level 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 following description of the categories of events available for auditing, 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 an instance 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.
  - You can audit failures, successes, both or none.
- Any operation on the instance 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.
- To clean up and/or view audit logs, run **archive** on a regular basis, then run **extract** on the archived file to save what is useful. The audit logs can then be deleted with standard file system delete commands.

## 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 optional parameters of the tool, 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.

To use the `-o p` option, which specifies the level of performance information, or to use the `-o e` option, which sets the explain mode, you require SYSMON authority.

### Required connection

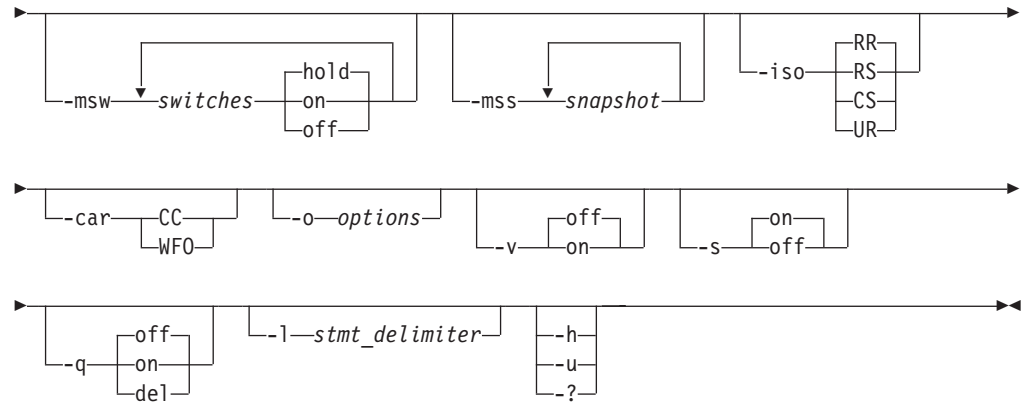
None. This command establishes a database connection.

### Command syntax

```

▶▶ db2batch [-d dbname] [-f file_name] [-a userid/passwd]
           [-m parameters_file] [-t delcol]
           [-r result_file [,summary_file]]
           [-z output_file [,summary_file]] [-c [on|off]]
           [-i [short|long|complete]] [-g [on|off]] [-w 32768|col_width]
           [-time [on|off]] [-cli [cache-size]]

```



## 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 **DB2DBFT** 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 quotation marks 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 following section on **-m** option 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:

## db2batch - Benchmark tool

- 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 quotation marks (') represents a binary string encoded as pairs of hex digits, 'g', 'G', 'n', or 'N' prefixed text enclosed in single quotation marks (') 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' >= '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 numeric parameter values (integers and decimal numbers).

**-t** *delcol*

Specifies a single character column separator. Specify -t TAB for a tab

## db2batch - Benchmark tool

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 [summary_file]`

Specifies an output file that will contain the query results. The default is standard output. Error messages are returned in the standard error. If the optional *summary\_file* is specified, it will contain the summary table.

`-z output_file [summary_file]`

Specifies an output file that will contain the query results and any error messages returned. The default is standard output. Error messages are also returned in the standard error. If the optional *summary\_file* is specified, it will contain the summary table. This option is available starting in Version 9.7 Fix Pack 1.

`-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 the additional processing time 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 2 G. 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 options: *uow*, *statement*, *table*, *bufferpool*, *lock*, *sort*, and *timestamp*. The special switch *all* sets all of the switches mentioned previously. For each switch that you specify, you must choose one of the following values:



**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, dcs\_applinfo\_all, db2, dbase, dbase\_all, dcs\_dbase, dcs\_dbase\_all, dbase\_remote, dbase\_remote\_all, agent\_id, dbase\_appls, appl\_all, dcs\_appl\_all, dcs\_appl\_handle, dcs\_dbase\_appls, dbase\_appls\_remote, appl\_remote\_all, dbase\_tables, appl\_locks\_agent\_id, dbase\_locks, dbase\_tablespace, 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 favor 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).

**-car** Specifies the concurrent access resolution to use for the **db2batch** operation. The **-car** parameter requires a properly configured database server and the isolation level parameter **-iso** set to CS.

**CC** Specifies that the **db2batch** operation should use the currently committed version of the data for applicable scans when it is in the process of being updated or deleted. Rows in the process of being inserted can be skipped. This option applies when the isolation level in effect is Cursor Stability or Read Stability (for Read Stability it skips uncommitted inserts only) and is ignored otherwise. Applicable scans include read-only scans that can be part of a read-only statement as well as a non read-only statement.

**WFO** Specifies that the **db2batch** operation should wait for the outcome of an operation. For Cursor Stability and higher scans, **db2batch** will wait for the commit or rollback when encountering data in the process of being updated or deleted. Rows in the process of being inserted are not skipped.

**-o options**

Control options. Valid options are:

## db2batch - Benchmark tool

### **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).

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 before 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 *stmt\_delimiter*** Specifies the termination character (statement 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

1. The following example is sample output from the command `db2batch -d`

```

crystl -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, -0.0000000000000005, 'demo');

* Elapsed Time is:      0.002926 seconds
-----
* SQL Statement Number 3:

insert into demo values (9223372036854775807, 0.0000000000000005, 'demodemo');

* Elapsed Time is:      0.005676 seconds
-----
* SQL Statement Number 4:

select * from demo;

C1                C2                C3
-----
-9223372036854775808 -5.000000000000000E-015 demo

```

## db2batch - Benchmark tool

```
9223372036854775807 +5.000000000000000E-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
```

### 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 data types: `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**).

- If you specify **-r** and **-z** option together, the **-r** option is ignored as the **-z** option includes what the **-r** specifies.

## 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.

## db2caem - Capture activity event monitor data tool

The **db2caem** tool automates the procedure of creating an activity event monitor.

Run the **db2caem** command to create the activity event monitor to capture data for an SQL statement. This data can be collected with the **db2support** command. The information collected and generated by the **db2caem** tool includes:

- Detailed activity information captured by an activity event monitor including monitor metrics, for example `total_cpu_time` for statement execution
- Formatted EXPLAIN output, including section actuals (statistics for different operators in the access plan).

The **db2caem** tool uses an activity event monitor to capture information about the statements and then extracts and formats the information.

The **db2caem** tool automates the process for creating an activity event monitor,

1. Enabling capture for the statements of interest
2. Invoking the statements (each statement is rolled back after being executed to prevent side effects in the database)
3. Formatting the output information (including exporting activity information for the statements of interest and generation of formatted explain output from the captured section and section actuals).

### Authorization

1. To create activity event monitor, the privileges must include one of the following authorities:

- DBADM authority
- SQLADM authority
- WLMADM authority

and also EXECUTE privilege on the `WLM_SET_CONN_ENV` procedure.

2. If there is not a need to create activity event monitor, the following privileges and authority are required:

- EXECUTE privilege on the `EXPLAIN_FROM_ACTIVITY` procedure
- INSERT privilege on the Explain tables in the specified schema
- SELECT privilege on the event monitor tables for the source activity event monitor

and also one of the following authorities:

- DATAACCESS authority on the activity event monitor tables
- CONTROL or SELECT privilege on the activity event monitor tables

### Required connection

None

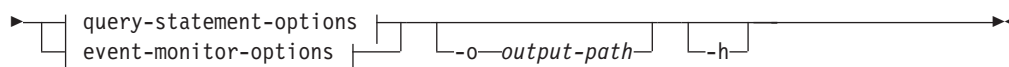
### Command syntax

```

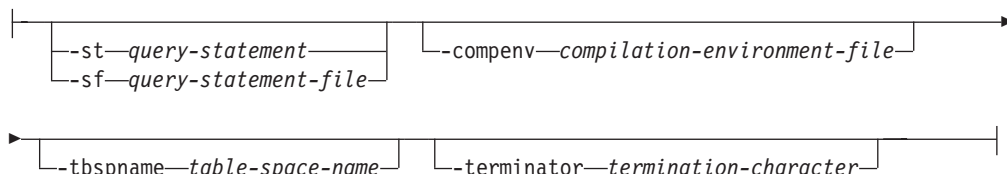
▶▶ db2caem [-d database-name] [-u userid -p password]

```

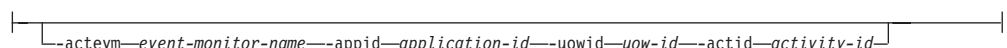
## db2caem - Capture activity event monitor data tool



### query-statement-options:



### event-monitor-options:



## Command parameters

- d** *database-name*  
Specifies the name of the database to be connected to.
- u** *userid*  
Specifies the user ID when connecting to the database.
- p** *password*  
Specifies the password for the user ID when connecting to the database.
- o** *output-path*  
The output files of db2caem will be written to the path that you specify.
- h** Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.

## query-statement-options

- st** *query-statement*  
Specifies the SQL statement for which activity event monitor data is to be captured.  
  
**Note:** The SQL statement will be executed against the specified database by the tool.
- sf** *query-statement-file*  
Specifies the file path containing the SQL statement for which activity event monitor data is being captured. Use the **-terminator** option to specify the character that marks the end of the SQL statement.  
  
**Note:** The SQL statement will be executed against the specified database by the tool.
- compenv** *compilation-environment-file*  
Specifies that the compilation environment will be used when the SQL statement is executed. The compilation environment (comp\_env\_desc) is in BLOB data type and specified through a file as an optional input. If the option is not provided, the default compilation environment will be used when executing the SQL statement.



## db2caem - Capture activity event monitor data tool

### **-tbspname** *table-space-name*

Specifies the table space name for which the activity event monitor will be created in. For a partitioned database environment, the table space should exist on all the database partitions where the SQL statement of interest will be run. If the option is not provided, the default table space will be used when there is a need to create the activity event monitor.

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

Specifies the character that indicates the end of the SQL statement in the **-sf** SQL file if there are multiple statements in the file. The default is a semicolon.

## event-monitor-options

The following options uniquely identify an SQL statement that has already been captured by an existing activity event monitor. They are specified together to identify the statement for which **activity data** and **explain output** should be extracted.

**Note:** Formatted explain output will only be gathered if the section for the statement was captured, and the formatted explain output will only include section actuals if section actuals had been captured for the statement.

### **-actevm** *event-monitor-name*

Specifies the name of the existing activities event monitor containing the data for the statement of interest.

### **-appid** *application-id*

Specifies the application identifier (appl\_id monitor element) uniquely identifying the application that issued the statement of interest.

### **-uowid** *uow-id*

Specifies the unit of work ID (uow\_id monitor element) in which the statement of interest was executed.

### **-actid** *activity-id*

Specifies the activity ID (activity\_id monitor element) of the statement of interest.

## Examples

The following examples show how you can use the **db2caem** tool to create the activity event monitor to capture data for an SQL statement:

- `db2caem -d sample -st "select * from staff"`

Creates the activity event monitor and capture information of details, section and values, as well as actuals for the SQL statement "select \* from staff".

- `db2caem -d sample -sf badquery.sql -terminator $`

Creates the activity event monitor and capture information of details, section and values, as well as actuals for the SQL statement specified in the file badquery.sql.

- `db2caem -d sample -actevm mymon -appid *LOCAL.mikita.100203234904 -uowid 44 -actid 1`

Captures the activity event monitor information of details, section and values, as well as actuals for the SQL statement identified by the event monitor options from the existing activity event monitor. The **db2caem** tool will not create activity event monitor in this example.

## db2caem - Capture activity event monitor data tool

### Usage notes

The **db2caem** tool is used to create the activity event monitor for capturing data which can be collected with the **db2support** command. DB2CAEM\_<timestamp> directory will be generated to contain all the information captured by the **db2caem** tool.

## db2cap - CLI/ODBC static package binding tool

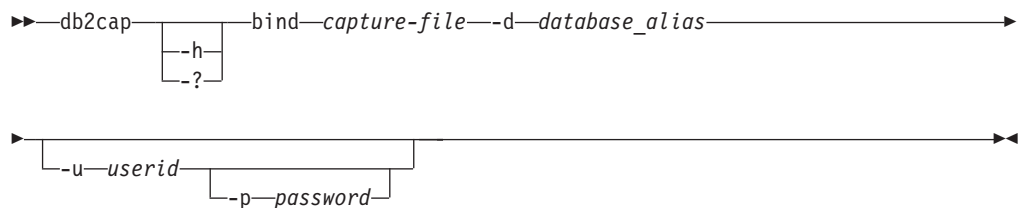
Binds a capture file to generate one or more static packages.

A *capture file* is generated during a static profiling session of a CLI, ODBC, or .NET 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, or .NET 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. This capture file is also known as the pureQueryXML file for .NET.

**-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 operating systems, but can be entered in either lowercase or uppercase on Windows operating systems. Static package binding for .NET applications is supported only 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.

## db2cap - CLI/ODBC static package binding tool

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

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.

## 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

```

▶▶ db2cat [ -d dbname ] [ -h ] [ -l ] [ -n name ] [ -o outfile ]
▶ [ -p descriptor ] [ -vi versionID ] [ -s schema ] [ -t ] [ -v ]
▶ [ -x ] [ -cb ]

```

### 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 values:

##### **check**

Display table check constraints packed descriptor.

##### **controls**

Displays packed descriptors for all enabled permissions and masks for a given table.

##### **rel**

Display referential integrity constraint packed descriptor.

##### **table**

Display table packed descriptor. This includes the inline length if at least one exists for the table.

## db2cat - System catalog analysis

### **summary**

Display summary table packed descriptor.

### **syscontrols**

Displays the packed descriptor for a given mask or permission.

### **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.

### **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.

### **-v**

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

### **-x**

Validates table space extent size in catalogs (does not require a table name).

### **-cb**

Cleans orphan rows from SYSCAT.BUFFERPOOLDBPARTITIONS (does not require a table name).

## **Examples**

### **Example 1**

The following command prints the packed descriptor information for the table `splt_tb1` with schema `raguk` from database `testdb` to terminal:

```
db2cat -d testdb -s raguk -t -n splt_tb1
```

### Usage notes

- The options **-d**, **-s** and **-n** are mandatory when executing the **db2cat** command.
- 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.

---

# 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 (that is, 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 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 authorities:

- 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 (that is, 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.



### 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.

### Note:

The **db2cfexp** command will not export the File data source location information from a client.

If you use the default location, no action is required. If you change the default location on a client, you will need to manually copy this location when exporting connectivity information.

To copy the File data source location from one client to another:

1. On the client you are exporting connectivity information, locate the `%DB2PATH%\TOOLS` directory.
2. Copy the `CA.properties` file.
3. On the client you are importing connectivity information, locate the `%DB2PATH%\TOOLS` directory.
4. Overwrite the existing `CA.properties` file with the copy taken from the originating client.

You have duplicated the File data source location from one client to another.

### 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 (that is, client instance to client instance) that was configured previously. It is especially useful 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 authorities:

- SYSADM
- SYSCTRL

#### Note:

- The root ID should not be used to run the tool.
- If a valid ID is used to run the tool, the ID must have the correct permission for the configuration profile to be imported.

#### 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 **db2cfexp** command.

## 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.

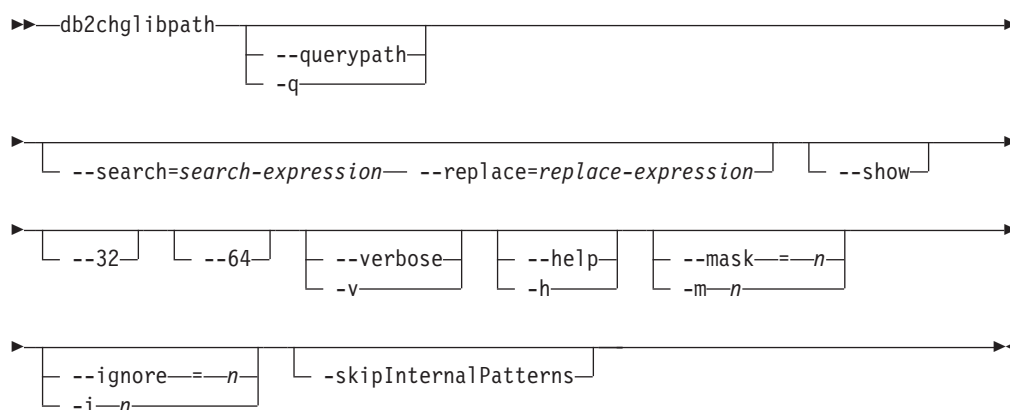
### Prerequisites

- 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 | -s=search-expression

Specifies the expression to be searched for.

## db2chglibpath - Modify the embedded runtime library search path

- replace** | **-r**=*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 files.
- 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.
- mask** | **-m**  
Suppresses error messages for exit values and can only be specified once. Exit values for the mask options are shown under the ignore option.
- ignore** | **-i**  
Suppresses a specific error message.

Exit values for mask and ignore options are:

- 0 - path successfully changed
- 1 - not all specified search and replace on the operations succeeded.
- 2 - file was of the right type but does not have a libpath
- 3 - file was not of the right type to have a libpath
- >3 - other errors

### **--skipInternalPatterns**

The pattern search and replace methods is performed internally to reclaim potential space on the resultant path. Use this option to avoid this replacement.

## 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/sqllib/lib32, issue:

```
db2chglibpath --search=/usr/opt/db2_08_01/lib --replace=/u/usr1/sqllib/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 upgrading applications and routines cannot be used or are unsuccessful.
- 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 before issuing this command.

## db2chgpath - Change embedded runtime path

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

**Note:** If SELinux (Security-enhanced Linux) is enabled after the DB2 database installations on Red Hat Enterprise Linux version 5 (RHEL5), you need to manually run this command for each DB2 database installation of the current release to make the DB2 database system work properly. See the *Usage notes* section for additional information.

### Authorization

Root installations require root user authority. For non-root installations, you must log on with the user ID that owns the non-root installation.

### 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 database 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 database product install location.

### Example

- To check all files under the DB2 database 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`

### Usage notes

On RHEL5 systems, if you have installed a DB2 database product, when SELinux was either uninstalled or disabled, and want to enable SELinux, these are the steps:

- Install SELinux rpms if necessary.
- Change `/etc/sysconfig/selinux`; set the status to "permissive" or "enforcing".
- Reboot the machine to apply SELinux labels to all files.

## **db2chgpath - Change embedded runtime path**

- Run **db2chgpath** to set the SELinux attribute that allows DB2 shared libraries with text relocations to be loaded (`textrel_shlib_t`).

## 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.

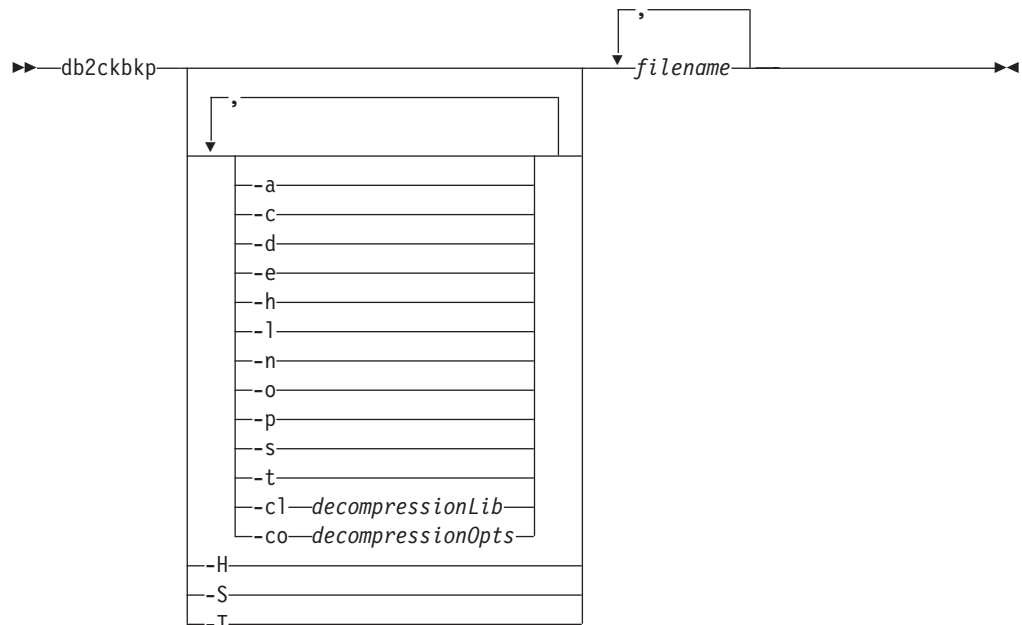
### Authorization

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 `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>
```

### Note:

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

For DMS table spaces:

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

### Note:

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 table spaces data. It only shows the total of all pages as SQLUDMSTABLESPACEDATA. The object types for SQLUDMSLOBDATA and SQLUDMSLONGDATA will be zero for DMS table spaces.
- 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.

**Note:**

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 operating systems)

```
db2ckbkp SAMPLE.0.krodger.DBPART000.19990817150714.001 SAMPLE.0.krodger.DBPART000.19990817150714.002
SAMPLE.0.krodger.DBPART000.19990817150714.003
```

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

**Note:** Using "CATN####" is only applicable to previous versions that used this form of naming convention.

## 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/
```

## db2ckbkp - Check backup

```
SQL00001/LOGSTREAM0000
Backup Buffer Size      -- 4194304
Number of Sessions     -- 1
Platform               -- 0
```

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

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

**Note:** Using "CATN####" is only applicable to previous versions that used this form of naming convention.

Example 3: The following example is sample output that is displayed from a backup image of 3 members in a DB2 pureScale environment.

```
BufAddr  MemberNum PoolID Token Type Offset FileSize ObjectSize OrigSize Object Name
-----
00000000:      0      0      0  19      0      268      268      0 "BACKUP.START.RECORD.MARKER"

numTbpsInDB : 3
numTbpsInImg : 3

Total members : 3
Member numbers: 0,1,2
```

### 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 Linux and UNIX operating 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.

## db2cklog - Check archive log file validity

You use the **db2cklog** command to check the validity of archive log files in order to determine whether or not the log files can be used during rollforward recovery of a database or table space. Either a single archive log file or a range of archive log files can be checked.

Archive log files that pass validation by the **db2cklog** command without any DBT error messages or warnings can be used during a rollforward recovery operation. If an archive log file fails validation with an error message or if a warning is returned, then you must not use that log file during rollforward recovery. A log file that returns an error during validation by the **db2cklog** command will cause the recovery operation to fail. If the validation of a log file returns a warning, then that log file might be invalid, unless the log file is still active. Only log files that are closed, such as archive log files, can be validated successfully.

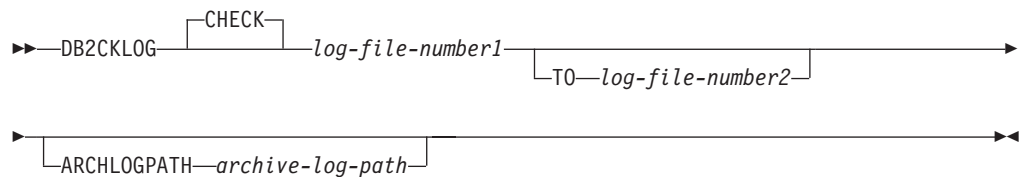
### Authorization

Anyone can run the command, but you must have read permission on the archive log files.

### Required connection

None

### Command syntax



### Command parameters

#### CHECK

Validates the archive log file or the range of archive log files by performing checks on the internal validity of the files. This is the default action.

#### *log-file-number1*

Specifies the numeric identifier of the log file to validate. For example, the numeric identifier of the S0000001.LOG log file is 1. If the *TO log-file-number2* parameter is also specified, then *log-file-number1* represents the first numeric identifier in a range of log files to check.

#### **TO** *log-file-number2*

Specifies that a range of numbered log files is to be validated (ranging from *log-file-number1* to *log-file-number2*). If *log-file-number2* is numbered lower than *log-file-number1*, then only *log-file-number1* is checked.

#### **ARCHLOGPATH** *archive-log-path*

Specifies a relative or an absolute path where the archive log files are stored. The default path is the current directory.

## db2cklog - Check archive log file validity

### Example

The following example shows the successful validation of the archive log file S0000003.LOG in the path tests (output is abridged). This file can be used during rollforward recovery.

```
$ db2cklog CHECK 3 ARCHLOGPATH tests
```

```
_____
      D B 2 C K L O G
_____
      DB2 Check Log File tool

...

"db2cklog": Finished processing log file "S0000003.LOG". Return code: "0".
```

The following example shows the successful validation of a range of archive log files (S0000003.LOG to S0000005.LOG; output is abridged). Successful validation of each file is indicated in the output. These files can be used during rollforward recovery.

```
$ db2cklog 3 T0 5
```

```
_____
      D B 2 C K L O G
_____
      DB2 Check Log File tool

...

"db2cklog": Finished processing log file "S0000003.LOG". Return code: "0".

...

"db2cklog": Finished processing log file "S0000004.LOG". Return code: "0".

...

"db2cklog": Finished processing log file "S0000005.LOG". Return code: "0".
```

The following example shows how the first log file in a range of archive log files returns an error and fails validation (output is abridged). Once an error is encountered, a DBT error message is returned, and the **db2cklog** command exits without processing further log files. This log file should not be used for rollforward recovery, because it will cause the recovery operation to fail.

```
$ db2cklog 0 T0 1
```

```
_____
      D B 2 C K L O G
_____
      DB2 Check Log File tool

...

DBT7053E Log file validation failed because the specified log file contains an
invalid log page followed by another invalid log page.

DBT7048E The db2cklog utility determined that the current log file is invalid.

"db2cklog": Finished processing log file "S0000000.LOG". Return code: "-2000".
```

## 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.

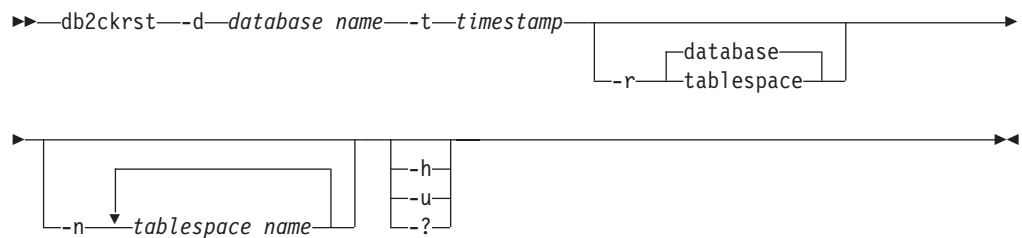
### Authorization

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
```

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

## db2ckrst - Check incremental restore image sequence

```
> 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 **db2ckrst** utility from being able to correctly analyze 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.

## db2ckupgrade - Check database for upgrade

Verifies that a database can be migrated.

### Scope

In a partitioned database environment, the **db2ckupgrade** command will check each database partition.

### Authorization

SYSADM

### Required connection

None

### Command syntax

```

▶▶ db2ckupgrade database [-l filename] [-not1]
                    [-e]
▶▶ [-u userid] [-p password]

```

### Command parameters

*database*

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

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

**-l filename**

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

**-not1** Disables the check for type-1 indexes. If this option is not specified, the **db2ckupgrade** command checks for type-1 indexes and generates the `type1_index_database-name.db2` script file, in the same directory indicated for the log file, containing **REORG INDEXES ALL** commands with the **ALLOW WRITE ACCESS** and **CONVERT** clauses for each identified type-1 index.

**-p password**

Specifies the password of the system administrator's user ID.

**-u userid**

Specifies the user ID of the system administrator.

### Usage notes

To run the **db2ckupgrade** command:

- On Linux and UNIX operating systems, install a new DB2 copy to which you want to upgrade. Then run the **db2ckupgrade** command from the `DB2DIR/bin` directory, where `DB2DIR` is the location where the DB2 copy is installed.
- On Windows operating systems, insert the DB2 database product CD to which you want to upgrade. Then run the **db2ckupgrade** command from the `db2\Windows\Utilities` directory on the CD.

## db2ckupgrade - Check database for upgrade

The **db2ckupgrade** command fails to run against databases which are catalogued as remote databases.

This command verifies that all the following conditions are true:

- A catalogued database actually exists.
- A database is not in an inconsistent state.
- A database is not in a backup pending state.
- A database is not in a restore pending state.
- A database is not in rollforward pending state.
- Tables are not in a load pending state.
- Tables are not in a redistribute pending state.
- For Version 9.8 or later, that all table space container paths use the same mount point.
- For Version 9.8 FixPack 3 or later, that the I/O write operations for the database are not suspended or are not being suspended.
- That there are no MQT's that depend on system views.
- Table spaces are in a normal state.
- A database does not contain user-defined types (UDTs) with the name ARRAY, BINARY, CURSOR, DECFLOAT, ROW, VARBINARY, or XML.
- A database does not contain the built-in DATALINK data type.
- A database does not have a schema with the name SYSPUBLIC.
- A database does not have orphan rows in system catalog tables that would cause database upgrade to fail.
- A database enabled as an HADR primary database allows successful connections.
- An HADR database role is not standby.
- If SYSCATSPACE is a DMS table space and AUTORESIZE is not enabled, SYSCATSPACE has at least 50% free pages of total pages.
- A database is not enabled for XML Extender.

A local database must pass all of these checks to succeed at the upgrade process. The **db2iupgrade** command calls the **db2ckupgrade** command with the **-not1** parameter, and specifies update.log as the log file for **db2ckupgrade**. The default log file created for **db2iupgrade** is /tmp/db2ckupgrade.log.processID. The **db2iupgrade** fails if the **db2ckupgrade** command finds that any of the previously listed conditions are not true, and returns the DBI1205E error code. The user needs to resolve these errors before upgrading the instance.

The **db2ckupgrade** command writes a warning message to the log file, as specified with the **-1** parameter, for any of the following conditions:

- Column names, routine parameter names, or variable names are called NULL.
- Type-1 indexes exist in the database.
- Workload connection attributes contain asterisks (\*).
- Database is enabled for DB2 WebSphere® MQ functions.

During installation on Windows operating systems, if you select a DB2 copy with the **upgrade** action in the **Work with Existing** window and you have local databases cataloged on your instances, a message box will warn you that you must run the **db2ckupgrade** command from the DB2 database product CD. Then you can choose one of the following actions:



## db2ckupgrade - Check database for upgrade

- Ignore the message and continue the installation process.
- Run the **db2ckupgrade** command. If this command runs successfully, continue the installation process. If you find errors, quit the installation process, fix any errors, and then rerun the installation process.
- Quit the installation process.

For Version 9.5 databases, if the **-not1** parameter is omitted, the **db2ckupgrade** command calls the **db2IdentifyType1** command to identify type-1 indexes and to generate a script to convert type-1 indexes to type-2 indexes for a specified database. The **db2IdentifyType1** command can take a long time to complete its processing. The running time of the **db2IdentifyType1** command is proportional to the number of tables in the database and the number of database partitions. Take into account the following performance considerations:

- For Version 9.5 databases with a large number of tables, large number of database partitions, or both, first run the **db2IdentifyType1** command on specific schemas or tables by using the **-s** or **-t** parameters until you process all your tables. Then run the **db2ckupgrade** command with the **-not1** parameter.
- For Version 9.5 partitioned database environments, run the **db2ckupgrade** command from one database partition, preferably the catalog partition for faster performance, to detect all type-1 indexes.

To verify that a database is ready for upgrade, refer to “Verifying that your databases are ready for upgrade” in *Upgrading to DB2 Version 10.1*.

## db2cli - DB2 interactive CLI

Starts the interactive Call Level Interface (CLI) environment for design and prototyping in CLI.

### Authorization

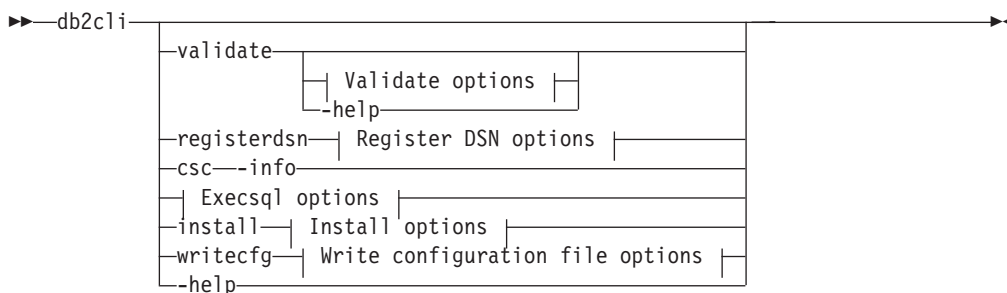
For the IBM Data Server Driver for ODBC and CLI, the **db2cli** command is located in the `bin` subdirectory of the directory where you installed this driver. For the IBM Data Server Client, this command is located in the `home_dir/sql1lib/bin` directory where `home_dir` is the home directory of the instance owner on UNIX operating systems. The command is located in the `DB2PATH\bin` where `DB2PATH` is the installation location of the DB2 copy on Windows operating systems.

None

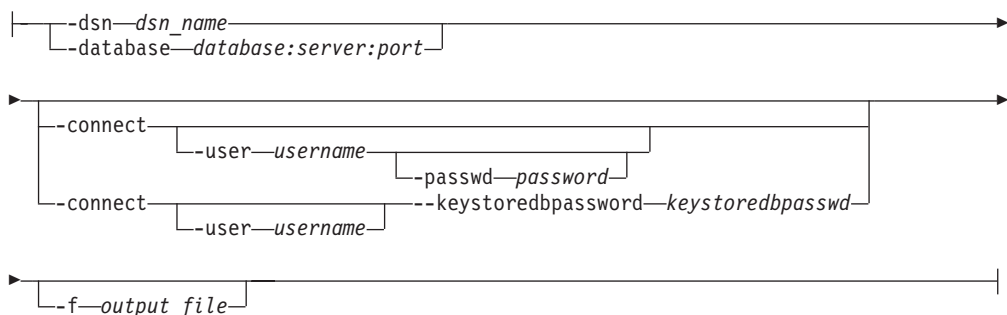
### Required connection

None

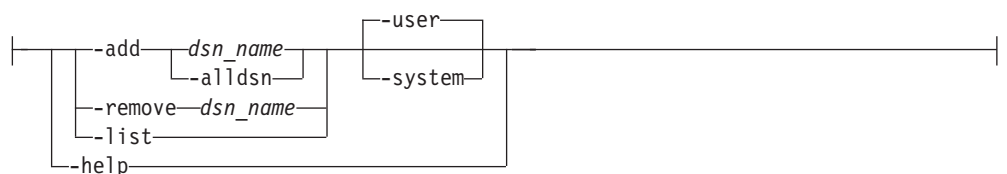
### Command syntax

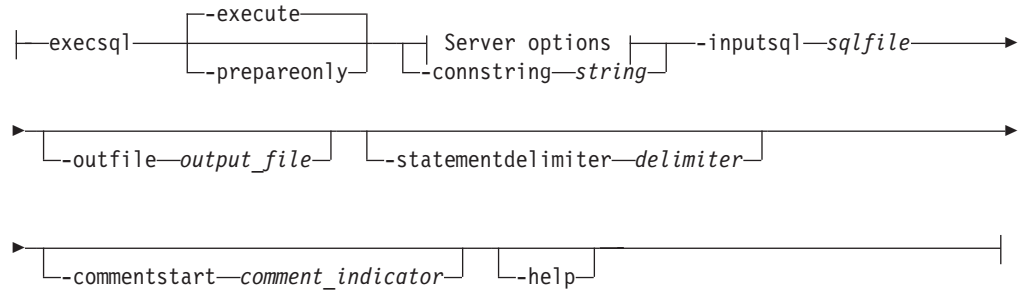
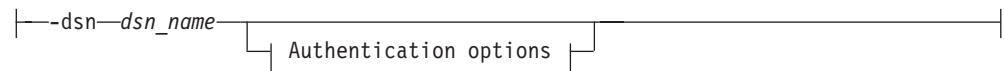
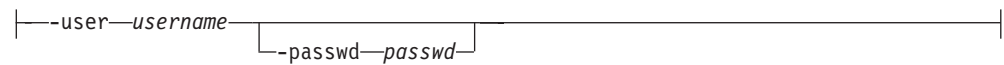
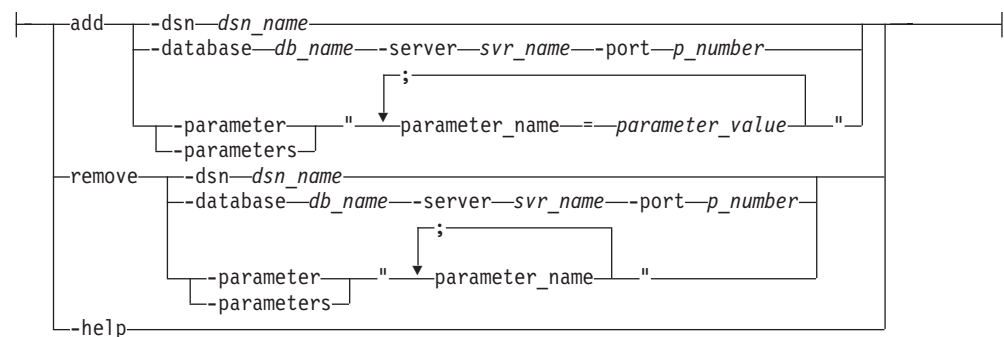


### Validate options:



### Register DSN options:



**Execsql options:****Server options:****Authentication options:****Install options (available only on Windows):****Write configuration file options:****Command parameters****validate**

Validates and tests the CLI environment configuration. It shows a list of keywords that are found in the `db2cli.ini` and `db2dsdriver.cfg` files. If any keywords are invalid for the specified data source or database name, they are listed as UNKNOWN.

## db2cli - DB2 interactive CLI

- dsn** *dsn\_name*  
Specifies the data source name (DSN) to validate.
  - database** *database:server:port*  
Specifies the database name, server name, and port number to validate. You must specify the information by using a colon to separate the database name, server name, and port number. The server name must consist of a fully qualified domain name (FQDN). For example, TESTDB:dbserver.example.com:19677
  - connect**  
Specifies the DSN or database to which the **db2cli validate** command connects and writes information about the connection attempt to the command output.
  - user** *username*  
Specifies the user name to use for the connection.
  - passwd** *password*  
Specifies the password to use for the connection.
  - keystoredbpassword** *keystoredbpasswd*  
Specifies the password for the SSL connection when authentication parameter is set to **CERTIFICATE**. The **db2cli validate -dsn -connect -keystoredbpassword** command is available starting in DB2 Version 9.7 Fix Pack 6 and later.
- Note:** If connecting to a gateway that is at a lower version than the client version, new features that are supported with a corresponding client-server combination might not be available.
- f** *OUTPUT\_FILE*  
Specifies an optional output file for the results of the **db2cli** command.
  - help** Shows the help information that is related to the usage of the **validate** parameter.

### registerdsn

Specifies the **db2cli** register DSN mode. Use this command parameter to register a DSN in the Windows operating system.

- add** *dsn\_name* | **-alldsn**  
Adds system or user ODBC data sources to the Microsoft ODBC Data Source Administrator.
- dsn\_name*  
Indicates the DSN to register. For DB2 Version 9.7 Fix Pack 4 and earlier releases, the value of *dsn\_name* must be the DSN that is defined in the `db2cli.ini` file or `db2dsdriver.cfg` file and cannot be a DBALIAS name. Starting in DB2 Version 9.7 Fix Pack 5 and later fix packs, the **-add** *dsn-name* parameter adds data sources for cataloged databases that are available in the local database directory. The *dsn-name* can indicate a data source that is defined in the `db2cli.ini` or `db2dsdriver.cfg` file, and a database alias in the local database directory in IBM Data Server clients.

- alldsn**  
Registers all the data sources that are defined in the

db2cli.ini file and db2dsdriver.cfg file. You must use this parameter with the **-add** parameter.

The **registerdsn -add -alldsn** parameter continues to search for data sources that are available only in the db2cli.ini and db2dsdriver.cfg files and does not add data sources for cataloged databases.

**-user** Registers data sources as user ODBC data sources. If no parameter is specified, data sources are registered as user ODBC data sources.

**-system**

Registers data sources as system ODBC data sources.

**-remove** *dsn\_name*

Removes a system or user ODBC Data Source from the Microsoft ODBC Data Source Administrator.

**-list** Lists all the system or user IBM Data Server ODBC Data Sources that are registered in the Microsoft ODBC Data Source Administrator.

**-help** Displays help information related to the **db2cli** register DSN mode.

**csc -info**

Displays connection supervisor client (CSC) information. A CSC is a dynamically loadable library that CLI uses for the purpose of end-to-end monitoring.

**-execsql**

Executes or prepares the SQL statements specified in an input file. Can also save output to a file.

**-execute**

Specifies that the SQL statements in the SQL script file are prepared and run. This is the default if no parameter is specified. Results are displayed in the console. To save the output to a file, specify the option **-output** with the absolute or relative path of the file. SQL statements in the SQL script file cannot have parameter markers.

**-prepareonly**

Specifies that the SQL statements in the file specified by the **-inputSql** option are prepared but not run. Use this option to check the syntax of the SQL statements without running the statements.

**-commentstart** *comment\_indicator*

Specifies the character combination that appear at the beginning of a line to indicate a comment line. The default value of *comment\_indicator* is two dashes (--). If a comment spans multiple lines, start each line with the *comment\_indicator* character combination. The maximum length of a comment line is 128 characters. In the input SQL file, the text after the comment characters can contain statement cursor attributes. The cursor attributes apply to the SQL statement immediately following the comment.

## db2cli - DB2 interactive CLI

- connstring** *string*  
Specifies the database name, server, and port number for the target database. The information must be specified using a colon to separate the database name, server and port number. For example, `TESTDB:dbserver.example.com:19677`, where `TESTDB` is the database name, `dbserver.example.com` is the server, and `19677` is the port number.
- dsn** *dsn\_name*  
Specifies the data source name.
- help** Displays summary usage information.
- inputsql** *sqlfile*  
Specifies the input file that contains SQL statements. The value of *sqlfile* is the absolute or relative path of the file. The SQL statements in the file are separated by a delimiter. Only preparable statements are used. DB2 commands such as **DESCRIBE TABLE** and **BIND** are not allowed.  
  
Only SQL statements and comments can be in the input SQL file. The file cannot contain CLI specific attributes or keywords. Batch SQL statements are not supported.
- outfile** *outputfile*  
Specifies the absolute or relative path of the file to store the output results. When this option is not specified, the results are displayed in the console.
- passwd** *password*  
Specifies the password to use for authenticating with the database.
- statementdelimiter** *delimiter*  
Specifies the character combination that is used in the input SQL file to separate statements. The default value of delimiter is a carriage return.
- user** *username*  
Specifies the user for authenticating with the database.

### install

- Registers or unregisters the IBM Data Server Driver for ODBC and CLI in the Windows registry.
- setup** Registers the current IBM Data Server Driver for ODBC and CLI in the Windows registry under ODBC. It also creates the necessary configuration folders and sample configuration files at the application data path.
  - cleanup**  
Unregisters the current IBM Data Server Driver for ODBC and CLI from the Windows registry under ODBC. The cleanup then removes the folders, configuration sample files, and license management files that are created at the application data path for the current installation. Any user-created files or folders are not removed unless you specify the **-force** option.
    - force** When specified with the **-cleanup** option, the entire install-specific folder is removed from the application data path. This folder belongs to this installation only. Any

user-created files or folders in this folder are removed without any prompt or warning.

**-help** Shows the help information that is related to the usage of the **install** parameter.

**writetcfg add|remove**

Updates the db2dsdriver.cfg configuration file.

**add -dsn | -database | -parameter[s]**

Adds information about data sources, databases, or parameters to the db2dsdriver.cfg configuration file.

**-dsn** *dsn\_name*

Specifies the new DSN that is to be added for an existing database element or an existing DSN to which a parameter element is to be added or updated.

If there is no *dsn* element with the *dsn\_name* in the configuration file, you must indicate the database connection information. A new *dsn* element in the *dsncollection* section is added for the database. If you do not specify the database connection information, the SQL1531N error message is returned.

If there is a *dsn* element with the *dsn\_name* in the configuration file, you must indicate the database connection information with the **-database** parameter. This connection information must match the existing connection information in the configuration file. Otherwise, the SQL1532N error message is returned.

**-database** *db\_name-server svr\_name-port port\_number*

Specifies the connection information for a database element. This information consists of the database name, the server name where the database resides, and the port number of the database server.

If a database element with the same connection information exists in the db2dsdriver.cfg configuration file, the specified parameter element is added to this database element. Otherwise, the specified parameter element is added to a new database element.

**-parameter[s]** "*<par1=par1\_val[;...;parN=parN\_val]*"

Specifies the parameter information to add or update a parameter element for the specified database or data source in the db2dsdriver.cfg configuration file.

To add a parameter to the global section, specify the parameter information without indicating a database or a data source.

If there is no parameter element for the specified data source or database in the configuration file, the parameter element is added to the specified data source or database.

If there is a parameter element for the specified data source or database in the configuration file, the existing value is updated with the value that you specify with

**-parameter[s]**.

Special characters, such as path separators in the value, must be preceded by the escape character “\”.

The **db2cli writcfg** command does not verify the syntax or validate the value of the specified parameter.

### **remove -dsn | -database | -parameter[s]**

Removes information about data sources, databases, or parameters from the `db2dsdriver.cfg` configuration file.

#### **-dsn** *dsn\_name*

Specifies the DSN for which you want to remove the parameter information or the entire data source section in the configuration file.

To remove parameter information, specify the corresponding DSN and the parameter information.

To remove the entire data source section, specify only the DSN without any parameter information.

#### **-database** *db\_name-server svr\_name-port port\_number*

Specifies the connection information for a database for which you want to remove the parameter information or the entire database section in the configuration file.

To remove parameter information, specify the database connection information for the corresponding database section and the parameter information.

To remove the entire database section, specify only the database connection information without any parameter information.

#### **-parameter[s]** “*par1[;par2;...;parN]*”

Specifies the parameter information that is to be removed from the specified database or data source in the `db2dsdriver.cfg` configuration file.

If the indicated parameter is not in the specified database or data source section of the configuration file, no action is taken.

Special characters, such as path separators in the value string, must be preceded by the character “\”.

**-help** Shows the help information that is related to the usage of the **writcfg** parameter.

**-help** Shows the **db2cli** command help information.

## Usage notes

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

Three 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 CLI.

**Support commands**

Commands that do not have an equivalent CLI function.

**Additional modes**

You can use the additional modes for the **db2cli** command to validate and test the CLI environment configuration.

You can issue commands interactively or from within a file. Similarly, you can display the command output on the terminal or write it to a file. A useful feature of the IBM Data Server Driver for ODBC and CLI is the ability to capture all of the commands that are entered during a session, and to write them to a file, thus creating a *command script* that you can rerun at a later time.

For IBM Data Server client packages on Windows 64-bit operating systems, the 32-bit version of **db2cli** (db2cli32.exe) is supported in addition to the 64-bit version of the **db2cli** command.

SQL statements are executed by using the `SQLExecDirect()` function. When executing SQL statements, the **db2cli** `execsql` command uses the database settings specified in the `db2cli.ini` and `db2dsdriver.cfg` files.

Error messages returned by the **db2cli** command are formatted using the same format as the `SQLGetDiagRec()` function.

**Examples****db2cli validate**

In the following example, the utility reads the [COMMON] section of `db2cli.ini` file, the [DSN] section for the **sample** DSN name in the `db2cli.ini` file, and the `<dsn>` element for the **sample** DSN name in the `db2dsdriver.cfg` file. Both valid and invalid keywords are displayed; invalid keywords are listed as UNKNOWN.

```
db2cli validate -dsn sample
```

In the next example, the utility reads the [COMMON] section of `db2cli.ini` file, the `<database>` section for the database **dbname1**, the server **server1.net1.com**, and port **50001**, in the `db2dsdriver.cfg` file. Both valid and invalid keywords are displayed; invalid keywords are listed as UNKNOWN.

```
db2cli validate -database dbname1:server1.net1.com:50001
```

**db2cli writecfg**

For the examples that are described in this section, assume that the `db2dsdriver.cfg` file has the following content:

```
<configuration>
  <dsncollection>
    <dsn alias="alias1" name="name1" host="server1.net1.com" port="50001">
      <parameter name="DisableAutoCommit" value="TRUE"/>
    </dsn>
  </dsncollection>
  <databases>
    <database name="name1" host="server1.net1.com" port="50001">
      <parameter name="CurrentSchema" value="OWNER1"/>
    </database>
  </databases>
```

## db2cli - DB2 interactive CLI

```
<parameters>
  <parameter name="IsolationLevel" value="SQL_TXN_READ_COMMITTED"/>
</parameters>
</configuration>
```

The following example adds a new data source element to the `db2dsdriver.cfg` configuration file:

```
$db2cli writecfg add -dsn alias2 -database name2 -server server1.net1.com -port 50001
```

As a result of this command, the `dsncollection` section is modified as follows:

```
<dsncollection>
  <dsn alias="alias1" name="name1" host="server1.net1.com" port="50001">
    <parameter name="DisableAutoCommit" value="TRUE"/>
  </dsn>
  <dsn alias="alias2" name="name2" host="server1.net1.com" port="50001"/>
</dsncollection>
```

The following example adds parameter information to an existing data source in the configuration file:

```
db2cli writecfg add -dsn alias2 -parameters "DisableAutoCommit=FALSE;CurrentSchema=OWNER2;pureQueryXml=C:\\clico"
```

As a result of this command, the `dsncollection` section is modified as follows:

```
<dsncollection>
  <dsn alias="alias1" name="name1" host="server1.net1.com" port="50001">
    <parameter name="DisableAutoCommit" value="TRUE"/>
  </dsn>
  <dsn alias="alias2" name="name2" host="server1.net1.com" port="50001">
    <parameter name="DisableAutoCommit" value="FALSE"/>
    <parameter name="CurrentSchema" value="OWNER2"/>
    ..<parameter name="pureQueryXml" value="C:\\clico"/>
  </dsn>
</dsncollection>
```

The following example adds a new database element with parameters in the configuration file:

```
db2cli writecfg add -database name2 -server server1.net1.com -port 50001 -parameters "LockTimeout=20;KeepAliveTimeout=20000"
```

As a result of this command, the `databases` section is modified as follows:

```
<databases>
  <database name="name1" host="server1.net1.com" port="50001">
    <parameter name="CurrentSchema" value="OWNER1"/>
  </database>
  <database name="name2" host="server1.net1.com" port="50001">
    <parameter name="LockTimeout" value="20"/>
    <parameter name="KeepAliveTimeout" value="20000"/>
  </database>
</databases>
```

The following example modifies an existing parameter for an existing `dsn` element in the configuration file:

```
db2cli writecfg add -dsn alias1 -parameter "DisableAutoCommit=FALSE"
```

As a result of this command, the `dsncollection` section is modified as follows:

```
<dsncollection>
  <dsn alias="alias1" name="name1" host="server1.net1.com" port="50001">
    <parameter name="DisableAutoCommit" value="FALSE"/>
  </dsn>
  <dsn alias="alias2" name="name2" host="server1.net1.com" port="50001">
```

```

    <parameter name="DisableAutoCommit" value="FALSE"/>
    <parameter name="CurrentSchema" value="OWNER2"/>
    ..<parameter name="pureQueryXml" value="C:\clico"/>
  </dsn>
</dsncollection>

```

The following example adds a parameter element to the global section in the configuration file:

```
db2cli writecfg add -parameter "RecieveTimeout=20000"
```

As a result of this command, the global section is modified as follows:

```

<parameters>
  <parameter name="IsolationLevel" value=" SQL_TXN_READ_COMMITTED"/>
  <parameter name="RecieveTimeout" value="20000"/>
</parameters>

```

The following example removes a parameter element from a database in the configuration file:

```
db2cli writecfg remove -database name1 -server server1.net1.com -port 50001 -parameter "CurrentSchema"
```

As a result of this command, the *databases* section is modified as follows:

```

<databases>
  <database name="name1" host="server1.net1.com" port="50001">
  </database>
  <database name="name2" host="server1.net1.com" port="50001">
    parameter name="LockTimeout" value="20"/>
    parameter name="KeepAliveTimeout" value="20000"/>
  </database>
</databases>

```

The following example removes a *dsn* element in the configuration file:

```
db2cli writecfg remove -dsn alias1
```

As a result of this command, the *dsncollection* section is modified as follows:

```

<dsncollection>
  <dsn alias="alias2" name="name2" host="server1.net1.com" port="50001">
    <parameter name="DisableAutoCommit" value="FALSE"/>
    <parameter name="CurrentSchema" value="OWNER2"/>
    <parameter name="pureQueryXml" value="C:\clico"/>
  </dsn>
</dsncollection>

```

### db2cli execsql

In following example assume that the following table and procedures are created in the SAMPLE database:

```

create table employee(empid integer, empname varchar(100))

CREATE PROCEDURE proc1 ( )
DYNAMIC RESULT SETS 1 P1:
BEGIN
  DECLARE cursor1 CURSOR WITH RETURN FOR SELECT * FROM fprem;
  OPEN cursor1;
END P1

CREATE PROCEDURE PROC2(IN ID1 INTEGER,OUT NAME VARCHAR(20))
BEGIN
DECLARE CUR1 CURSOR WITH RETURN TO CALLER FOR SELECT * FROM EMPLOYEE1 WHERE ID=ID1;
  OPEN CUR1;
END

```

This example also assumes the SQL file test.sql contains the following text:

## db2cli - DB2 interactive CLI

```
--Populate table( employee )
insert into employee(empid, empname) values(1, 'Adam')
insert into employee(empid, empname) values(2, 'Atul')
select empid, empname from employee

--Execute the stored procedure
Call proc1( )
```

Enter the following **db2cli** command in a console window to run the SQL statements in the file:

```
db2cli execsql -dsn sample -inputsq1 test.sql
```

The following text is displayed in the console window:

```
IBM DATABASE 2 Interactive CLI Sample Program
(C) COPYRIGHT International Business Machines Corp. 1993,1996
All Rights Reserved
Licensed Materials - Property of IBM
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
```

```
insert into employee(empid, empname) values(1, 'Adam')
The SQL command completed successfully.
```

```
insert into employee(empid, empname) values(2, 'Atul')
The SQL command completed successfully.
```

```
select empid, empname from employee
```

```
EMPID EMPNAME
1, Adam
2, Atul
```

```
Call proc1()
```

```
EMPID EMPNAME
1, Adam
2, Atul
```

Run a CALL statement for a stored procedure that has OUT arguments. The question mark (?) can be used as an OUT parameter.

The following example assumes that an SQL script file test2.sql contains the following text:

```
CALL PROC2( 1, ?)
```

Enter the following **db2cli** command in a console window to run the SQL statements in the file:

```
db2cli execsql -dsn sample -inputsq1 test2.sql
```

The following text is displayed in the console window:

```
Value of output parameters
-----
Parameter Name : NAME
Parameter Value : -

ID
-----
1
```

Specify the **-prepareonly** option to prepare the SQL statements without running them. The DDL statements that are needed for the SQL statements must be run before you run the **db2cli** **execsql** command with the **-prepareonly** option.

The following example assumes that the SQL file test3.sql contains the following text:

```
--populate table( employee )
insert into employee(empid, empname) values(1, 'Adam');
insert into employee(empid, empname) values(2, 'Atul');
select empid, empname from employee;
```

Also, assume that the table EMPLOYEE was created in the database. Enter the following **db2cli** command in a console window to prepare the SQL statements in the file:

```
db2cli execsql -prepareonly -dsn sample -inputsq1 test3.sql
```

The following text is displayed in the console window:

```
IBM DATABASE 2 Interactive CLI Sample Program
(C) COPYRIGHT International Business Machines Corp. 1993,1996
All Rights Reserved
Licensed Materials - Property of IBM
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
```

```
insert into employee(empid, empname) values(1, 'Adam')
The SQL command prepared successfully.
```

```
insert into employee(empid, empname) values(2, 'Atul')
The SQL command prepared successfully.
```

```
select empid, empname from employee
The SQL command prepared successfully.
```

If you place DDL statements that are required for DML statements in the same file, the DML statements that require the DDL statements fail.

For example, assume that the following text is in the file test4.sql, and assume that the EMPLOYEE table has not been created in the database:

```
--create and populate table( employee )
create table employee(empid integer, empname varchar(100));
insert into employee(empid, empname) values(1, 'Adam');
insert into employee(empid, empname) values(2, 'Atul');
select empid, empname from employee;
```

```
-- try to create another table with the same name
create table employee(empid integer, empname varchar(100));
```

The CREATE TABLE statement must be run before the INSERT and SELECT statements can be run successfully.

Enter the following **db2cli** command in a console window to prepare the SQL statements in the file:

```
db2cli execsql -prepareonly -dsn sample -inputsq1 test4.sql
```

The following text is displayed in the console window:

```
IBM DATABASE 2 Interactive CLI Sample Program
(C) COPYRIGHT International Business Machines Corp. 1993,1996
All Rights Reserved
Licensed Materials - Property of IBM
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
```

```
create table employee(empid integer, empname varchar(100))
The SQL command prepared successfully.
```

```
insert into employee(empid, empname) values(1, 'Adam')
The SQL command failed. During SQL processing it returned:
[IBM][CLI Driver][DB2/6000] SQL0204N "EMPLOYEE" is an undefined name.  SQLSTATE=42704
```

```
insert into employee(empid, empname) values(2, 'Atul')
The SQL command failed. During SQL processing it returned:
```

## db2cli - DB2 interactive CLI

```
[IBM][CLI Driver][DB2/6000] SQL0204N "EMPLOYEE" is an undefined name.  SQLSTATE=42704  
select empid, empname from employee  
The SQL command failed. During SQL processing it returned:  
[IBM][CLI Driver][DB2/6000] SQL0204N "EMPLOYEE" is an undefined name.  SQLSTATE=42704  
create table employee(empid integer, empname varchar(100))  
The SQL command prepared successfully.
```

In this example, the two CREATE SQL statements prepared successfully, however the EMPLOYEE table was not created in the database. The INSERT and SELECT statements did not prepare successfully because the EMPLOYEE table was not in the database.



---

### db2cptsa - Install or update DB2 HA scripts

Installs or updates the DB2 High Availability (HA) scripts in `/usr/sbin/rsct/sapolicies/db2` on UNIX and Linux operating systems. You need these DB2 HA scripts to use the IBM Tivoli System Automation for Multiplatforms (SA MP) with the DB2 HA feature.

#### Authorization

Root user authority

#### Required connection

None

#### Command syntax

```
db2cptsa [-c] [-f] [-r] [-h | -?]
```

#### Command parameters

- c** Verifies that the DB2 HA scripts exist in `/usr/sbin/rsct/sapolicies/db2`, and that they are at the proper level.
- f** Forces a reinstall of the DB2 HA scripts in `/usr/sbin/rsct/sapolicies/db2`. Without this argument, if the version of the DB2 HA scripts that are already installed is the same as or higher than the version of the scripts being installed, then the installed scripts are not overwritten.
- r** Removes the directory `/usr/sbin/rsct/sapolicies/db2`. This directory is where the DB2 HA scripts for SA MP are located. These scripts and this directory will only be removed if SA MP is not installed.
- h | -?** Displays help information.

#### Usage notes

By default, this utility installs the DB2 HA scripts in `/usr/sbin/rsct/sapolicies/db2` if they aren't already installed there, or if the version of the scripts already installed is older than the version of the scripts being installed. This utility installs or updates the DB2 HA scripts if and only if SA MP is already installed.

This command can be found on the DB2 install media in the `db2/plat/tsamp` directory, where *plat* is:

- `aix` for DB2 for AIX 5L™
- `linux` for DB2 for Linux on 32-bit AMD and Intel systems (x86)
- `linuxamd64` for DB2 for Linux on AMD64 and Intel EM64T systems (x64)
- `linuxppc` for DB2 for Linux on POWER® (System i and pSeries®) systems
- `linux390` for DB2 for Linux on System z9® and zSeries®

The command is also available at `DB2DIR/install/tsamp` directory where `DB2DIR` is the installation path of the DB2 database product for UNIX and Linux operating systems.



## db2dart - Database analysis and reporting tool

Examines databases for architectural correctness and reports any encountered errors.

Reports generated by the **db2dart** command are encoded in the same code page as the database being analyzed by the command. Similarly, parameter values specified as part of the **db2dart** command are interpreted with the same code page as the database being analyzed. The character string that is in the **db2dart** report file and input value is encoded in the database code page.

The **db2dart** command does not perform code page conversions.

When invoking the **db2dart** command, you can specify only one action. An action can support a varying number of options.

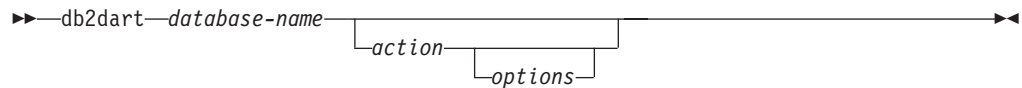
### Authorization

You must have SYSADM authority to use the **db2dart** command.

### 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 one or more tables. Requires two inputs: a table space ID, and either, the table object ID or a list of table object IDs, or the table name.
- /TSF** Inspects only storage group and table space files and containers.
- /TSC** Inspects the table space constructs of one or more table spaces, but does not inspect tables. Requires one input: the table space ID or a list of table space IDs.
- /TS** Inspects one or more table spaces and their tables. Requires one input: the table space ID or a list of table space IDs.
- /ATSC** Inspects constructs of all table spaces, but not their tables.

#### Data formatting actions

- /DD** Dumps formatted table data. If present, inline LOB data is also shown. 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.

- For nonpartitioned indexes on partitioned tables, the **/DI** action uses INDEX\_OBJECTID and TBSPACEID from SYSCAT.INDEXES as the first two inputs to the **/OI** and **/TSI** options. The table name (**/TN**) option is not supported for the action.
- For partitioned indexes on partitioned tables, the **/DI** action uses PARTITIONOBJECTID and TBSPACEID from SYSCAT.DATAPARTITIONS. The table name (**/TN**) option is not supported for the action.

**/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. The data shows whether a block has been reclaimed for use by the table space following a reorganization to reclaim multidimensional clustering (MDC) or insert time clustering (ITC) table blocks that were empty.

**/DP** Dumps pages in hex format.

- For permanent object in DMS table space, action **/DP** requires three input values consisting of table space ID, page number to start with, and number of pages.
- For permanent object in SMS table space, action **/DP** requires five input values consisting of table space ID, object ID, page number to start with, number of pages, and object type.

**/DTSF** Dumps formatted table space and storage group file information.

**/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.

The dumped delimited ASCII file is encoded in the database code page. The **db2dart** command does not perform code page conversions.

The **/DDEL** parameter supports only the following column data types. If a table contains columns with any other data type, the column is skipped and not included in the delimited ASCII file.

- SMALLINT
- FLOAT
- REAL
- INTEGER
- TIME
- DECIMAL
- CHAR()
- VARCHAR()
- DATE
- TIMESTAMP

- **BIGINT**

If a column of type CHAR and VARCHAR contains any binary data, or is defined with FOR BIT DATA, the **/DDEL** parameter generates the DEL file which contains the binary data. When you load data from the DEL file to the table using the **LOAD** command, ensure that you always specify the modified by delprioritychar option. When you insert data into the table from the DEL file using the **IMPORT** command, make sure that you always specify the modified by delprioritychar codepage=x option where x is the code page of the data in the input data set.

### **/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 (required high water mark).

## **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. For partitioned indexes, these values can be obtained from INDPARTITIONOBJECTID and INDPARTITIONTSPACEID for SYSCAT.INDEXPARTITIONS.

**/RHWM** Reduces high water mark through empty SMP extents. When specifying this parameter the database must be offline. Requires one input value: table space ID.

## **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. For the **/T** action, a comma-separated list of up to 64 objects IDs can be specified. If the corresponding **/TSI** option contains more than one input ID, only the first ID is used. Duplicate IDs are skipped. Logical IDs can be specified for the **/T** action.

## db2dart - Database analysis and reporting tool

### */TN table-name*

Specifies the table name.

### */TSI tablespace-id*

Specifies the table space ID. For the **/TS** or **/TSC** actions, a comma-separated list of up to 64 physical table space IDs can be specified. Duplicate IDs are skipped.

### */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.

### */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. When used with the **/DP** action, the **p** suffix can be used for pool relative addressing. 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.

### **/WHAT DBBP** *option*

Specifies the database backup pending state. Valid values are:

- OFF** Off state.  
**ON** On state.

### **/QCK** *sum*

Specifies which quick option to perform. You can specify one option or add the values together to perform multiple quick options.

- 1** The **/QCK 1** option applies to only the **/DB**, **/T**, and **/TS** actions. This option 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). This is the default option.
- 2** The **/QCK 2** option applies to only the **/DB**, **/T**, **/TS**, **/DD**, **/DI**, **/DM**, **/DEMP**, **/DDEL**, **/DXA**, and **/DXH** actions. This option skips the system catalog table lookup on nonpartitioned database environments and on the catalog partition of partitioned database environments. This option has no effect on non-catalog partitions of partitioned database environments. The **/QCK 2** option does not apply to the actions mentioned earlier if the **/TN** option is specified with a table name or if the **/OI** and **/TSI** options are specified with logical IDs.
- 4** The **/QCK 4** option applies to only the **/T**, **/TS**, and **/TSC** actions. This option skips special system catalog table inspection or system catalog table space inspection. For the **/TS**, and **/TSC** actions, the **/QCK 4** option skips the special system catalog table inspection. For the **/T** action, the **/QCK 4** option skips inspection of the system catalog table space constructs.
- 8** The **/QCK 8** option applies to only the **/T**, and **/TS** actions. This option skips the inspection of containers. For the **/T** action, the **/QCK 8** option skips the inspection of all container files. For the **/TS** action, the **/QCK 8** option inspects only container files that are associated with the specified table space.

### **/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.

## Examples

### Example 1

To dump 1000 pages of formatted index data on a non-range partitioned table with object ID 4 in a table space with ID 2 and starting with page zero, use the following **db2dart** command:

## db2dart - Database analysis and reporting tool

```
db2dart IXMLDB /DI /TSI 2 /OI 4 /V Y /PS 0 /NP 1000
```

### Example 2

To dump formatted block map data on range partitioned tables, use the following **db2dart** command:

```
db2dart IXMLDB /DM /TSI 2 /OI 8 /V Y /PS 0 /NP 1000
```

Where 8 is the value of **partitionobjectid** and 2 is the value of **tblspaceid** from SYSCAT.DATAPARTITIONS.

### Example 3

To dump formatted table space and storage group file information for database testdb, use the following command:

```
db2dart testdb /DTSF
```

The following is an example output generated by the previous command:  
Storage group file (automatic storage) report phase start.

```
Header version:          33
Header flavour:          1
Checksum:                0x5402d18f
Number of storage groups: 1
Default storage group ID: 0
Header Last LSN:        0x0000000000003D4A4

Storage group ID:        0
Storage group name:      IBMSTOGROUP
Data Tag:                5
Flavour:                 3
Version:                 5
State flags:             0x0000000000000000
Last LSN:                0x0000000000003D4A4
Initial LSN:            0x0000000000000000
Checksum:                0x1e587275

Number of storage paths: 1
Storage path # 0:        /filesystem1 (id = 0, state = 0x0)
```

Storage group file (automatic storage) report phase end.

Tablespace file report phase start.  
Tablespace information for current database:

-----

```
Number of defined tablespaces: 4
High water mark of used pools: 3
Number of disabled tablespaces: 0
```

Individual tablespace details:

-----

Information for Tablespace ID: 0

-----

```
Tablespace name: SYSCATSPACE
Table space flags (HEX): 3102
Table space type: Database Managed Space (DMS), Automatic Storage, Auto-Resize
Page size: 4096
Extent size: 4
Prefetch size: 4
Version: 104
Tablespace state: 0
```

```

Number of quiescers: 0
Storage Group ID: 0
Source Storage Group ID: -1
Data Tag: 0
Usable pages in tablespace: 25340
Total pages in tablespace: 25344
Initial Size: 1048576 bytes
Increment : -4096 bytes
Maximum Size: None
Last Resize: None
Last Resize Failed: No
SMP page for first free extent: 4
SMP page for last allocated tablespace extent. 4
SMP extent number of the last initialized SMP extent: 0
High Water Mark: 25164
.
.
.

```

#### Example 4

To inspect table spaces with IDs 3, 4, 5, 8, 9, and 11, and to skip the system catalog table lookup and the special system catalog table inspection, use the following **db2dart** command. Only the containers associated with the specified table spaces are inspected.

```
db2dart <dbname> /TS /TSI 3,4,5,6,9,11 /QCK 14
```

Where the /QCK 14 option represents the addition of quick options 2, 4, and 8. The /QCK 14 option performs all the individual operations of quick options 2, 4, and 8.

#### Usage notes

- 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 action, the options cannot be specified from the command line, and must be entered when prompted by **db2dart**.
- 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.
- 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.
- 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.
- For partitioned tables, the /DD, /DM, /DEMP, /DDEL, /DP, /DXA, /DXH actions use partitionobjectid and tbspaceid from syscat.datapartitions as the input to the table object ID (/OI) and table space ID (/TSI) for a specific partition. The table name option (/TN) is not supported for these actions. The /T action supports the table name or global table object ID when use with global table space ID to

## db2dart - Database analysis and reporting tool

check the entire table, and also supports using partitionobjectid and tbspaceid from syscat.datapartitions as the input to **/OI** and **/TSI** to check a specific partition.

- In general, run the **db2dart** command when the database is offline. However, you do not need an offline database if you are specifying either the **/DHWM** and **/LHWM** actions. The report could be generated without the database being offline, but the reliability of the results will vary depending on how much write/update activity has occurred recently (less activity implies more reliable results).



---

## db2daslevel - Show DAS level

Shows the current level of the DAS on the system.

Output from this command goes to the console by default.

**Important:** This command has been deprecated and might be removed in a future release because the DB2 administration server (DAS) has been deprecated in Version 9.7. For more information, see “DB2 administration server (DAS) has been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html>.

### Authorization

None

### Required Connection

None

### Command Syntax

▶▶—db2daslevel—————▶▶

### Command parameters

None

## 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.

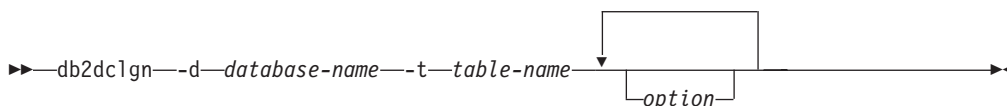
### Authorization

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 options:

#### -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, prefixed by "IND-" (for COBOL) or "ind\_" (for the other languages). The default behavior is to not generate indicator variables.

**-l** *language*

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

**-n** *name*

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

**-o** *output-file*

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

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

**-p** *password*

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

**-r** *remarks*

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

**-s** *structure-name*

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

**-u** *userid*

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

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

**-w** *DBCS-var-type*

Specifies whether `sqldbchar` or `wchar_t` is to be used for a GRAPHIC, VARGRAPHIC, or 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 particular server. Encoding can be either LUW or OS390. If OS390 is specified, the generated file would look identical to a file generated by OS/390.

## db2dc1gn - Declaration generator

### Examples

```
db2dc1gn -d sample -t emp_resume -l cobol -a replace
```

## db2diag - db2diag logs analysis tool

This utility is a tool to filter and format both single and rotating **db2diag** log files. The **db2diag** tool reads from rotating **db2diag** log files if the **diagsize** database manager configuration parameter is set. Otherwise, by default, it reads from the default `db2diag.log` file.

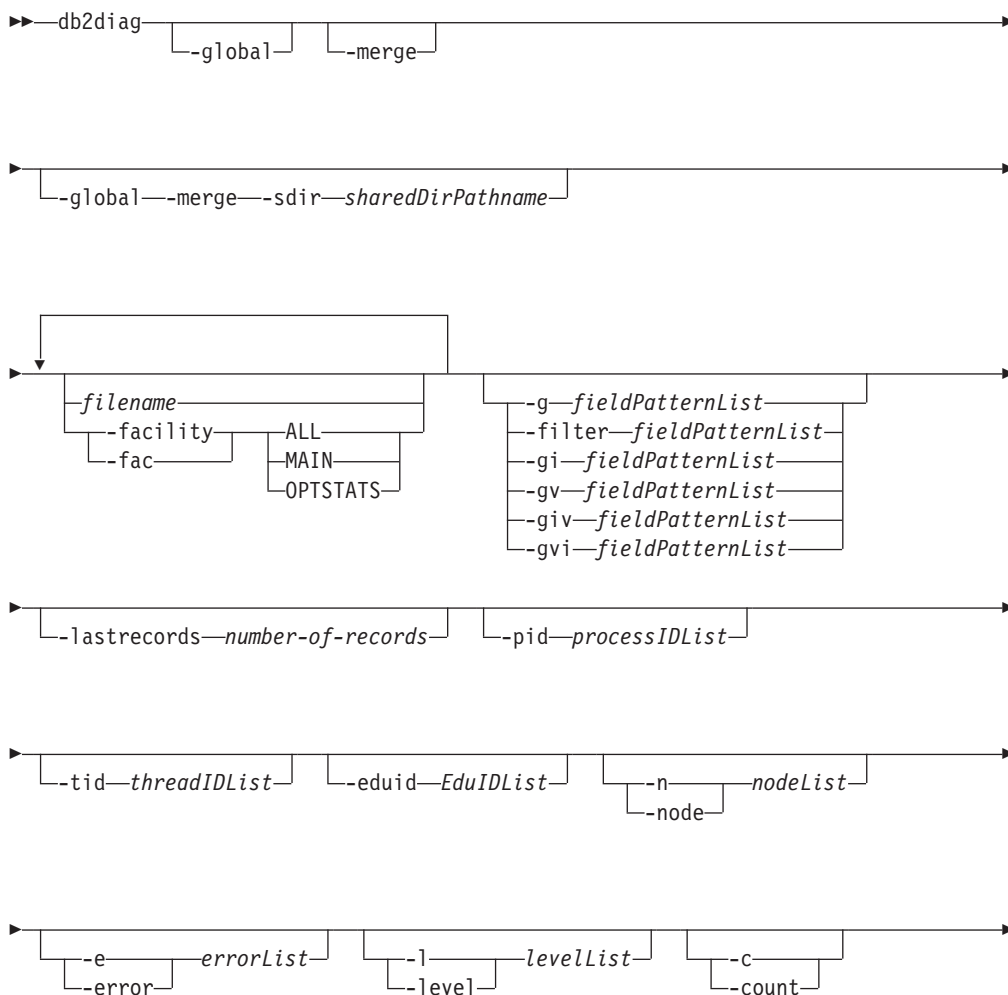
### Authorization

None

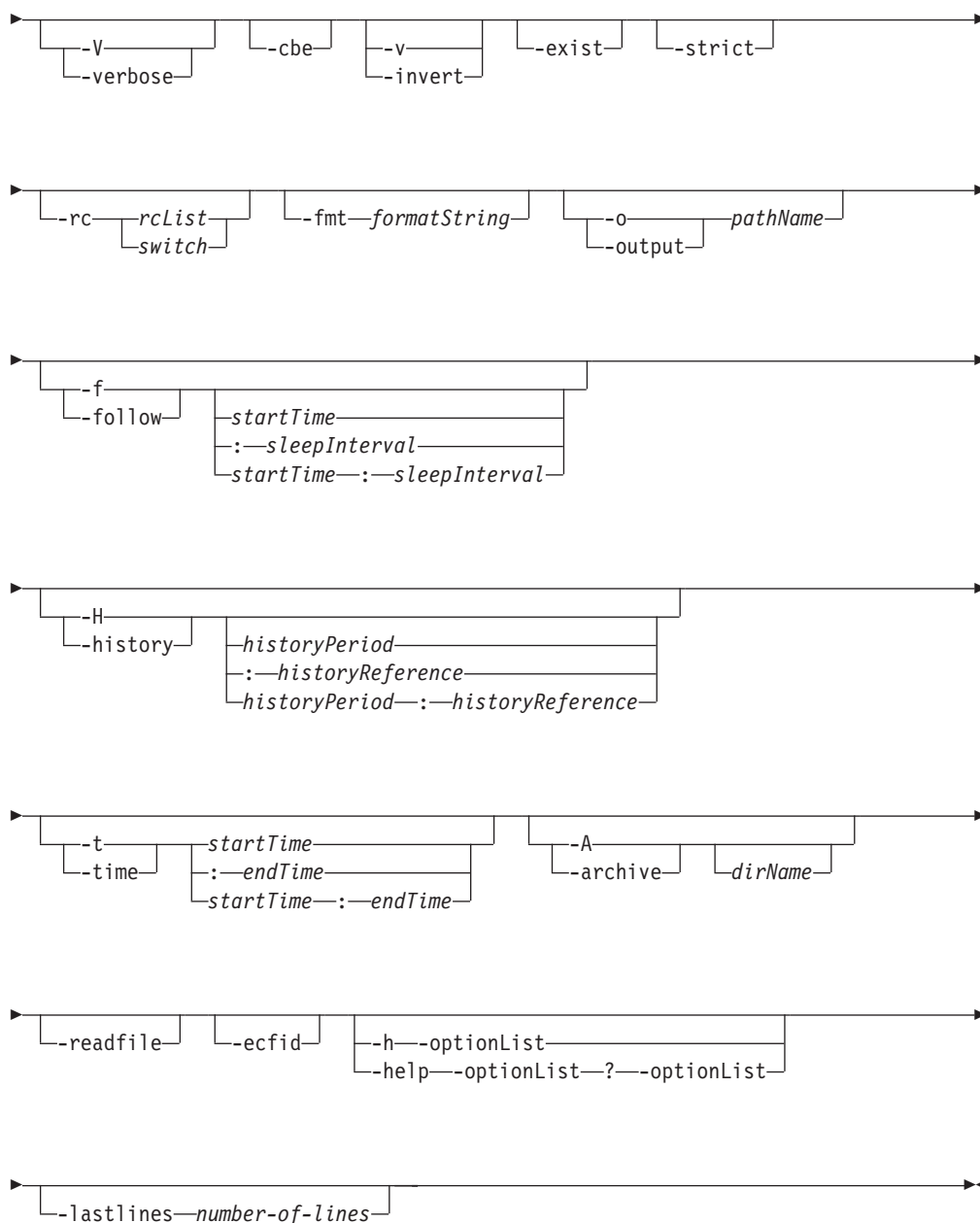
### Required connection

None

### Command syntax



## db2diag - db2diag logs analysis tool



### Command parameters

#### -global

Specifies that all the **db2diag** log files from all the database partitions on all the hosts are included in the log file processing.

**Note:** This command parameter is available in DB2 Version 9.8 Fix Pack 2 and later fix packs. This option supports rotating diagnostic log files and files located in split diagnostic data directories. In DB2 Version 9.8 Fix Pack 2, this option cannot be used in combination with the **-follow** option. In DB2 Version 9.8 Fix Pack 3 and later fix packs, this option can be used in combination with the **-follow** option.

**-merge** Merges diagnostic log files and sorts the records based on the timestamp. This option supports rotating diagnostic log files and files located in split diagnostic data directories.

If this command parameter is not followed by two or more space-separated *filename* values, the **db2diag** log files in the directory or directories specified by the **diagpath** database manager configuration parameter are merged. If the diagnostic data directory path is split across multiple database partitions, only the **db2diag** log files in the database partitions of the current host are merged.

If only one *filename* is specified, or only one diagnostic file exists in the path specified in the **diagpath** database manager configuration parameter, the single diagnostic log file is processed by the command as though the **-merge** command parameter was not specified.

**Note:** This command parameter is available in DB2 Version 9.7 Fix Pack 1 and later fix packs. It cannot be used in combination with **-facility**, **-follow**, or **-archive** options.

**-global -merge -sdir** *sharedDirPathname*

Specifying the **-global** and **-merge** options together results in all the **db2diag** log files from all the database partitions on all the hosts to be merged, and the records sorted based on the timestamp. This option supports rotating diagnostic log files and files located in split diagnostic data directories.

**Note:** The **-sdir** *sharedDirPathname* option must be specified to temporarily store the merged diagnostic log files obtained from the different hosts. The temporary merged diagnostic log files are deleted after processing has been completed. The *sharedDirPathname* must specify a shared directory to which all hosts have access and write permission.

*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, the directory or directories set by the **diagpath** database manager configuration parameter is searched.

**-facility | -fac**

Reads the files from the corresponding facility. A facility is a logical grouping of records. For example, all optimizer statistics records are grouped into the OPTSTATS facility. The output will be in text format by default. Valid facility options are the following values:

**ALL** Returns records from all facilities.

**MAIN** Returns records from DB2 general diagnostic logs, such as the **db2diag** log file, and rotating event logs.

**OPTSTATS**

Returns records related to optimizer statistics.

**-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

## db2diag - db2diag logs analysis tool

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:

### **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**

A unique alphanumeric identifier for a record, such as I11455A696.

### **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.

**eduid** EDU ID.

### **eduname**

EDU name.

### **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.

**probe** Probe number.

### **function**

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



**appid** The application ID. This value is the same as the **appl\_id** monitor element data. For detailed information about how to interpret this value, see “appl\_id - Application ID monitor element”.

**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.

**Event descriptions:**

**impact** User impact (for events only).

**startevent**  
Start event description (\*).

## db2diag - db2diag logs analysis tool

**stopevent**  
Stop event description (\*).

**changeevent**  
Change event description (\*).

**init** Initialization event description (\*).

**fini** Finish/finalize event description (\*).

**startup**  
Startup event description (\*).

**terminate**  
Terminate event description (\*).

**bringdown**  
Bringdown event description (\*).

**interrupt**  
Interrupt event description (\*).

**associate**  
Associate event description (\*).

**disassociate**  
Disassociate event description (\*).

**changecfg**  
Change configuration event description (\*).

**transfer**  
Transfer event description (\*).

**dispatch**  
Dispatch event description (\*).

**switch** Switch event description (\*).

**report** Report event description (\*).

**get** Get event description (\*).

**free** Free event description (\*).

**open** Open event description (\*).

**close** Close event description (\*).

**work** Work event description (\*).

**wait** Wait event description (\*).

**available**  
Available event description (\*).

**connect**  
Connect event description (\*).

**disconnect**  
Disconnect event description (\*).

**accept** Accept event description (\*).

**recv** Receive event description (\*).

**send** Send event description (\*).

**create** Create event description (\*).

**destroy**

Destroy event description (\*).

**request**

Request event description (\*).

**reply** Reply event description (\*).

**dependency**

Dependency event description (\*).

**write** Write event description (\*).

**read** Read event description (\*).

**reset** Reset event description (\*).

**collect**

Collect event description (\*).

**add** Add event description (\*).

**alter** Alter event description (\*).

**drop** Drop event description (\*).

**invalidate**

Invalidate event description (\*).

**grant** Grant event description (\*).

**revoke** Revoke event description (\*).

(\*) Each event field has the following subfields:

***{event}*type**

Event type (START, STOP, READ, WRITE, GET).

***{event}*desc**

Event description (header with event information).

***{event}*state**

Event state (success, failure, start, stop, in progress, idle) or event progress (in %).

***{event}*attr**

Event attributes (business level, cached, sync, async, internal, external, logical, physical, auto, manual, temporary, permanent).

***{event}*objid**

Unique object identifier (TABLE, CFG, DBM).

***{event}*objname**

Event object name (for example, "schema.tablename").

***{event}*objdata**

Object data (used if object is not a string or simple integer type, for example, data structure or some complex type).

***{event}*qtype**

Event qualifier type (FROM, TO, ON, FOR, AT, BY, CONTEXT).

## db2diag - db2diag logs analysis tool

### *{event}*qname

Event qualifier name/value (for example, FOR "DB ABC").

### *{event}*qdhdr

Event qualifier data header (contains type, text description and size of data). Used together with the *{event}*qdata field.

### *{event}*qdata

Event qualifier data (used if qualifier is not a string or simple integer type, for example, some data structure or complex type).

In the preceding list, keyword *{event}* should be substituted by event type for a specific event (for example, start, stop, change, read, write).

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 ( $1 \leq n \leq 64$ ). 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.

### **-filter** *fieldPatternList* | **-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 values:

- = 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.

- gv** *fieldPatternList*  
Searches for messages that do not match the specified pattern.
- gvi** | **-giv** *fieldPatternList*  
Same as **-gv**, but case-insensitive.
- lastrecords** *number-of-records*  
Displays and filters the last number of records specified from the db2diag log file. For each db2diag log file, this parameter checks whether the number of records specified are available. If the number of records available in the log file is less than the number of records you specified, the **db2diag** command processes all the records available in the file. If split diagnostic data directory paths are used, the last number of records specified is returned for each db2diag log file in each path.
- pid** *processIDList*  
Displays only log messages with the process IDs listed.
- tid** *threadIDList*  
Displays only log messages with the thread IDs listed.
- eduid** *EduIDList*  
Finds all records with a specified EDU ID from a list of EDU IDs containing one or more comma separated numeric values.
- 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*  
Finds all records with a specified severity level from a list of severity levels containing one or more comma separated text values, namely: Info, Warning, Error, Severe, Critical and Event.
- 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. This can be used for scripts to simplify parsing.
- 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 single or rotating **db2diag** log 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

## db2diag - db2diag logs analysis tool

as they become available. As of Version 9.8 Fix Pack 3, only the records from the last 8 kilobytes of the input file are processed.

The follow mode will also handle rotating **db2diag** log files. For example, the command will read the latest rotating diagnostic log file in use (db2diag.23.log) and follow along to the next created rotating log file (db2diag.24.log) when the db2diag.23.log log file meets its size limit.

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 closest time earlier than the specified time stamp 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:historyReference*

Specifies that logged messages are displayed that were recorded within the time period after the start time specified by *historyReference* (if an explicit positive value for *historyPeriod* is given), or logged messages are displayed that were recorded within the time period before the end time specified by *historyReference* (if a negative value for *historyPeriod* is given, or by default).

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*.

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 both single and rotating diagnostic log files. 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.

## db2diag - db2diag logs analysis tool

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 directories specified by the **diagpath** and **alt\_diagpath** configuration parameters are 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 directories specified by the **diagpath** and **alt\_diagpath** configuration parameters are searched for the **db2diag** log file. If found, it is archived in the directory specified by the **diagpath** or **alt\_diagpath** configuration parameter.

Starting with Version 9.7, Fixpack 4, the **db2diag -archive** option is available with IBM Data Server Driver Package and IBM Data Server for ODBC and CLI. This option enables you to archive the diagnostic log file on an instance-less client. For example:

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

### -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. Running the **db2diag** command using **rah** or **db2\_all** also requires the **-readfile** option to be 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 ecfld

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

### -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.

**-lastlines** *number-of-lines*

Displays and filters the last number of lines specified from the db2diag log file. For each db2diag log file, this parameter checks whether the number of lines specified are available. If the number of lines available in the log file is less than the number of records you specified, the db2diag command processes all the lines available in the file. If split diagnostic data directory paths are used, the last number of lines specified is returned for each db2diag log file in each path.

**Examples**

The following is a list of some examples which illustrate how to use the **db2diag** command under various circumstances:

- To merge all **db2diag** log files in the diagnostic data directory path, execute the following command:

```
db2diag -merge
```

If the diagnostic data directory path is split according to database partitions, this command merges the **db2diag** log files from all the database partitions of the current host. If the diagnostic data directory path is not split, the single diagnostic log file is processed by the command as though the **-merge** option was not specified.

- In this example, the default diagnostic data directory path was split according to physical host and database partition by setting the **diagpath** database manager configuration parameter using the following command:

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

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

```
- ~/sqllib/db2dump/HOST_bower/NODE0000/db2diag.log
- ~/sqllib/db2dump/HOST_bower/NODE0001/db2diag.log
- ~/sqllib/db2dump/HOST_bower/NODE0002/db2diag.log
- ~/sqllib/db2dump/HOST_horton/NODE0003/db2diag.log
- ~/sqllib/db2dump/HOST_horton/NODE0004/db2diag.log
- ~/sqllib/db2dump/HOST_horton/NODE0005/db2diag.log
```

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

```
db2diag -global
```

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

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

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

- To display all critical error messages, enter:

## db2diag - db2diag logs analysis tool

```
db2diag -level critical
```

or

```
db2diag -g 'level=Critical'
```

- To display all severe error messages produced by the process with the process ID (PID) 52356 and on database partition 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"
```

- 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 are 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/db2diag.\*.log files (Rotating logs if the database manager **diagsize** configuration parameter is set) from a directory specified by the **diagpath** configuration parameter.
- 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.

- To display only logged records containing eduid = 123, enter:  

```
db2diag -eduid 123
```
- To display all records containing eduid = 123 or eduid = 5678, enter:  

```
db2diag -eduid "123,5678"
```
- To display all severe error messages produced by a thread with eduid = 15, enter:  

```
db2diag -g "level=Severe, eduid=15"
```

or, which is equivalent,  

```
db2diag -g level=Severe | db2diag -eduid 15
```

- To display the last 5 formatted records from database partition 1, enter:  

```
db2diag -lastrecords 5 -node 1 -fmt "%{ts} %{node}"
```
- To read last 10 lines from all the db2diag.log files:  

```
db2diag -lastlines 10
```
- To merge the records in last 20 lines of each log file:  

```
db2diag -merge file1 file2 file3... -lastlines 20
```
- To display the records in last 20 lines of each db2diag.log file from all hosts:  

```
db2diag -global -lastlines 20
```
- To display all the records in last 100 lines which have Level=Error:  

```
db2diag -g level=Error -lastlines 100
```

### Usage notes

- 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**.
- 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** configuration parameter is searched next. If the **db2diag** log file is not found, **db2diag** returns an error and exits.
- 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.
- 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

## db2diag - db2diag logs analysis tool

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.

- The **-exist** option overrides 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.
- 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.
- The **-fmt** option overrides the **-strict,-cbe** and **-verbose** options.
- 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.
- It is recommended to specify the **-readfile** option when using **db2diag** in scripts. It will ensure reading from a file ignoring any terminal input.
- 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.
- Severe errors resulting from DB2 Text Search can be found logged in the **db2diag** log file.
- Be aware that using this tool to read and filter rotating **db2diag** log files (when the **diagsize** database configuration parameter is nonzero) will result in all the rotating diagnostic log files, to a series maximum of 10 files, to be read and filtered.
- Attempts to connect to the database while a restore is in progress will result in error messages in the **db2diag.log**. These error messages are ignorable if the restore command succeeds.

## db2drdat - DRDA trace

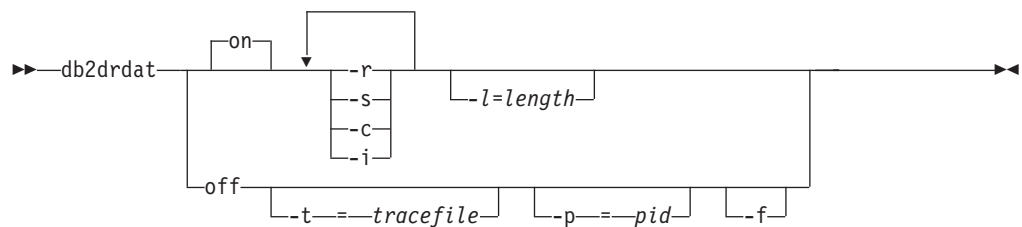
Allows the user to capture the DRDA data stream exchanged between a DRDA Application Requestor (AR) and the DB2 DRDA Application Server (AS).

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

## db2drdat - DRDA trace

- 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.



## db2drvmp - Map DB2 database drive

### 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.



---

## db2empfa - Enable multi-page file allocation

Enables the use of multi-page file allocation for a database. With multi-page 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 multi-page 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.

## db2envar.bat - Set environment of the current command window

Sets the environment of your current command window for the DB2 copy that **db2envar.bat** is executed from. This is useful if you want to switch between different DB2 copies from the command line.

This command is only available on Windows operating systems.

### Authorization

None

### Required Connection

None

### Command Syntax

▶▶—db2envar.bat—▶▶

### Command parameters

None

### Usage notes

When there are multiple DB2 copies on a machine, the full path should be used to indicate which **db2envar.bat** is to be executed. For example, if you want to set up the environment for the DB2 copy that is installed under e:\sql11ib, you should issue: \sql11ib\bin\db2envar.bat.

## 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**) in which case, all of the following authorization is required:

- CONNECT authority (or an authority that implicitly includes CONNECT)
- SELECT privilege on the following catalog tables (or an authority that implicitly includes SELECT on the catalog tables):
  - SYSIBM.SYSTABLES
  - SYSIBM.SYSEVENTMONITORS

If the event monitor is db2detaildeadlock, then one of the following authorities or privilege is additionally required:

- SYSMON
- SYSMOINT
- SYSCTRL
- SYSADM
- EXECUTE privilege on the SNAPSHOT\_DATABASE table function.
- DATAACCESS

### 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

**db2evmon** generates the same output regardless of whether the command is issued while connecting to the database or specifying the path option.

- If the instance is not already started when **db2evmon** is issued with the **-db** and **-evm** options, the command will start the instance.

## db2evmon - Event monitor productivity tool

- If the instance is not already started when **db2evmon** is issued with the **-path** option, the command will not start the instance. The instance needs to be started explicitly.

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.

## 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 schema-name] [-partitioned]
           |
           |
           |
▶ --evm event-monitor-name event type

```

### Command parameters

**-schema** *schema-name*

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** *event-monitor-name*

The name of the event monitor.

*event type*

Any of the event types available on the FOR clause of the CREATE EVENT MONITOR statement. The values possible are:

- ACTIVITIES
- BUFFERPOOLS
- CONNECTIONS
- CHANGE HISTORY
- DATABASE
- DEADLOCKS (also DEADLOCKS WITH DETAILS, DEADLOCKS WITH DETAILS HISTORY, DEADLOCKS WITH DETAILS HISTORY VALUES)\*
- LOCKING
- PACKAGE CACHE
- STATEMENTS
- STATISTICS
- TABLES
- TABLESPACES
- THRESHOLD VIOLATIONS
- TRANSACTIONS\*

## db2evtb1 - Generate event monitor target table definitions

- UNIT OF WORK

\* This event monitor type is deprecated.

### Examples

```
db2evtb1 -schema smith -evm test01 database, tables, tablespaces, bufferpools
```

### Usage notes

Output is written to standard output.

Defining WRITE TO TABLE event monitors is more straightforward when using the **db2evtb1** tool. For example, the following steps can be followed to define and activate an event monitor.

1. Use **db2evtb1** 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.

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.

When a LOCKING, UNIT OF WORK or PACKAGE CACHE event monitor is among the event monitors to be created, the DDL produced by **db2evtb1** creates regular, not UE tables.

## db2exfmt - Explain table format

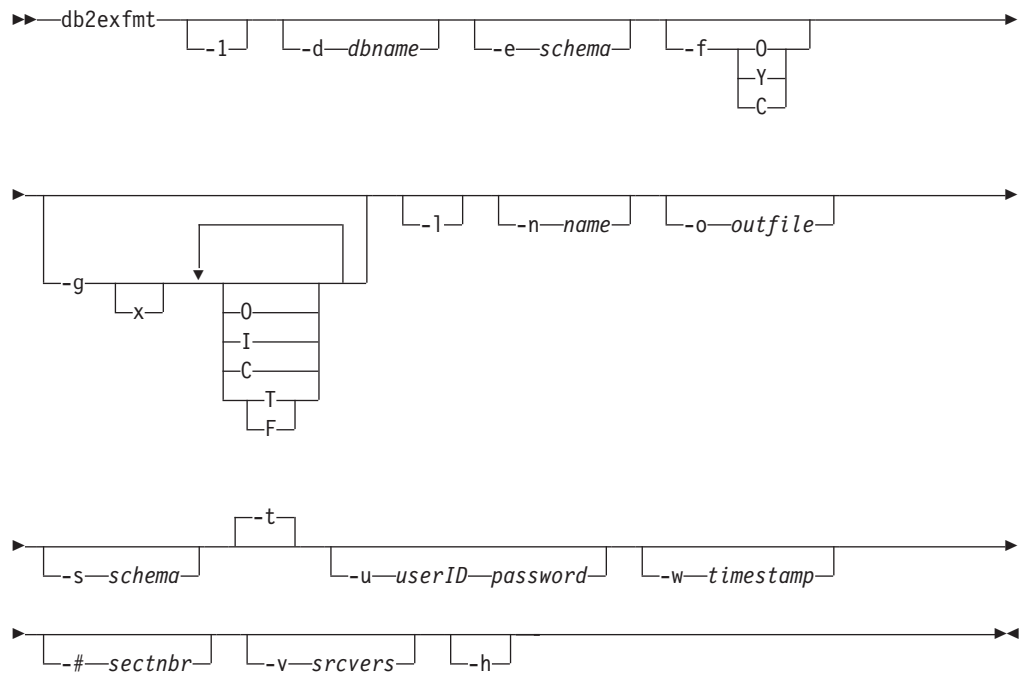
Formats the contents of the EXPLAIN tables.

This tool is located in the `misc` subdirectory of the instance `sqllib` directory. This tool uses the statistics from the EXPLAIN snapshot, if the snapshot is available.

### Authorization

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

### Command syntax



### Command parameters

#### db2exfmt

If no options are specified, then the command enters interactive mode and you will be prompted to make entries.

**-l** Use defaults `-e % -n % -s % -v % -w -l -# 0`

If Explain schema is not supplied, the contents of the environment variable **\$USER**, or **\$USERNAME** will be used as a default. If this variable is not found, the user will be prompted for an Explain schema.

**-d** *dbname*

Name of the database containing packages.

**-e** *schema*

Explain table SQL schema.

**-f**

Formatting flags. Multiple flags can be combined together as a string. For example, to achieve an output similar to the previous versions of DB2 database products, the C option and Y option can be combined as `-f CY`.

## db2exfmt - Explain table format

- O** Operator summary.
  - Y** Force formatting of the original statement even if column EXPLAIN\_STATEMENT.EXPLAIN\_TEXT contains formatting. The default behavior is to automatically detect if the statement requires formatting and use the original formatting when it exists.
  - C** Use a more compact mode when formatting statements and predicates. The default is an expanded mode that is easier to read. If Y is not specified then C only takes effect if automatic detection determines that the statement requires formatting.
- g** Graph plan.
- x** Turn OFF options (default is to turn them ON).

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.
- F** Include first tuple cost in 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.

Any combination of these options is allowed, except F and T, which are mutually exclusive.

- 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.
- 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.
- v srcvers**  
Source version of source of Explain request (default %)
- 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.

The **db2exfmt** command displays the statistics from the EXPLAIN snapshot, if the snapshot is available. Otherwise, **db2exfmt** displays statistics stored in the EXPLAIN\_OBJECT table and also displays some statistics retrieved directly from the system catalog.

The per-partition usage, transferrate and prefetchsize values returned are retrieved when the **db2exfmt** command is executed therefore their values can differ from the actual values used when the statement was explained.

The following are EXPLAIN snapshot examples.

```
db2 explain plan with snapshot for query
db2exfmt
```

or,

```
db2 set current explain mode yes
db2 set current explain snapshot yes
run the query
db2exfmt
```

### db2exmig - Migrate explain tables

Migrates explain tables. 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 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 **db2exmig** command will preserve any user added columns on the explain tables.

#### Authorization

If the **db2exmig** application package is bound then the required authorization is one of the following authorities:

- DBADM authority
- EXECUTE authority on the **db2exmig** application package and SELECT privilege or CONTROL privilege on the following system catalogs:
  - SYSCAT.COLUMNS
  - SYSCAT.TABLES
  - SYSCAT.REFERENCES

If the **db2exmig** application package is not bound then the required authorization is DBADM authority.

#### 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.

#### Usage notes

You can determine the **db2exmig** application package name by using the command: `db2bfd -b db2exmig.bnd`. The `db2exmig.bnd` files are located in the `sql1lib/bnd` folder.

## db2expln - SQL and XQuery Explain

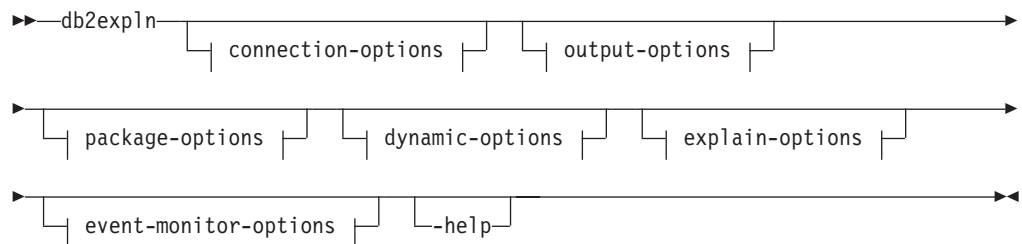
The **db2expln** tool describes the access plan selected for SQL and XQuery statements. Use the tool to obtain a quick explanation of the chosen access plan when explain data was not captured. For static SQL and XQuery statements, **db2expln** examines the packages stored in the system catalog tables. For dynamic SQL and XQuery statements, **db2expln** examines the query cache sections.

### Authorization

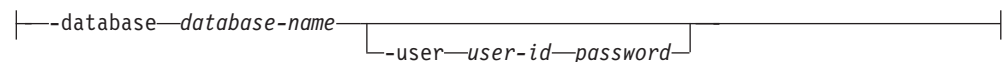
DBADM or one of the following authorizations or privileges:

- For static statements, SELECT privilege on the catalog tables
- For dynamic statements, SELECT privilege on the catalog tables plus one of the following authorizations or privileges:
  - Sufficient privileges to compile the statement
  - EXPLAIN authority
  - SQLADM authority

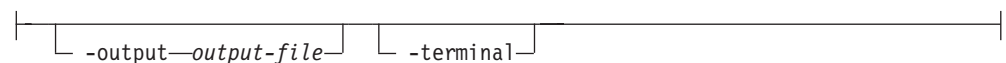
### Command syntax



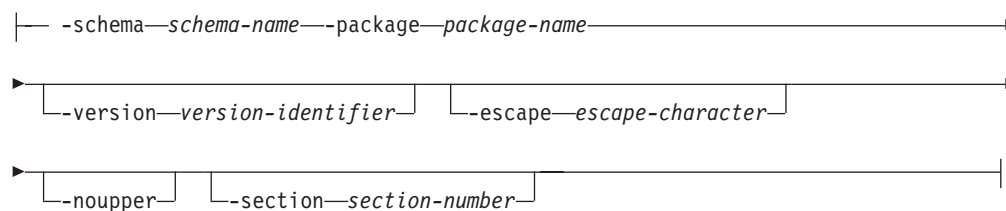
#### connection-options:



#### output-options:

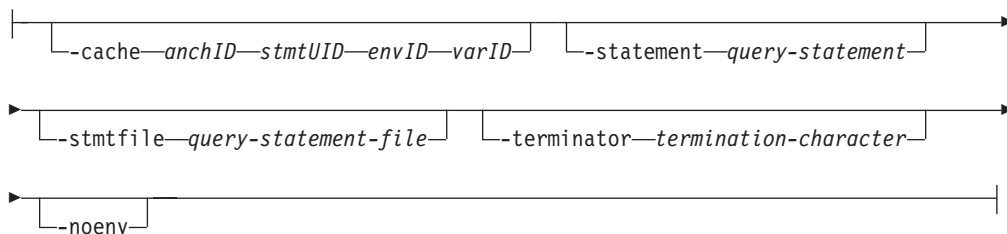


#### package-options:

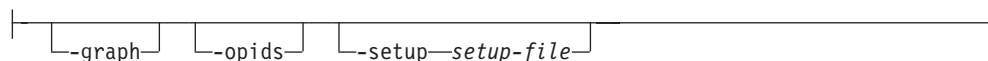


#### dynamic-options:

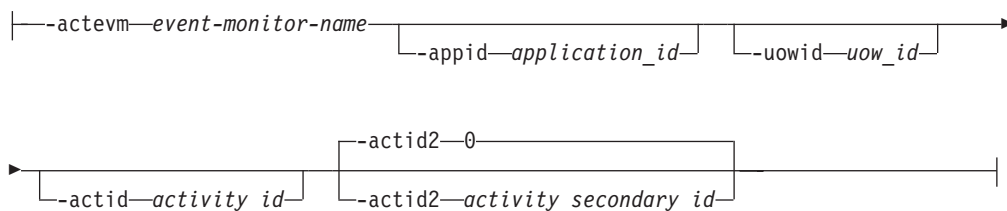
## db2expln - SQL and XQuery Explain



### explain-options:



### event-monitor-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% ....
```

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 uppercase before searching for matching packages.

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

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

## db2expln - SQL and XQuery Explain

### **-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.

### **-cache** *anchID, stmtUID, envID, varID*

Specifies the dynamic SQL cache from which the statement, identified by the given identifiers (IDs), is retrieved. The IDs can be obtained using the **db2pd** command with the **-dynamic** option.

### **-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.

### **-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.

### **-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.

### **-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 statement 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 then the original optimizer plan graph is constructed. 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 **-?**.

**-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 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.

### event-monitor-options:

These options specify one or more section environments from an activities event monitor to be explained.

**-actevm** *event-monitor-name*

Specifies the name of the activities event monitor whose activitystmt logical grouping contains the section environments (in the **section\_env** monitor element) to be explained.

**-appid** *application-id*

Specifies the application identifier (**appl\_id** monitor element) uniquely identifying the application that issued the activities whose section environments are to be explained. **-actevm** must be specified if **-appid** is specified.

## db2expln - SQL and XQuery Explain

- uowid** *uow-id*  
Specifies the unit of work ID (**uow\_id** monitor element) whose section environments are to be explained. The unit of work ID is unique only within a given application. **-actevm** must be specified if **-uowid** is specified.
- actid** *activity-id*  
Specifies the activity ID (**activity\_id** monitor element) whose section environments are to be explained. The activity ID is only unique within a given unit of work. **-actevm** must be specified if **-actid** is specified.
- actid2** *activity-secondary-id*  
Specifies the activity secondary ID (**activity\_secondary\_id** monitor element) whose section environments are to be explained. This defaults to zero if not specified. **-actevm** must be specified if **-actid2** is specified.

### 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 listed previously 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.

The following messages can be returned by **db2expln**:

- No packages found for database package pattern: "<creator>".<package> with version "<version>"  
This message will appear in the output if no packages were found in the database that matched the specified pattern.
- Bind messages can be found in db2expln.msg  
This message will appear in the output if the bind of db2expln.bnd was not successful. Further information about the problems encountered will be found in the db2expln.msg file in the current directory.
- Section number overridden to 0 (all sections) for potential multiple packages.  
This message will appear in the output if multiple packages might be encountered by **db2expln**. This action will be taken if one of the pattern matching characters is used in the package or creator input arguments.
- Bind messages for <bind file> can be found in <message file>  
This message will appear if the bind of the specified bind file was not successful. Further information about the problems encountered will be found in the specified message file on the database server.



- No static sections qualify from package.  
This message will appear in the output if the specified package only contains dynamic query statements, which means that there are no static sections.
- Package "<creator>".<package>", "<version>", is not valid. Rebind the package and then rerun db2expln.  
This message will appear in the output if the specified package is currently not valid. Reissue the BIND or REBIND command for the plan to re-create a valid package in the database, and then rerun **db2expln**.

The following statements will not be explained:

- BEGIN/END COMPOUND
- BEGIN/END DECLARE SECTION
- CLOSE cursor
- COMMIT and ROLLBACK
- CONNECT
- DESCRIBE
- Dynamic DECLARE CURSOR
- EXECUTE
- EXECUTE IMMEDIATE
- FETCH
- INCLUDE
- OPEN cursor
- PREPARE
- SQL control statements
- WHENEVER

Each sub-statement within a compound SQL statement might have its own section, which can be explained by **db2expln**.

**Note:** The **db2expln** command does not exclude any XQuery statements.

### 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
```

## db2expln - SQL and XQuery Explain

Explain the following statement:

```
SELECT e.lastname, e.job, d.deptname, d.location, p.projname
FROM employee AS e, department AS d, project AS p
WHERE e.workdept = d.deptno AND e.workdept = p.deptno
```

The following command:

```
db2expln -database SAMPLE
        -statement "SELECT e.lastname, e.job,
        d.deptname, d.location, p.projname
        FROM employee AS e, department AS d, project AS p
        WHERE e.workdept = d.deptno AND e.workdept = p.deptno"
        -terminal
```

returns:

```
DB2 Enterprise Server Edition n.n, nnnn-xxx (c) Copyright IBM Corp. 1991, yyyy
Licensed Material - Program Property of IBM
IBM DB2 Database SQL and XQUERY Explain Tool
```

```
***** DYNAMIC *****
```

```
===== STATEMENT =====
```

```
Isolation Level          = Cursor Stability
Blocking                  = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = No
Intra-Partition Parallel = No

SQL Path                  = "SYSIBM", "SYSFUN", "SYSPROC", "SYSIBMADM",
                          "SDINIRO"
```

Statement:

```
SELECT e.lastname, e.job, d.deptname, d.location, p.projname
FROM employee AS e, department AS d, project AS p
WHERE e.workdept =d.deptno AND e.workdept =p.deptno
```

Section Code Page = 1208

Estimated Cost = 22.802252

Estimated Cardinality = 105.000000

Access Table Name = SDINIRO.PROJECT ID = 2,10

```
#Columns = 2
Skip Inserted Rows
Avoid Locking Committed Data
Currently Committed for Cursor Stability
Relation Scan
| Prefetch: Eligible
Lock Intents
| Table: Intent Share
| Row : Next Key Share
Sargable Predicate(s)
| Process Build Table for Hash Join
```

Hash Join

```
Estimated Build Size: 4000
Estimated Probe Size: 4000
Access Table Name = SDINIRO.DEPARTMENT ID = 2,6
| #Columns = 3
| Skip Inserted Rows
| Avoid Locking Committed Data
| Currently Committed for Cursor Stability
```

```

|      |      | Relation Scan
|      |      |   Prefetch: Eligible
|      |      | Lock Intents
|      |      |   Table: Intent Share
|      |      |   Row  : Next Key Share
|      |      | Sargable Predicate(s)
|      |      |   Process Probe Table for Hash Join
Hash Join
|      |      | Estimated Build Size: 4000
|      |      | Estimated Probe Size: 4000
|      |      | Access Table Name = SDINIRO.EMPLOYEE ID = 2,7
|      |      |   #Columns = 3
|      |      |   Skip Inserted Rows
|      |      |   Avoid Locking Committed Data
|      |      |   Currently Committed for Cursor Stability
|      |      | Relation Scan
|      |      |   Prefetch: Eligible
|      |      | Lock Intents
|      |      |   Table: Intent Share
|      |      |   Row  : Next Key Share
|      |      | Sargable Predicate(s)
|      |      |   Process Probe Table for Hash Join
Return Data to Application
|      |      | #Columns = 5

End of section

```

## db2extsec - Set permissions for DB2 objects

Sets the permissions for DB2 database objects (for example, files, directories, network shares, registry keys and services) on updated DB2 database system installations.

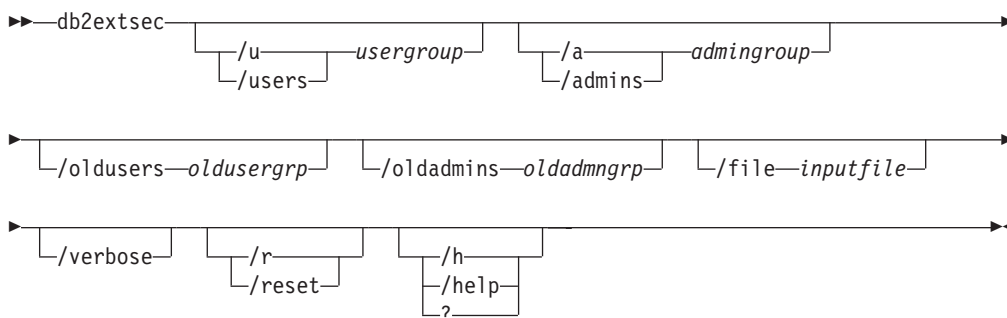
### Authorization

SYSADM

### Required connection

None

### Command syntax



### Command parameters

**/u** | **/users** *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. The *usergroup* can be a local group or a domain group. To specify a local group, you can specify the group name with or without the machine name. For example, DB2USERS, or MYWKSTN\DB2USERS. To specify a domain group, you specify the *usergroup* in the form of DOMAIN\GROUP. For example, MYDOMAIN\DB2USERS.

**/a** | **/admins** *admingroup*

Specifies the name of the administration group to be added. If this option is not specified, the default DB2 administration group (DB2ADMNS) is used. The *admingroup* can be a local group or a domain group. To specify a local group, you can specify the group name with or without the machine name. For example, DB2ADMNS, or MYWKSTN\DB2ADMNS. To specify a domain group, you specify the *admingroup* in the form of DOMAIN\GROUP. For example, MYDOMAIN\DB2ADMNS.

**Note:** The following 3 parameters, `/oldusers`, `/oldadmins`, and `/file`, are required when you are changing the extended security group names and have file or directory objects that have been created outside of the default locations (that is, the install directory or database directories). The **db2extsec** command can only change permissions to a known set of DB2 files. If the user had created private DB2 files with extended security, then the user will need to provide the locations of these file, so the **db2extsec** command can change the permissions on these files with the new extended security group names. The location of the files are to be supplied in the *inputfile* using the `/file` option.

## db2extsec - Set permissions for DB2 objects

**/oldusers** *oldusergrp*

The old DB2 users group name to be changed.

**/oldadmins** *oldadmngrp*

The old DB2 admins group name to be changed.

**/file** *inputfile*

File listing additional files/directories for which the permissions need to be updated.

**/verbose**

Output extra information.

**/r | /reset**

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.

**/h | /help | ?**

Displays the command help information.

### Examples

To enable extended security and use the domain groups mydom\db2users and mydom\db2admns to protect your DB2 objects:

```
db2extsec /u mydom\db2users /a mydom\db2admns
```

To reset extended security to its previous setting (see the preceding section on **/reset** option):

```
db2extsec /reset
```

To enable extended security, but also change the security group for the files/directories listed in c:\mylist.lst from local group db2admns and db2users to domain groups mydom\db2admns and mydom\db2users:

```
db2extsec /users mydom\db2users /admins mydom\db2admns /oldadmins db2admns  
/oldusers db2users /file c:\mylist.lst
```

**Note:** The format of the input file is as follows:

```
* This is a comment
```

```
D:\MYBACKUPDIR
```

```
D:\MYEXPORTDIR
```

```
D:\MYMISCFILE\myfile.dat
```

```
* This is another comment
```

```
E:\MYOTHERBACKUPDIR
```

```
E:\MYOTHEREXPORTDIR
```

```
* These are more comments
```

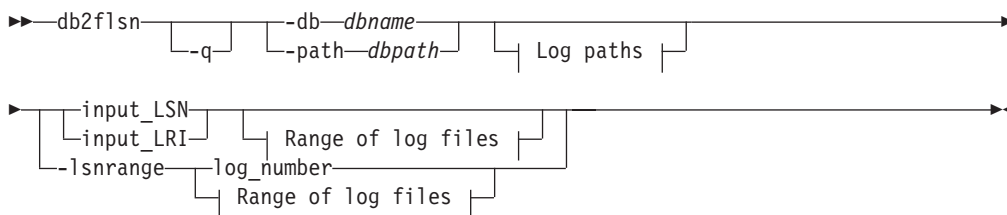
## db2flsn - Find log sequence number

The **db2flsn** command returns the name of the file in a log stream that contains the log record identified by a specified log sequence number (LSN).

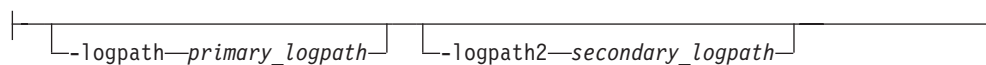
### Authorization

None

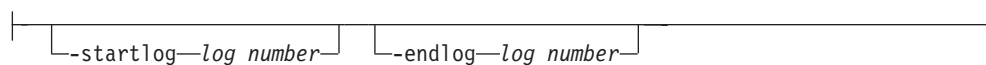
### Command syntax



### Log paths:



### Range of log files:



### Command parameters

**-q** Specifies that only the log file name be printed. No error or warning messages will be printed, and status can be determined only 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
- -106 Invalid database
- -108 The LSN or LRI is before the earliest log file **db2flsn** could open.
- -109 The LSN or LRI is before the log specified by **-startlog** parameter.
- -110 The LSN or LRI could not be found as no log files could be opened.
- -120 The LSN or LRI could not be found due to an error. Try using **-lsnrange** option.

Other valid return codes are:

- 0 Successful execution
- 100 Warning, the LSN or LRI could be in a later log file, returning the last file that **db2flsn** could open.
- 101 Warning, LSN or LRI could be in a later log file, returning the file specified by **-endlog** parameter. To find the exact file, use a larger value for the **-endlog** parameter.

**-db** *dbname*

Specifies the database name which you want to investigate.

**-path** *dbpath*

Specifies the full path to the directory where the LFH files, SQLLOGCTL.LFH.1 and its mirror copy SQLLOGCTL.LFH.2, reside.

*input\_LSN*

Specifies a 16 character string that represents an 8 byte hexadecimal value with leading zeros.

*input\_LRI*

Specifies a 34 character string that represents a 17 byte hexadecimal value with leading zeros.

**-lsnrange**

Specifies a log number or a range of log file numbers. The LSN value range is returned for each of the log numbers provided. For this option, log files must be present and readable.

*log\_number*

Specifies a number that represents a log file number, for example, 52 is the *log\_number* for log file S0000052.LOG.

**-logpath** *primary\_logpath*

Specifies the primary log path. If not provided, the database primary log path is used.

**-logpath2** *secondary\_logpath2*

Specifies the secondary or temporary log path. To use log files from archive, use a temporary directory for retrieval. That temporary path can then be as **-logpath2** parameter, so that **db2flsn** can read the specified logs. If **logpath2** is not provided, the database mirror log path is used when available.

**-startlog** *log\_number*

If *log\_number* is provided, **db2flsn** does not search for log files before *log\_number*.

**-endlog** *log\_number*

If *log\_number* is provided, **db2flsn** does not search for log files after *log\_number*.

## Examples

Finding an LSN:

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

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

```
db2flsn -db flsntest 0000000000FA0000
Given LSN is contained in log file S0000002.LOG
```

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

```
db2flsn -path /db2/NODE0000/SQL00001 0000000000FA4368
Given LSN is contained in log file S0000002.LOG
```

**Using -lsnrange option:**

```
db2flsn -lsnrange -startlog 20 -endlog 27
S0000020.LOG: has LSN range 000000000023B3D to 000000000023E0A
```

## db2flsn - Find log sequence number

S0000021.LOG: has LSN range 000000000023E0B to 0000000000240D8  
S0000022.LOG: is not empty, but has no log records starting within its boundaries.  
S0000023.LOG: has LSN range 0000000000240D9 to 0000000000243B9  
S0000024.LOG: is unused/blank.  
S0000025.LOG: is unused/blank.  
S0000026.LOG: could not be opened or was invalid. Check file permissions.  
S0000027.LOG: could not be opened or was invalid. Check file permissions.

**Warning messages when an LSN could not be found, but it cannot be in an earlier log file:**

**db2flsn 0000000000243D7**

Input LSN 0000000000243D7 is within or after log file S0000023.LOG:

S0000023.LOG: Log file starts with LSN 0000000000243BA.

S0000024.LOG: Log file could not be opened or was invalid/unused.

Log paths used:  
(./LOGSTREAM0000) and  
().

If this is not the end of the log stream, retrieve more log files into a temporary directory (to avoid overwriting active log files) and re-run db2flsn tool with -logpath2 option to use the log files from that temporary directory as well. Check access permissions on log files to make sure db2flsn can open necessary log files.

To see the first LSN of a log file, use the -lsnrange option:  
db2flsn -lsnrange <log file number> -logpath2 <temp log dir>

To see the first LSN of a set of log files, use:  
db2flsn -lsnrange -logpath2 <temp log dir> \  
-startlog <first log> -endlog <end log>

**db2flsn -q 0000000000243D7**

S0000023.LOG

=> **Return code is: 100 (Warning, the LSN could be in a later log file, returning the last file that db2flsn could open).**

**db2flsn 0000000000243D7 -endlog 23**

Input LSN 0000000000243D7 is after log file S0000023.LOG.

This log file has LSN range 0000000000243BA - 0000000000243D6.

To find the exact log file, try a larger value for the -endlog parameter.

Log paths used:  
(./LOGSTREAM0000) and  
().

To see the first LSN of a log file, use the -lsnrange option:  
db2flsn -lsnrange <log file number> -logpath2 <temp log dir>

To see the first LSN of a set of log files, use:  
db2flsn -lsnrange -logpath2 <temp log dir> \  
-startlog <first log> -endlog <end log>

**db2flsn -q 0000000000243D7 -endlog 23**

S0000023.LOG

=> **Return code is: 101 (Warning, LSN could be in a later log file, returning the file specified by -endlog parameter. To find the exact file, use a larger value for the -endlog parameter.)**

**db2flsn -lsnrange 23**

S0000023.LOG: has LSN range 0000000000243BA to 0000000000243D6

=> This shows that LSN 0000000000243D7 has not been produced yet on this log stream. However, when it is generated, this LSN is guaranteed that it cannot be in a log file before S0000023.LOG.



## db2flsn - Find log sequence number

Error messages, when an LSN could not be found. In this example, S0000000.LOG has already been archived and not found in history file either:

```
db2flsn 0000000000000001
```

```
Input LSN 0000000000000001 could not be found. Input LSN is before the earliest LSN found 0000000000021E6B:
```

```
S0000001.LOG: Log file starts with LSN 0000000000021E6B.
```

```
S0000000.LOG: Log file could not be opened or was invalid/unused.
```

```
Log paths used:  
(./LOGSTREAM0000) and  
().
```

Retrieve log files before S0000001.LOG into a temporary directory (to avoid overwriting active log files) and re-run db2flsn tool with -logpath2 option to include the log files from that temporary directory:

```
db2flsn <LSN> -db <db> -logpath2 <temp log dir> -startlog <first log>  
-endlog <last log>
```

To see the first LSN of a log file, use the -lsnrange option:

```
db2flsn -lsnrange <log file number> -logpath2 <temp log dir>
```

To see the first LSN of a set of log files, use:

```
db2flsn -lsnrange -logpath2 <temp log dir> \  
-startlog <first log> -endlog <eng log>
```

```
db2flsn -q 0000000000000001
```

=> with "-q" option, no output is produced as the log file could not be found. Error code returned is -108 (The LSN is before the earliest log file db2flsn could open).

```
db2flsn 0000000000000001 -startlog 1
```

```
Input LSN(0000000000000001) is before log file S0000001.LOG. This log file starts with LSN 0000000000021E6B.
```

```
Log paths used:  
(./LOGSTREAM0000) and  
().
```

To see the first LSN of a log file, use the -lsnrange option:

```
db2flsn -lsnrange <log file number> -logpath2 <temp log dir>
```

To see the first LSN of a set of log files, use:

```
db2flsn -lsnrange -logpath2 <temp log dir> \  
-startlog <first log> -endlog <end log>
```

```
db2flsn -q 0000000000000001 -startlog 1
```

=> with "-q" option, no output is produced as the log file could not be found. Error code returned is -109 (The LSN is before the log specified by -startlog parameter.)

### Usage notes

- If **-db** and **-path** are not specified, **db2flsn** takes the current directory as database path where the LFH files (SQLOGCTL.LFH.1 and SQLOGCTL.LFH.2) are located. **db2flsn** also requires access to the GLFH files (SQLOGCTL.GLFH.1 and SQLOGCTL.GLFH.2), which are opened either from the path specified by the **-path** parameter, or from their default location for the given database.
- The **db2flsn** tool tries to open the history file from either the directory provided by **-path** parameter or the default path where a history file is located for the given database. The history file contains the starting LSN for log files for

## db2flsn - Find log sequence number

databases using log archiving. In such cases, **db2flsn** looks in the history file when it cannot open an already archived log file. A database is using log archiving if it is configured with the **logarchmeth1** or **logarchmeth2** configuration parameters set to a value other than OFF or LOGRETAIN.

- This tool works with both recoverable and non-recoverable (circular logging) databases. A database is recoverable if it is configured with the **logarchmeth1** or **logarchmeth2** configuration parameters set to a value other than OFF.

## db2fm - DB2 fault monitor

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

This command is available on UNIX and Linux operating systems.

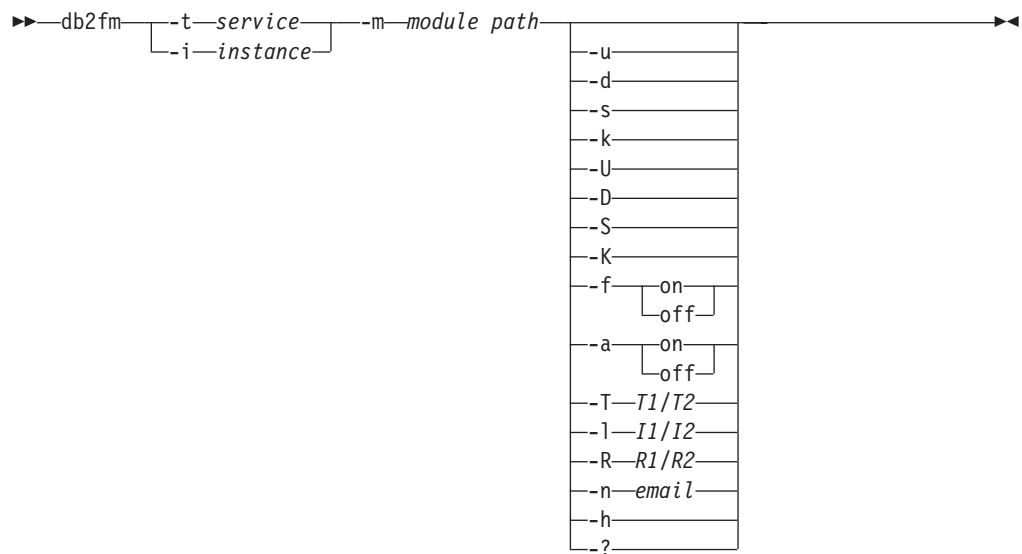
### Authorization

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.

## db2fm - DB2 fault monitor

- 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 deactivates fault monitoring. If this option is set to 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
  - **-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.
- R R1/R2**  
Sets the number of retries for the status method and action before giving up.
- n e-mail**  
Sets the email address for notification of events.
- h | -?**  
Displays command usage help.

### Usage notes

- If you want to set up email notifications to be sent by the fault monitor daemon, you must have SMTP running on the local host.

## db2fmcu - DB2 fault monitor controller

DB2 Fault Monitor is the DB2 database facility that automatically starts an instance after a crash. It can also auto restart an instance on machine reboot. You can configure the DB2 fault monitor on Linux and UNIX operating systems using the DB2 fault monitor controller command. The command must be run as root because it accesses the system's `inittab` file.

### Authorization

Root user authority

### Required connection

None

### Command syntax

```

db2fmcu -u -p db2fmc_d_path -d -f inittab

```

### Command parameters

**-u -p** *db2fmc\_d\_path*

This option re-configures the `inittab` file to include the fault monitor controller (FMC) at system startup, where *db2fmc\_d\_path* is the complete path to the FMC daemon (**db2fmc\_d**) object, for example `/opt/IBM/db2/bin/db2fmc_d`. On Linux systems where `/etc/inittab` has been deprecated (for example, Red Hat Enterprise Linux 6), specifying this option creates or replaces the `db2fmc_d.conf` file under the `/etc/init` directory.

**-d** This option changes the `inittab` file configuration to prevent the FMC from being run at system startup. On Linux systems where `/etc/inittab` has been deprecated (for example, Red Hat Enterprise Linux 6), specifying this option removes the `db2fmc_d.conf` file.

**-f** *inittab*

This option specifies a path to the `inittab` file.

### Example

To start the fault monitor controller at system startup by re-configuring the `inittab` file, run the following command:

```
db2fmcu -u -p /opt/IBM/db2/bin/db2fmc_d
```

To prevent the fault monitor controller from being launched at system startup, run the following command:

```
db2fmcu -d
```

### Usage notes

- If you changed `/etc/inittab` manually, you need to send `SIGHUP` to process 1 to ask it to re-scan `/etc/inittab` right away. Otherwise, it can take some time before the next re-scan happens. If you updated `/etc/inittab` via **db2fmcu**, you do not need to send the signal as it is already done by the **db2fmcu** command.

## db2fmcu - DB2 fault monitor controller

- In the Solaris Operating System, the processes listed in the `/etc/inittab` file inherit the default system project settings. Ensure that the kernel parameters for the project are set high enough to accommodate the DB2 processes or start the **db2fmc**d process under a different project that has the appropriate kernel settings. To start the **db2fmc**d process under a different project, edit the `fmc` entry in the `/etc/inittab` file to use `newtask`. For example, add `/usr/bin/newtask -p` in front of the **db2fmc**d path. For more information about setting kernel parameters, see the **db2osconf** command.

## db2fodc - DB2 first occurrence data collection

The **db2fodc** utility captures symptom-based data about the DB2 instance to help in problem determination situations. It is intended to collect information about potential hangs, severe performance issues, and various types of errors.

### Purpose

The **db2fodc** command can be used for manual first occurrence data collection (FODC) on problems that cannot trigger automatic FODC, such as hangs or severe performance problems. It also can be used to collect data about index errors. The **db2fodc** tool captures data, to be included in the FODC package, and places it inside an FODC package directory, created either in the default diagnostic path or in an FODC directory path you specify using the **-fodcpath** parameter.

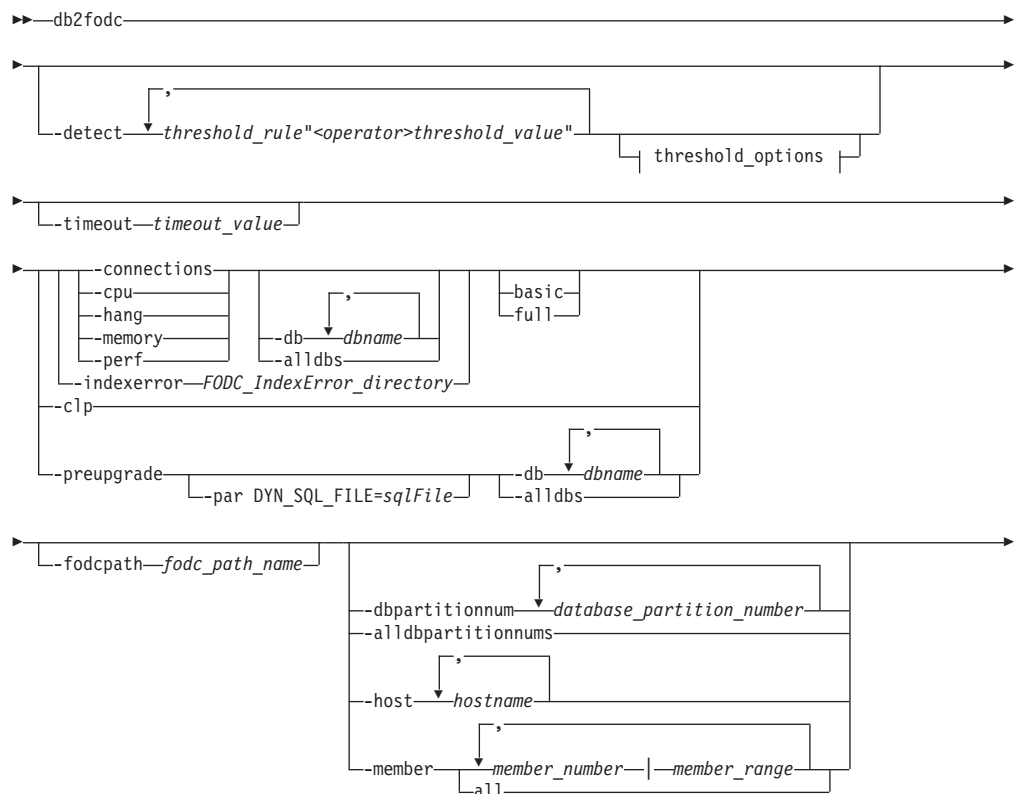
The **db2fodc** tool supports additional manual collection types and supports triggering automatic diagnostic data collection when a user-defined threshold condition is exceeded.

### Authorization

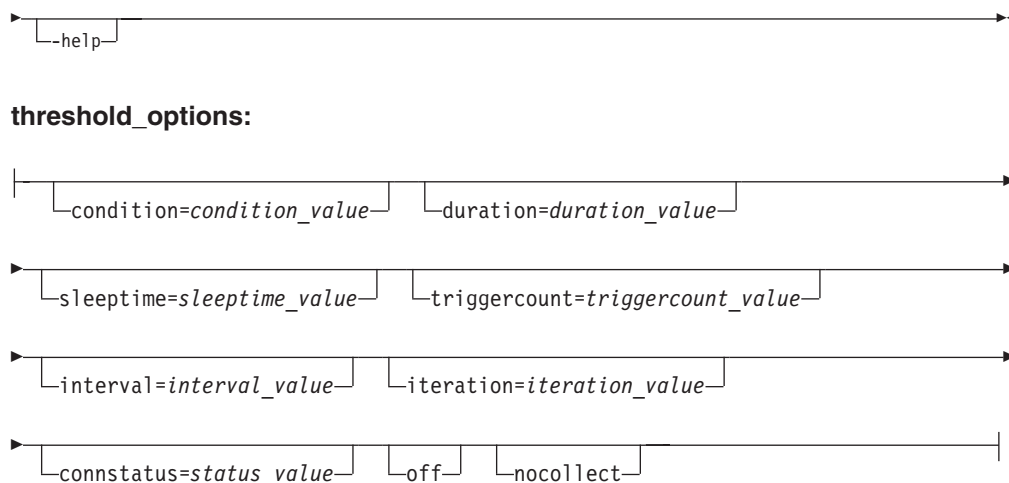
One of the following authorities:

- On Linux and UNIX systems, the *sysadm* authority level. You must also be the instance owner.
- On Windows operating systems, the *sysadm* authority level.

### Command syntax



## db2fodc - DB2 first occurrence data collection



### Command parameters

#### -detect

Detects a trigger condition based on one or several threshold rules you specify. At least one valid threshold rule must be specified. Detection is logged in the db2diag log files. If the trigger condition is detected the number of times specified by the triggercount option, diagnostic data collection is performed.

#### *threshold\_rule*

One of the supported threshold rule names, indicating the condition the threshold is to detect. Available rules cover processor usage, memory usage, and database connections. You can specify multiple threshold rules. For more information, see Supported threshold detection rules.

#### *<operator>*

One of the supported operators, either >= (greater than or equal to) or <= (less than or equal to).

#### *threshold\_value*

A numeric value for the threshold rule specified. Only non-negative integers can be specified.

#### **condition=condition\_value**

A logical operator for ANDing or ORing threshold rules when multiple threshold rules are specified. Either and or or can be specified. The default is to AND threshold rules.

#### **duration=duration\_value**

Specifies the length of time, in hours, during which threshold detection and diagnostic data collection are enabled.

#### **sleeptime=sleeptime\_value**

Specifies the time to wait, in seconds, before starting the next iteration of threshold detection and diagnostic data collection. The default is 1 second.

#### **triggercount=triggercount\_value**

Specifies the consecutive number of times the threshold condition must be detected before diagnostic data collection is triggered. The default is 5 times. After each detection of the trigger condition, detection sleeps for the interval specified before resuming.



**interval=***interval\_value*

Specifies the time to wait, in seconds, before resuming the detection of threshold condition within the current iteration. The default is 1 second.

**iteration=***iteration\_value*

Specifies the maximum number of iterations to perform for threshold detection and diagnostic data collection. The default is 1 iteration.

**connstatus=***status\_value*

Specifies the connection status for the connection. The default is to count all connection statuses for applications connected to the database. Valid connection status keywords include CommitActive, Compiling, ConnectCompleted, ConnectPending, CreatingDatabase, Decoupled, DisconnectPending, FederatedRequestPending, Heuristical-lyCommitted, HeuristicallyAborted, Lock-wait, PendingRemoteRequest, Performin-gLoad, PerformingBackup, PerformingUnload, QuiescingTable-space, Re-questInterrupted, Recompiling, RestartingDatabase, RestoringDatabase, RollbackActive, RollbackToSavepoint, TransactionEnded, TransactionPrepared, UOW-Executing, UOW-Waiting, UOWQueued, Unknown, Wait-Autonomous, and WaitToDisableTablespace

**off**

Stops all threshold detection and turns off currently active threshold rules. If other options are also specified when off is specified, the other options are ignored. Turning off threshold detection requires up to 60 seconds to take full effect.

**nocollect**

Specifies to only detect when threshold conditions are met, but diagnostic data collection is not performed. Threshold detection is logged in the db2diag log files.

**-c1p**

Collects environment and configuration related information, used to troubleshoot problems related to instance creation. The **db2fodc -c1p** command does not support the **-member** parameter but it does support the **-host** parameter. The collected information is stored in a new directory named `FODC_C1p_timestamp_member`.

**Note:** The **-c1p** parameter is supported only on Linux and UNIX operating systems. If you issue this command on a Windows operating system, no data will be collected.

**-connections**

Collects connection-related diagnostic data, used to diagnose problems such as sudden spikes in the number of applications in the executing or compiling state or new database connections being denied. The collected information is stored in a new directory named `FODC_Connections_timestamp_member`.

**-cpu**

Collects processor-related performance and diagnostic data, used to diagnose problems such as high processor utilization rates, a high number of running processes, or high processor wait times. The collected information is stored in a new directory named `FODC_Cpu_timestamp_member`.

**-hang**

Collects FODC data related to a potential hang situation or a serious performance issue. The FODC package is prepared as a result of running this

## db2fodc - DB2 first occurrence data collection

option. When this parameter is used, the instance is considered unusable and needs restarting. Data collection is performed as quickly as possible gathering as much information as possible.

### -indexerror *FODC\_IndexError\_directory*

Collects data related to an index error. *FODC\_IndexError\_directory* is required and must contain the *db2cos\_indexerror\_short(.bat)* script or the *db2cos\_indexerror\_long(.bat)* script. Ensure that no **db2dart** reports exist in the same *FODC\_IndexError\_directory*. If the *db2cos\_indexerror\_long(.bat)* script contains the **db2dart /t** command, the full collection mode requires that the database the **db2dart /t** command is run against be offline.

### -memory

Collects memory-related diagnostic data, used to diagnose problems such as no free memory available, swap space being used at a high rate, excessive paging or a suspected a memory leak. The collected information is stored in a new directory named *FODC\_Memory\_timestamp\_member*.

### -perf

Collects data related to a performance issue. This option should be used when the instance is still usable and restarting it is not necessary. This option should affect the system less than the **-hang** parameter. The full collection mode will be run, without user interaction and includes output from the **db2trc** command and additional snapshots. The FODC package is prepared as a result of running this option.

### -preupgrade

Collects performance related information before a critical upgrade or update such as upgrading an instance or updating to the next fix pack. This helps with troubleshooting any problems that might occur after the upgrade or update. This parameter must be specified with a database. After the collection of data is completed, the results are placed in a directory named *FODC\_Preupgrade\_timestamp\_member*. To obtain sufficient performance data to troubleshoot any future problems, you might need to issue this command several times, both at peak and idle usage times.

**Note:** This command can take a long time to complete as it collects performance data.

### Options:

#### basic

The basic collection mode will be run, without user interaction.

#### full

The full collection mode will be run, without user interaction. This option requires significantly more resources and time to run than basic collection mode.

#### -alldbpartitionnums

Specifies that this command is to run on all active database partition servers in the instance. **db2fodc** will report information from database partition servers on the same physical machine that **db2fodc** is being run on.

#### -alldbs

Collects FODC data related to all active databases. This option is active by default.

## db2fodc - DB2 first occurrence data collection

### **-db** *dbname*

Collects FODC data related to a specified database or databases. For example:

```
db2fodc -hang -db sample,dbsample
```

### **-dbpartitionnum** *database\_partition\_number*

Collects FODC data related to all the specified database partition numbers.

### **-fodcpath** *fodc\_path\_name*

Specifies the full path to the directory where the FODC data package is created. The path you specify must be writable by the members on the machine and by the fmp processes running on the member or partition.

### **-help**

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

### **-host** *hostname*

Specifies the host or hosts on which the command is issued. The command is issued for all members that reside on the host. If a host name is not specified, the command is issued on the local host for the default member. If multiple hosts are specified, all host names must be valid for the command to complete. This option cannot be combined with the **-member** option.

### **-member** *member\_number* | *member\_range*

Specifies the member or members on which the command is issued. If this option is not specified, the command is issued on the current member. Multiple members can be specified as a comma separated list of *member\_number* (member1, member2), or using *member\_range*, where *member\_range* is a range of members (member1-member3), or using any combination of the first two methods. This option cannot be combined with the **-host** option.

### **all**

Specifies that the command is issued on all members defined in `db2nodes.cfg`

### **-par DYN\_SQL\_FILE=***sqlFile*

Collects FODC data related to an SQL file that is specified by the user, available with the **-preupgrade** parameter only.

### **-timeout** *timeout\_value*

Specifies a timeout period for the call-out script invoked by the **db2fodc** command. If the timeout is reached before the call-out script completes diagnostic data collection, then the call-out script process is killed. There is no default timeout. The timeout is specified as *nh ym xs*, where *n* represents hours, *y* represents minutes, and *x* represents seconds. If no h, m, or n suffix is specified, the timeout is considered in seconds. Examples of valid timeout periods include **-timeout 2h 30m 45s** and **-timeout 600**.

**Note:** The **-timeout** option is not supported on Windows operating systems.

## Supported threshold detection rules

The following processor usage and memory threshold rules are supported:

Table 49. Processor usage and memory rules for different operating systems

Operating system	AIX <sup>1</sup>	Linux <sup>2</sup>	HP-UX <sup>3</sup>	Solaris <sup>4</sup>	Windows
Run queue (rqueue)	r	r	r	r	r
Block queue (bqueue)	b	b	b	b	Not applicable
Active memory (avm)	avm	active	avm	Not applicable	used
Free memory (free)	fre	free	free	free	free
Paging in (pi)	pi	Not applicable	pi	pi	pi
Paging out (po)	po	Not applicable	po	po	po
Swapping in (si)	Not applicable	si	si	si	Not applicable
Swapping out (so)	Not applicable	so	so	so	Not applicable
Page scanned (sr)	sr	Not applicable	sr	sr	Not applicable
User CPU (us)	us	us	us	us	usr
System CPU (sy)	sy	sy	sy	sy	sys
User and system CPU (us_sy)	us+sy	us+sy	us+sy	us+sy	us+sy
Idle CPU (id)	id	id	id	id	idl
Context switches (cs)	cs	cs	cs	cs	cs/s
Command used	<b>vmstat</b>	<b>vmstst -a</b>	<b>vmstat -S</b>	<b>vmstat -S</b>	<b>db2pd -vmstat</b>

Table 49. Processor usage and memory rules for different operating systems (continued)

Operating system	AIX <sup>1</sup>	Linux <sup>2</sup>	HP-UX <sup>3</sup>	Solaris <sup>4</sup>	Windows
<b>Note:</b>					
1. For AIX operating systems:					
<ul style="list-style-type: none"> <li>• Memory: Information about the usage of virtual and real memory. Virtual pages are considered active if they have been accessed. A large portion of real memory is utilized as a cache for file system data. It is not unusual for the size of the free list to remain small. A page is 4096 bytes.</li> <li>• pi: Pages paged in from paging space.</li> <li>• po: Pages paged out to paging space.</li> <li>• sr: Pages scanned by page replacement algorithm.</li> <li>• avm: Active virtual pages.</li> <li>• fre: Size of the free list.</li> </ul>					
2. For Linux operating systems:					
<ul style="list-style-type: none"> <li>• Memory: All linux blocks are currently 1024 bytes.</li> <li>• si: Amount of memory swapped in from disk per second.</li> <li>• so: Amount of memory swapped to disk per second.</li> <li>• inact: The amount of inactive memory.</li> <li>• active: The amount of active memory.</li> </ul>					
3. For HP-UX operating systems:					
<ul style="list-style-type: none"> <li>• Memory: Information about the usage of virtual and real memory. Virtual pages are considered active if they belong to processes that are running or have run in the last 20 seconds.</li> <li>• si: Processes swapped in.</li> <li>• so: Processes swapped out.</li> <li>• pi: Pages paged in.</li> <li>• po: Pages paged out.</li> <li>• sr: Pages scanned by clock algorithm per second.</li> <li>• avm: Active virtual pages.</li> <li>• free: Size of the free list.</li> </ul>					
4. For Solaris operating systems:					
<ul style="list-style-type: none"> <li>• si: Reports swap ins measured in Kbps.</li> <li>• so: Reports swap outs measured in Kbps.</li> <li>• pi: The amount of Kb paged in.</li> <li>• po: The amount of Kb paged out.</li> <li>• sr: Pages scanned by clock algorithm.</li> <li>• swap: amount of swap space currently available measured in Kb.</li> <li>• free: Size of the free list.</li> </ul>					

Table 50. Other memory rules for different operating systems

Operating system	Used swap space (swapused)	Command used
AIX	Percentage value located under the Percent Used column	<b>lsps -s</b>
Linux	(total swap space/used swap space)*100%	<b>free</b>

## db2fodc - DB2 first occurrence data collection

Table 50. Other memory rules for different operating systems (continued)

Operating system	Used swap space (swapused)	Command used
HP-UX	Total percentage value located under the PCT USED column	<b>swapinfo -tam</b>
Solaris	(used swap space/(available swap space+used swap space))*100%	<b>swap -s</b>
Windows	Not applicable	Not applicable

Table 51. Connection rules within a database scope for different operating systems

Operating system	Connections	Command used
AIX	Number of connected applications in status specified by the <b>connstatus</b> option.	<b>db2pd -application</b>
Linux	Number of connected applications in status specified by the <b>connstatus</b> option.	<b>db2pd -application</b>
HP-UX	Number of connected applications in status specified by the <b>connstatus</b> option.	<b>db2pd -application</b>
Solaris	Number of connected applications in status specified by the <b>connstatus</b> option.	<b>db2pd -application</b>
Windows	Number of connected applications in status specified by the <b>connstatus</b> option.	<b>db2pd -application</b>

### Examples

To collect data during a potential hang without stopping the database manager:

```
db2fodc -hang -alldbs
```

Default DB2FODC registry variables and parameters are used. A new directory prefixed with FODC\_Hang\_ is created under the current diagnostic path (an error is generated if it already exists). db2cos\_hang script is executed to collect manual FODC data into one or more files, deposited into the directory.

To collect data from a specific database:

```
db2fodc -db SAMPLE -hang
```

Data collection is restricted to database SAMPLE. A new directory prefixed with FODC\_Hang\_ is automatically created under the current diagnostic path. The db2cos\_hang script is executed to collect manual FODC data into the FODC package stored in the directory.

## db2fodc - DB2 first occurrence data collection

To collect data during a performance issue from a specific database using the full collection script:

```
db2fodc -db SAMPLE -perf full
```

Data collection is restricted to database SAMPLE. A new directory prefixed with FODC\_Perf\_ is created under the current diagnostic path. The db2cos\_perf script is executed to collect manual FODC data into one or more files, deposited into the directory.

To collect data about index errors with default (basic) mode without stopping the database manager:

```
db2fodc -indexerror FODC_IndexError_directory
```

db2cos\_indexerror\_short(.bat) script is executed to collect manual FODC data into one or more files. The files are deposited into the directory FODC\_IndexError\_directory.

To collect data during a performance issue on members 10, 11, 12, 13, and 15:

```
db2fodc -perf -member 10-13,15
```

To collect data during a potential hang situation on the local host:

```
db2fodc -hang
```

To detect a threshold condition and to trigger automatic diagnostic data collection when the threshold condition is exceeded multiple times, create a threshold rule like the following query:

```
db2fodc -memory basic -detect free"<=10" connections">=1000" sleeptime="30" iteration="10" interval="10" triggercount="4" duration="5" -member 3
```

The effect of this threshold rule is as follows: On member 3, perform detection to check if the conditions specified by the threshold rules *free<=10* and *connections>=1000* are met. These threshold rules specify that the size of the free list must be 10 or less and the number of connections must be 1000 or more to be detected. FODC memory collection is triggered on member 3 when the threshold conditions for the trigger count are met. For FODC collection to be triggered, the trigger conditions must exist for 40 seconds (triggercount of 4 x 10 second interval = 40 seconds). Detection will sleep for 30 seconds between each iteration, and the total time that detection is enabled for is 5 hours. When FODC memory collection is triggered, a new directory prefixed with FODC\_Memory\_ is created under the current diagnostic path.

To stop the detection of threshold conditions and to turn off all currently active threshold rules, issue the following command. After you issue the command, it can take up to 60 seconds to stop all detection.

```
db2fodc -detect off
```

### Usage notes

When manually invoking the **db2fodc** command, you can indicate the location of where the FODC package directory is created by specifying the **-fodcpath** parameter option together with the command. For automatic FODC, or if you do not specify the **-fodcpath** parameter with the **db2fodc** command, and you specified a list of partitions or members, the **db2fodc** command uses the **FODCPATH** parameter setting for the **DB2FODC** registry variable of the first partition or member from the list specified. If the value for that **FODCPATH** parameter is not set, **db2fodc** uses the

## db2fodc - DB2 first occurrence data collection

instance level **FODCPATH** setting. If you do not specify the **-fodcpath** parameter and do not specify a list of partitions or members, the **db2fodc** command first tries to use the **FODCPATH** parameter setting for the current partition or member; if not set, the instance level setting is used. By default, if you do not specify any FODC path, first occurrence data capture sends diagnostic information to the current diagnostic directory path (**diagpath** or **alt\_diagpath**).

**db2fodc -hang** and **db2fodc -perf** may be run in a multi-partitioned environment with multiple physical database partitions. In this environment

```
db2fodc -perf <full | basic> -member all other_options
```

or

```
db2fodc -perf <full | basic> -alldbpartitionnums other_options
```

should be used to invoke **db2fodc** during a potential hang or severe performance issue at all DB2 members in a single invocation. Use the suboption **full** or **basic** to set a collection mode which does not require user interaction. Options **-member all** and **-member** work for local member or remote member. If **-member** or **-host** is not specified, by default, only information from the current member is collected.

**db2fodc -indexerror** may be run in a multi-partitioned environment with multiple physical or logical database partitions. In this environment,

```
db2_all "<<+member#< db2fodc -indexerror FODC_IndexError_directory <basic | full>"
```

should be used to invoke **db2fodc** to collect information for an index error at a specific DB2 member. Replace member# with the number of the specific member. This number is the last number in the directory name **<FODC\_IndexError\_timestamp\_PID\_EDUID\_Member#>**. An **FODC\_IndexError\_directory** with an absolute path must be specified and the suboption **full** or **basic** may be used to set a collection mode. An absolute path is only required when using **db2fodc -indexerror** with the **db2\_all** command.

**db2fodc -indexerror** will output the progress of the command and log the messages, if any, in the **db2diag** log file.

**db2fodc -indexerror** requires a **FODC\_IndexError\_directory** which contains the **db2cos\_indexerror\_short(.bat)** script and/or **db2cos\_indexerror\_long(.bat)** script as input.

The **db2cos\_indexerror\_short(.bat)** and **db2cos\_indexerror\_long(.bat)** scripts are located in the **FODC\_IndexError\_directory** created during the automatic index error FODC process. The scripts contain multiple **db2dart** commands. After running the scripts, the generated **db2dart** reports will be found in the same **FODC\_IndexError\_directory** that the scripts are located in. Issuing a manual **db2fodc -indexerror** command will not create a new directory. **db2fodc -indexerror** will generate new **db2dart** reports in the same **FODC\_IndexError\_directory** created by the automatic index error FODC process.

Do not rename or move the **FODC\_IndexError\_directory**. The **db2dart** commands in the scripts need this directory path to correctly generate reports.

If you need to run **db2fodc -indexerror** manually, check the **FODC\_IndexError\_directory** for any existing **db2dart** reports. Reports will have the extension **.rpt** and **.rpthex**. If reports exist, you should rename these reports or move them to a subdirectory under the **FODC\_IndexError\_directory** before running



## **db2fodc - DB2 first occurrence data collection**

**db2fodc -indexerror** manually. This will preserve the existing reports for the **db2support** tool to collect and will allow **db2fodc -indexerror** to create new **db2dart** reports.

### db2fopt - Specify query optimizer parameters

The **db2fopt** command specifies parameters for use by the query optimizer. This command can be used when setting up a test system which has less physical resources than the production system.

For example, if the production system is running with **sortheap**=20000 and the test system can only run with **sortheap**=5000, you can use **db2fopt** on the test system to set the **opt\_sortheap** optimizer parameter to 20000. This will direct the optimizer to use 20000 as the sort heap value when evaluating access plans while the **sortheap** database configuration parameter is set to 5000.

**Scope** This command only affects the database partition on which it is executed.

#### Authorization

To query parameters using the get option: none

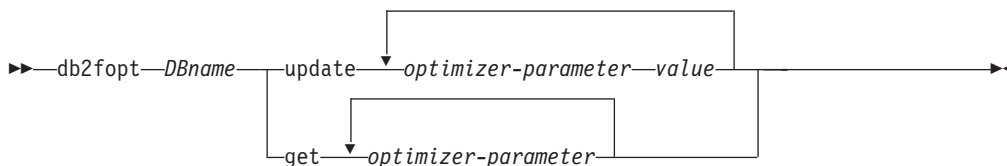
To update parameters, one of the following authorities is required:

- SYSADM
- SYSCTRL
- SYSMANT

#### Required connection

None

#### Command syntax



#### Command parameters

*DBname*

Alias name of the database.

**update** *optimizer-parameter value*

Use this command to update optimizer parameters.

- opt\_buffpage
- opt\_sortheap
- opt\_locklist
- opt\_maxlocks

**get** *optimizer-parameter*

Use this command to query optimizer parameter values.

- opt\_buffpage
- opt\_sortheap
- opt\_locklist
- opt\_maxlocks

## Usage notes

This tool is sometimes used in partitioned database environments that consist of heterogeneous database partition configurations. In this case, statement compilation occurs on a coordinator database partition which can have different database configuration settings from that of the database partitions in the instance on which query processing takes place.

If an optimizer parameter has a value of 0, then no optimizer value has been specified. Statement compilation will use the value from the database configuration.

Updating an optimizer parameter to a value of 0 will reset a previously updated value.

Specifying a non-numeric or a negative value on an update action will set the value to 0.

For an update to take effect, all connections must be terminated on the database partition, and the database partition must be deactivated if previously activated.

The optimizer parameters are only used for statement compilation. In partitioned database environments, they must be set on coordinator database partitions.

To determine the actual values to specify on the test system, you can obtain an explain output from the production system by using the **db2exfmt** tool, and review the *Database context* section.

## Examples

*Example 1:* Query the values of **opt\_sortheap**, **opt\_locklist**, and **opt\_maxlocks**.

```
db2fopt testdb get opt_sortheap opt_locklist opt_maxlocks
```

*Example 2:* Set the **opt\_buffpage** and **opt\_sortheap** parameters.

For example, the following database context section is returned from the production system.

Database Context:

```
-----
Parallelism:          None
CPU Speed:            1.456395e-07
Comm Speed:           0
Buffer Pool size:     89000
Sort Heap size:       10000
Database Heap size:   1200
Lock List size:       8000
Maximum Lock List:    10
Average Applications:  1
Locks Available:      93030
```

On the test system, you can use the **db2fopt** command to set **opt\_buffpage** to 89000, **opt\_sortheap** to 10000, **opt\_locklist** to 8000, and **opt\_maxlocks** to 10.

```
db2fopt testdb update opt_buffpage 89000 opt_sortheap 10000 opt_locklist 8000
                                opt_maxlocks 10
```



## 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*/sql1lib/bin, where *INSTHOME* is the home directory of the instance owner. On Windows operating systems, this command is located in the sql1lib\bin subdirectory.

### Authorization

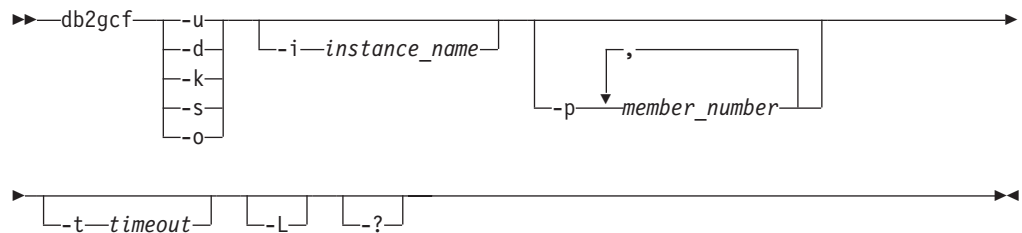
One of the following authorities:

- Instance owner
- Root user authority on Linux and UNIX operating systems or Local Administrator authority on Windows operating systems

### Required connection

None

### Command syntax



### Command parameters

- u Starts specified member for specified instance.
- d Stops specified member for specified instance.
- k Removes all processes associated with the specified instance. On a DB2 Enterprise Server Edition system, this parameter requires that the remote shell utility rsh or ssh is setup on either a single-partition database instance or a multi-partition database instance.
- s Returns status of the specified member and the specified instance. The possible states are:
  - Available: The specified member 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 - Control DB2 instance

db2gcf Error: Neither DB2INSTANCE is set nor instance passed.

### **-p** *member\_number*

In a partitioned database environment, specifies member numbers to perform action against on local node only (remote members are not monitored with this command). Specify the member numbers without any spaces, but separate with commas. 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 file in the instance log directory. Non-instance specific information will be logged to system log files.

### **-?**

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

## Example

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

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

The following output is returned:

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

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

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

The following output is returned:

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

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

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

The following output is returned:

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

## Usage notes

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

## Return codes

The following is a list of all the return codes for this command.

## db2gcf - Control DB2 instance

db2gcf Return Values :

- 0 : db2 service(start,stop,kill) success or db2gcf -s status Available
- 1 : db2 service(start,stop) failed or db2gcf -s status Not Available
- 2 : db2gcf has been called with wrong number of parameters
- 3 : gcfmodule failed to execute the requested service

## 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.

**Important:** With the workload management features introduced in DB2 Version 9.5, the DB2 governor utility was deprecated in Version 9.7 and might be removed in a future release. It is not supported in DB2 pureScale environments. For more information, see “DB2 Governor and Query Patroller have been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0054901.html>.

**Note:** Starting with the DB2 for Linux, UNIX, and Windows Version 10.1 release, the AIX 5.3 operating system is not supported. DB2 for Linux, UNIX, and Windows Version 9.7 is the last release to support the AIX 5.3 operating system. The AIX 6.1 operating system is the minimum supported level.

### Authorization

One of the following authorities:

- 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_a11` 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

```

db2gov
----->
|
| START database [DBPARTITIONNUM db-partition-number] config-file log-file
|----->
|
| STOP database [DBPARTITIONNUM db-partition-number]
|----->

```

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

## Usage notes

In the [action] clause of the governor configuration file, the nice *nnn* parameter can be set to raise or lower the relative priority of agents working for an application. For additional information, see "Governor rule elements" in the guide called *Tuning Database Performance*.

**Note:** On AIX 6.1 or higher, the instance owner must have the `CAP_NUMA_ATTACH` capability to be able to raise the relative priority of agents working for the application. To grant this capability, logon as root and run the following command:

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
```

On Solaris 10 or higher, the instance owner must have the `proc_prioctl` privilege to be able to raise the relative priority of agents working for the application. To grant this privilege, logon as root and run the following command:

```
usermod -K defaultpriv=basic,proc_prioctl db2user
```

In this example, `proc_prioctl` is added to the default privilege set of user `db2user`.

Moreover, when DB2 is running in a non-global zone of Solaris, the `proc_prioctl` privilege must be added to the zone's limit privilege set. To grant this privilege to the zone, logon as root and run the following command:

```
global# zonecfg -z db2zone
zonecfg:db2zone> set limitpriv="default,proc_prioctl"
```

In this example, `proc_prioctl` is added to the limit privilege set of zone `db2zone`.

On Solaris 9, there is no facility for DB2 to raise the relative priority of agents. Upgrade to Solaris 10 or higher to use the `ACTION NICE` clause of DB2 governor.

---

## 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.

**Important:** With the workload management features introduced in DB2 Version 9.5, the DB2 governor utility was deprecated in Version 9.7 and might be removed in a future release. It is not supported in DB2 pureScale environments. For more information, see “DB2 Governor and Query Patroller have been deprecated” at <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0054901.html>.

### 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

## 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

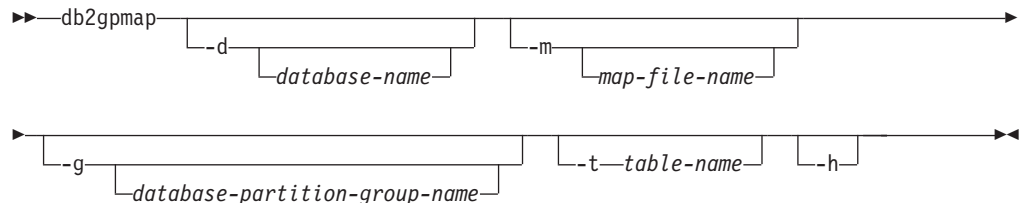
Both of the following authorities:

- 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** *database-name*

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** *map-file-name*

Specifies the fully qualified file name where the distribution map will be saved. The default is db2split.map.

#### **-g** *database-partition-group-name*

Specifies the name of the database partition group for which to generate a distribution map. The default is IBMDEFAULTGROUP.

#### **-t** *table-name*

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
```

## db2iauto - Autostart instance

Enables or disables the autostart of an instance after each system restart. This command is available on Linux and UNIX operating systems only. This command is not supported for DB2 pureScale instances.

### Authorization

One of the following authorities:

- Root user authority
- SYSADM

### Required connection

None

### Command syntax

```
▶▶ db2iauto [-on | -off] instance-name ▶▶
```

### Command parameters

- on** Enables autostart for the specified instance.
- off** Disables autostart for the specified instance.

*instance-name*

The login name of the instance.

## 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 operating systems.

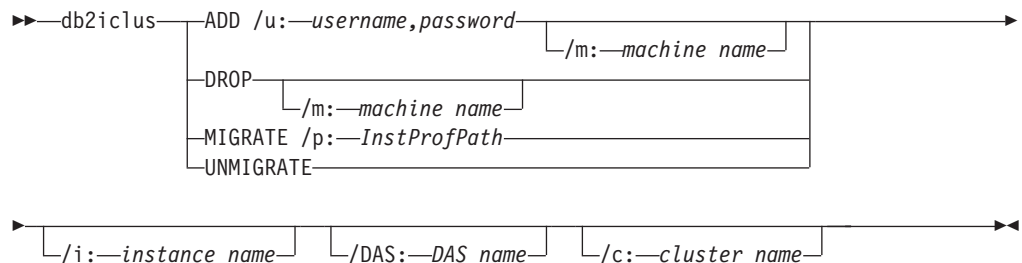
### Authorization

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 or 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:>db2wo1fi i  
ok
```

If the **db2wo1fi** 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:>db2wo1fi u  
ok  
c:>db2wo1fi 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 "mscs5" 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 required, 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.

## db2icrt - Create instance

Create a DB2 instance, including a DB2 pureScale instance. This command can also be used to create an initial DB2 member and cluster caching facility as part of the creation of the DB2 pureScale instance.

On Linux and UNIX operating systems, **db2icrt** is located in *DB2DIR/instance*, where *DB2DIR* represents the installation directory in which the DB2 database system is installed. On Windows operating systems, **db2icrt** is located in *DB2PATH\bin*, where *DB2PATH* is the directory where the DB2 copy is installed.

The **db2icrt** command creates a DB2 instance in the home directory of the instance owner. You can create only one DB2 pureScale instance per DB2 pureScale environment.

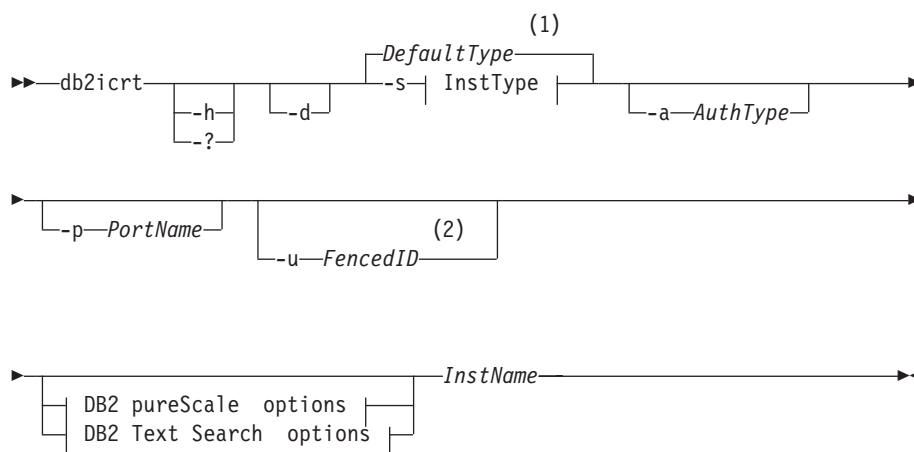
**Note:** This command is not available for a non-root installation of DB2 database products.

### Authorization

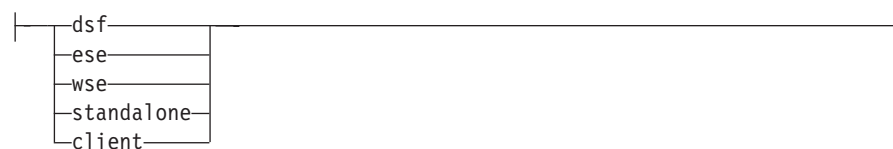
Root user authority is required on Linux and UNIX operating systems or Local Administrator authority on Windows operating systems.

### Command syntax

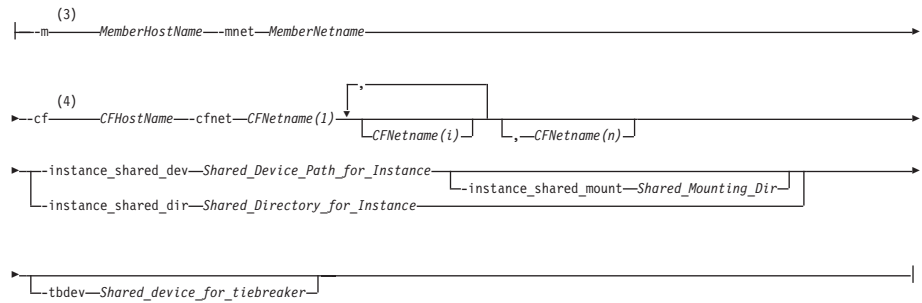
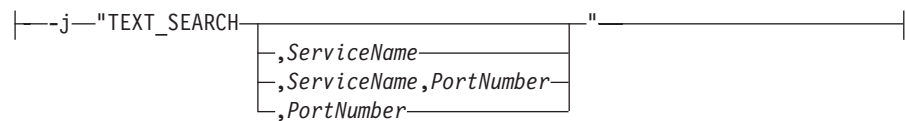
For Linux and UNIX operating systems



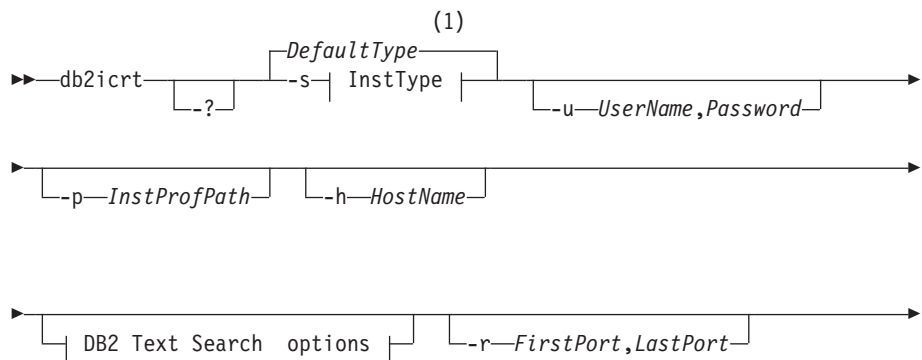
### InstType:





**DB2 pureScale options:****DB2 Text Search options:****Notes:**

- 1 If the instance type is not specified with `-s`, the default instance type created is the highest type listed in *InstType* that DB2 copy is licensed to create.
- 2 When creating client instances, `-u FencedID` is not a valid option.
- 3 The *MemberHostName:MemberNetname* format has been deprecated for the `-m` option, and might be discontinued in the future. The new format, with both `-m` and `-mnet` options, is required for IPv6 support with DB2 pureScale Feature.
- 4 The *CFHostName:CFNetames* format has been deprecated for the `-cf` option, and might be discontinued in the future. The new format, with both `-cf` and `-cfnet` options, is required for IPv6 support with DB2 pureScale Feature.

**For Windows operating systems**

## db2icrt - Create instance

► *InstName* ◄

### InstType:

dsf
ese
wse
standalone
client

### DB2 Text Search options:

--j "TEXT_SEARCH	"
	,ServiceName
	,ServiceName,PortNumber
	,PortNumber

### Notes:

- 1 If the instance type is not specified with *-s*, the default instance type created is the highest type listed in *InstType* that DB2 copy is licensed to create.

## Command parameters

### For Linux and UNIX operating systems

- ? Displays the usage information.
- h Displays the usage information.
- d Turns on debug mode. Saves the trace file with default name in */tmp* as *db2icrt.trc.ProcessID*. Use this option only when instructed by DB2 database support
- a *AuthType*  
Specifies the authentication type (SERVER, CLIENT, or SERVER\_ENCRYPT) for the instance. The default is SERVER.
- j "TEXT\_SEARCH"  
Configures the DB2 Text Search server with generated default values for service name and TCP/IP port number. This parameter cannot be used if the instance type is *client*.
  - j "TEXT\_SEARCH, *servicename*"  
Configures the DB2 Text Search server with the provided service name and an automatically generated port number. If the service name has a port number assigned in the services file, it uses the assigned port number.
  - j "TEXT\_SEARCH, *servicename*, *portnumber*"  
Configures the DB2 Text Search server with the provided service name and port number.
  - j "TEXT\_SEARCH, *portnumber*"

Configures the DB2 Text Search server with a default service name and the provided port number. Valid port numbers must be within the 1024 - 65535 range.

**-p** <TCP/IP PortName>

Specifies the TCP/IP port name or number used by the instance. This option also configures the database manager configuration parameter **SVCENAME** for the DB2 instance.

**-m** *MemberHostName:NetName1*

Specifies the host to set up as a DB2 member during instance creation. This parameter is mandatory in a DB2 pureScale environment. Only one DB2 member can be set up by the **db2icrt** command. Additional DB2 members can be added with the **db2iupdt -add** command. The *NetName1* syntax is deprecated and might be discontinued in a future release. Use the **-mnet** parameter instead.

The *MemberHostName* should be the canonical host name (for example, the output of 'hostname' command run on a local host). The *NetName1* value specified here must belong to the same subnet as specified in the **-cf** parameter.

**-mnet** *MemberNetName*

This parameter replaces the deprecated *:NetName1* syntax of the **-m MemberHostName:NetName1** parameter. Specifies the cluster interconnect netname, which is the hostname of the interconnect used for high speed communication between members and cluster caching facilities (also referred to as CF) in a DB2 pureScale instance.

**-cf** *CFHostName:NetName2*

Specifies the host to set up as a cluster caching facility (also referred to as CF) during instance creation. This parameter is mandatory in a DB2 pureScale environment. Only one CF can be set up by the **db2icrt** command. Additional CFs can be added by using the **db2iupdt -add** command. The *NetName2* syntax is deprecated and might be discontinued in a future release. Use the **-cfnet** parameter instead.

**-cfnet** *CFNetName*

This parameter replaces the deprecated *:NetName2* syntax of the **-cf CFHostName:NetName2** parameter. Specifies the cluster interconnect netname, which is the hostname of the interconnect used for high speed communication between members and CFs in a DB2 pureScale instance.

The *CFNetName* must belong to the same subnet as specified in the **-m** parameter, and must correspond to a cluster interconnect netname (for example, *db2\_<hostname\_ib0>*).

**-instance\_shared\_dev** *Shared\_Device\_Path\_for\_Instance*

Specifies a shared disk device path required to set up a DB2 pureScale instance to hold instance shared files and default database path. For example, */dev/hdisk1*. The shared directory must be accessible on all the hosts for the DB2 pureScale instance. The value of this option cannot have the same value as the **-tbdev** option.

## db2icrt - Create instance

When the **-instance\_shared\_dev** parameter is specified, the DB2 installer creates a DB2 cluster file system.

The **-instance\_shared\_dev** parameter and the **-instance\_shared\_dir** parameter are mutually exclusive.

**-instance\_shared\_mount** *Shared\_Mounting\_Dir*

Specifies the mount point for a new IBM General Parallel File System (GPFS) file system. The specified path must be a new and empty path that is not nested inside an existing GPFS file system.

**-instance\_shared\_dir** *Shared\_Directory\_for\_Instance*

Specifies a directory in a shared file system (GPFS) required to set up a DB2 pureScale instance to hold instance shared files and default database path. For example, /sharedfs. The disk must be accessible on all the hosts for the DB2 pureScale instance. The value of this option cannot have the same value as the **-tbdev** option or the installation path.

When the **-instance\_shared\_dir** parameter is specified, the DB2 installer uses a user-managed file system. The user-managed file system must be available on all hosts, and must be a GPFS file system.

The **-instance\_shared\_dir** parameter and the **-instance\_shared\_dev** parameter are mutually exclusive.

**-tbdev** *Shared\_device\_for\_tiebreaker*

Specifies a shared device path for a device that will act as a tiebreaker in the DB2 pureScale environment to ensure that the integrity of the data is maintained. The value of this option cannot have the same value as either the **-instance\_shared\_dev** option or the **-instance\_shared\_dir** option. This option is required when the DB2 cluster services tiebreaker is created for the first time. The disk device should not have any file system associated with it. This option is invalid if a DB2 cluster services Peer Domain already exists.

**-s** *InstType*

Specifies the type of instance to create. Use the **-s** option only when you are creating an instance other than the default associated with the installed product from which you are running **db2icrt**. Valid values are:

**dsf** Used to create a DB2 pureScale instance for a DB2 database server with local and remote clients. This option is the default instance type for the IBM DB2 pureScale Feature.

**ese** Used to create an instance for a database server with local and remote clients. This option is the default instance type for DB2 Enterprise Server Edition or DB2 Advanced Enterprise Server Edition.

**wse** Used to create an instance for a database server with local and remote clients. This option is the default instance type for DB2 Workgroup Server Edition, DB2 Express® Edition or DB2 Express-C, and DB2 Connect Enterprise Edition.

**standalone**

Used to create an instance for a database server with local clients. It is the default instance type for DB2 Personal Edition.

**client** Used to create an instance for a client. This option is the default instance type for IBM Data Server Client, IBM Data Server Runtime Client, and DB2 Connect Personal Edition.

DB2 database products support their default instance types and the instance types lower than their default ones. For instance, DB2 Enterprise Server Edition supports the instance types of `ese`, `wse`, `standalone`, and `client`.

**-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. The instance name must be the last argument of the **db2icrt** command.

**For Windows operating systems***InstName*

Specifies the name of the instance.

**-s InstType**

Specifies the type of instance to create. Currently, there are four kinds of DB2 instance types. Valid values are:

**client** Used to create an instance for a client. This option is the default instance type for IBM Data Server Client, IBM Data Server Runtime Client, and DB2 Connect Personal Edition.

**standalone**

Used to create an instance for a database server with local clients. It is the default instance type for DB2 Personal Edition.

**ese** Used to create an instance for a database server with local and remote clients with partitioned database environment support. The

`-s ese -u Username, Password`

options have to be used with **db2icrt** to create the ESE instance type and a partitioned database environment instance.

**wse** Used to create an instance for a database server with local and remote clients. This option is the default instance type for DB2 Workgroup Server Edition, DB2 Express Edition or DB2 Express-C, and DB2 Connect Enterprise Edition.

DB2 database products support their default instance types and the instance types lower than their default ones. For instance, DB2 Enterprise Server Edition supports the instance types of `ese`, `wse`, `standalone`, and `client`.

## db2icrt - Create instance

**-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
```

**/j "TEXT\_SEARCH"**

Configures the DB2 Text Search server with generated default values for service name and TCP/IP port number. This parameter cannot be used if the instance type is `client`.

**/j "TEXT\_SEARCH, servicename"**

Configures the DB2 Text Search server with the provided service name and an automatically generated port number. If the service name has a port number assigned in the services file, it uses the assigned port number.

**/j "TEXT\_SEARCH, servicename, portnumber"**

Configures the DB2 Text Search server with the provided service name and port number.

**/j "TEXT\_SEARCH, portnumber"**

Configures the DB2 Text Search server with a default service name and the provided port number. Valid port numbers must be within the 1024 - 65535 range.

**-?** Displays usage information.

### Examples

1. To create a DB2 pureScale instance for the instance owner `db2sdin1` and fenced user `db2sdfel`, run the following command:

```
DB2DIR/instance/db2icrt
-cf host1.domain.com -cfnet host1.domain.com-ib0
-m host2.domain.com -mnet host2.domain.com-ib0
-instance_shared_dev /dev/hdisk1
-tbdev /dev/hdisk2
-u db2sdfel
db2sdin1
```

where `DB2DIR` represents the installation location of your DB2 copy. The DB2 pureScale instance `db2sdin1` will have a CF on `host1`, and a member on `host2`. This command also uses `/dev/hdisk1` to create a shared file system to store instance shared files and sets up `/dev/hdisk2` as the shared device path for the tiebreaker.

- To create a DB2 Enterprise Server Edition instance for the user ID `db2inst1`, run the following command:

```
DB2DIR/instance/db2icrt -s ese -u db2fenc1 db2inst1
```

where *DB2DIR* represents the installation location of your DB2 copy.

- To create a DB2 pureScale instance that uses an existing file system (GPFS) managed by the DB2 product for the instance owner `db2sdin1` and the fenced user `db2sdfe1`, run the following command:

```
DB2DIR/instance/db2icrt
-cf host1.domain.com -cfnet host1.domain.com-ib0
-m host2.domain.com -mnet host2.domain.com-ib0
-tbdev /dev/hdisk2
-u db2sdfe1
db2sdin1
```

where *DB2DIR* represents the installation location of your DB2 copy.

- To create a DB2 pureScale instance with an existing user-managed GPFS file system (`/gpfs_shared_dir`) for the instance owner `db2sdin1` and the fenced user `db2sdfe1`, run the following command:

```
DB2DIR/instance/db2icrt
-cf host1.domain.com -cfnet host1.domain.com-ib0
-m host2.domain.com -mnet host2.domain.com-ib0
-instance_shared_dir /gpfs_shared_dir
-tbdev /dev/hdisk2
-u db2sdfe1
db2sdin1
```

where *DB2DIR* represents the installation location of your DB2 copy.

- 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 instance user must exist on all hosts with the same UID, GID, group name, and home directory path. The same rule applies for the fenced user. After the **db2icrt** command is successfully run, the DB2 installer will set up SSH for the instance user across hosts.
- When using the **db2icrt** command, the name of the instance must match the name of an existing user.
- You can have only one instance per DB2 pureScale environment.
- When creating DB2 instances, consider the following restrictions:
  - If existing IDs are used to create DB2 instances, make sure that the IDs are not locked and do not have passwords expired.
  - Group ID (GID) must be within range 100-65535.
- You can also use the **db2isetup** command to create and update DB2 instances and add multiple hosts with a graphical interface.

## db2icrt - Create instance

- If you are using the **su** command instead of the **login** command to become the root user, you must issue the **su** command with the **-** option to indicate that the process environment is to be set as if you had logged in to the system with the **login** command.
- You must not source the DB2 instance environment for the root user. Running **db2icrt** when you sourced the DB2 instance environment is not supported.
- If you have previously created a DB2 pureScale instance and have dropped it, you cannot re-create it using the **-instance\_shared\_dev** parameter specification since the DB2 cluster file system might already have been created. To specify the previously created shared file system:
  - If the existing GPFS shared file system was created and managed by DB2 pureScale Feature, the **-instance\_shared\_dev** parameter and the **-instance\_shared\_dir** parameter should not be used.
  - If the existing GPFS shared file system was not created and managed by DB2 pureScale Feature, use the **-instance\_shared\_dir** parameter.
- On AIX 6.1 (or higher), when running this command from a shared DB2 copy in a system workload partition (WPAR) global environment, this command must be run as the root user. WPAR is not supported in a DB2 pureScale environment.



## db2idrop - Remove instance

Removes a DB2 instance that was created by **db2icrt**.

You can only drop instances that are listed by the **db2ilist** command for the same DB2 copy where you are issuing the **db2idrop** command from. You can also use the **db2idrop** command to drop a DB2 pureScale instance.

On Linux and UNIX operating 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.

**Note:** A non-root-installed DB2 instance, on Linux and UNIX operating systems, cannot be dropped using this command. The only option is to uninstall the non-root DB2 copy. See the following *Usage notes* section for more details.

### Authorization

Root user authority on Linux and UNIX operating systems or Local Administrator on Windows operating systems.

### Command syntax

For Linux and UNIX operating systems

```

▶▶ db2idrop [-d] [-h] [-?] DB2 pureScale options
▶ Outside Of DB2 pureScale options -InstName

```

#### DB2 pureScale options:

```

[-g]

```

#### Outside Of DB2 pureScale options:

```

[-f]

```

For Windows operating systems

```

▶▶ db2idrop [-f] [-h] -InstName

```

### Command parameters

For Linux and UNIX operating systems

- d Enters debug mode, for use by DB2 database support.
- f This parameter is deprecated.

## db2idrop - Remove instance

Specifies the force applications flag. If this flag is specified all the applications using the instance will be forced to terminate. This parameter is not supported on a DB2 pureScale environment.

**-g** This parameter is required when **db2idrop** is used with a DB2 pureScale instance. Specifies that you want to drop the DB2 pureScale instance on all hosts. This parameter requires all DB2 members and all cluster caching facilities are stopped on all the hosts in the DB2 pureScale instance. This option will be ignored for dropping any other instance type

**-h | -?** Displays the usage information.

*InstName*  
Specifies the name of the instance.

### 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.

### Example

If you created db2inst1 on a Linux or UNIX operating system by issuing the following command:

```
/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

- Before an instance is dropped, ensure that the DB2 database manager has been stopped on all hosts 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. 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.
- If you want to save DB2 Text Search configurations and plan to reuse instance databases, you need to take the extra step of saving the config directory (on UNIX: *instance\_home/sql1lib/db2tss/config* and on Windows: *instance\_profile\_path\instance\_name\db2tss\config*) or config directory contents before issuing the **db2idrop** command. After the new instance is created, the config directory can be restored. However, restoring the config directory is only applicable if the new instance created is of the same release and fix pack level.
- A non-root-installed instance cannot be dropped on Linux and UNIX operating systems. To remove this DB2 instance, the only option available to the user is to uninstall the non-root copy of DB2 by running **db2\_deinstall -a**.
- On Linux and UNIX operating systems, if you are using the **su** command instead of the **login** command to become the root user, you must issue the **su**

command with the `-` option to indicate that the process environment is to be set as if you had logged in to the system using the `login` command.

- On Linux and UNIX operating systems, you must not source the DB2 instance environment for the root user. Running `db2idrop` when you sourced the DB2 instance environment is not supported.
- In a DB2 pureScale environment, the `-g` parameter is mandatory. In this case, the instance is dropped on all hosts. However, the IBM General Parallel File System (GPFS) on the installation-initiating host (IIH) is not deleted, nor is the GPFS file system. You must manually remove the file system and uninstall GPFS.
- On Windows operating systems, if an instance is clustered with Microsoft Cluster Service (MSCS), then you can uncluster that instance by issuing the `db2mcs` or `db2iclus` command before dropping the instance.
- On AIX 6.1 (or higher), when running this command from a shared DB2 copy in a system workload partition (WPAR) global environment, this command must be run as the root user. WPAR is not supported in a DB2 pureScale environment.

### db2IdentifyType1 - Type-1 index identification tool

Identifies and generates into an output file the appropriate **REORG INDEXES ALL** commands with the **ALLOW WRITE ACCESS** and **CONVERT** clauses which you can use to convert type-1 indexes to type-2 indexes for a specified database.

This command only examines user-defined indexes and is intended to be run on databases that have not yet been upgraded to Version 9.7 or later releases.

**Important:** The **db2IdentifyType1** command has been deprecated in Version 10.1 and might be removed in future release. Type-1 indexes were discontinued in Version 9.7. You can still use this command to help you convert type-1 indexes before upgrading databases from Version 9.5 to Version 10.1. For more information, see “db2IdentifyType1 command has been deprecated” in *What's New for DB2 Version 10.1*.

You do not need to run this command if you are upgrading from Version 9.7. Version 9.7 databases do not contain any type-1 indexes. By default, all new indexes created in Version 8 (or later) are type-2 indexes, except when you created an index on a table that already had type-1 indexes or when the **DB2\_INDEX\_TYPE2** registry variable was set to OFF. You might also have type-1 indexes on databases that were created on DB2 Universal Database Version 7 (or earlier) and that have not yet been converted.

To minimize the performance impact on the first access to tables with type-1 indexes, run **db2IdentifyType1**, and convert these indexes before upgrading your database to Version 9.7. If some tables still have type-1 indexes on them during the database upgrade, those indexes are marked invalid. After the upgrade, those indexes are converted to type-2 indexes on first access to the tables, or on database restart if you set the **indexrec** configuration parameter to RESTART. The tables are inaccessible until you have rebuilt the indexes. See the Usage Notes section for other important performance notes and details about the impact of running this command in partitioned database environments.

This command does not perform the index conversion. After you generate the output file, you can review it, edit its content if necessary, and then run the output file by using the following command:

```
db2 -tvf filename
```

Run the **db2IdentifyType1** command from the following directories:

- On Linux and UNIX operating systems: from the DB2DIR/bin directory, where DB2DIR is the location where the DB2 copy is installed
- On Windows operating systems: from the db2\Windows\Utilities directory on the product CD

### Authorization

For databases not upgraded to Version 9.7, one of the following authorization levels is required:

- SYSADM
- DBADM

For databases upgraded to Version 9.7:

- DBADM

## Required connection

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

## Command syntax

```
db2IdentifyType1 -d database-name -o filename [-s schema-name]
[-t table-name] [-h]
```

## Command parameters

### -d database-name

Specifies the name of the database to query.

### -o filename

Specifies the path and the name of a file to which to write the REORG INDEX commands. The db2IdentifyType1.err log file is created in the same path. The db2IdentifyType1.err log file contains troubleshooting information and is created only if the **db2IdentifyType1** command fails. You can qualify the file name, which cannot exceed 246 characters, by using a full or relative path name. If a file with the same name exists, it is overwritten.

### -s schema-name

Specifies the schema (creator user ID) of the tables to query. If you do not specify a schema, tables with all schemas are queried. You cannot specify more than one schema name at a time. You must specify schema name identifiers as they are shown in the system catalog tables, using uppercase. You must enclose delimited schema name identifiers in double quotation marks.

### -t table-name

Specifies the name of a table to query. If you do not specify a table name, all tables are queried. You cannot specify more than one table name at a time. You must specify table name identifiers as they are shown in the system catalog tables, using uppercase. You must enclose delimited table name identifiers in double quotation marks.

**-h** Displays help information. If you specify this option, all other options are ignored.

## Usage notes

The generated output file must be run from a Version 9.5 client since the **CONVERT** clause is discontinued in Version 10.1.

You do not have to use this tool unless there are type-1 indexes in a database or unless you are unsure whether a database contains type-1 indexes. If you know that you have many type-1 indexes, then you might choose to run REORG INDEXES with the CONVERT option against all of the tables in your database, or against specific tables with type-1 indexes, rather than using this command to identify specific tables. REORG INDEXES with the CONVERT option has no effect if the indexes are already type-2.

## **db2IdentifyType1 - Type-1 index identification tool**

For databases with a large number of tables, the **db2IdentifyType1** command can take a long time to complete its processing, you should use the **-s** or **-t** option to scan only a specific subset at a time.

Performance and usage considerations for partitioned database environments: In partitioned database environments, it is possible that type-1 indexes do not exist on all of the database partitions on which a table resides. This could occur if an index was converted to a type-2 index on only a subset of the database partitions. It is also possible that a table resides in a database partition group which does not include every database partition. In Version 9.5 partitioned database environments, the **db2IdentifyType1** command needs to run only on one database partition to detect all type-1 indexes, regardless of whether these indexes exist on all database partitions. For best performance, you should run it on the catalog database partition. The command might take a long time to complete if the database has many partitions because data is gathered from each partition.

## 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 operating 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.

### Usage notes

- On Linux and UNIX operating systems, if you are using the **su** command instead of the **login** command to become the root user, you must issue the **su** command with the **-** option to indicate that the process environment is to be set as if you had logged in to the system using the **login** command.
- On Linux and UNIX operating systems, you must not source the DB2 instance environment for the root user. Running **db2ilist** when you sourced the DB2 instance environment is not supported.
- On AIX 6.1 (or higher), when running this command from a shared DB2 copy in a system workload partition (WPAR) global environment, this command must be run as the root user. WPAR is not supported in a DB2 pureScale environment.

---

### 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 rollforward pending state, or used as a backup image to restore the primary database.

You must issue this command before you can use a split-mirror database.

#### Authorization

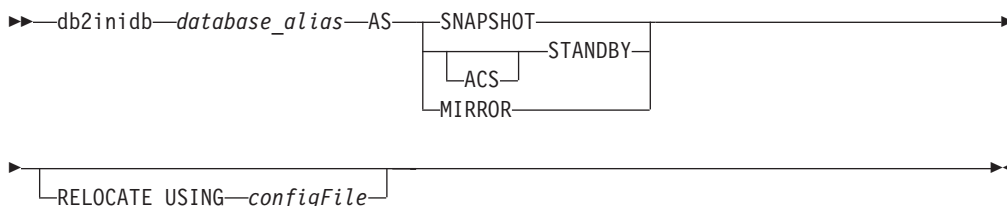
one of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT

#### Required connection

None

#### Command syntax



#### 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 rollforward 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.

##### ACS

Specifies that the **db2inidb** command is to be used against an ACS snapshot copy of the database to perform the **STANDBY** action. This option is required because the **db2inidb** command can only be issued against split mirror database snapshots created with the **SET WRITE SUSPEND | RESUME** command.

Together, the use of the **ACS STANDBY** option initiates the ACS snapshot copy to be placed into a rollforward pending state so that the **DB2 BACKUP** command can be successfully issued against the snapshot image. Without this, any attempt to connect to the snapshot image



results in that copy of the database being placed in the RESTORE\_PENDING state, removing its usefulness as a backup copy for future recovery.

This feature was introduced specifically for interfacing with storage managers such as IBM Tivoli Storage FlashCopy® Manager, for the purpose of producing an offloaded DB2 backup that is based upon an ACS snapshot. Using this option for any other purpose, to mount or modify the contents of an ACS snapshot copy, even including the production of a DB2 backup, can lead to undefined behavior in the future.

**MIRROR** Specifies that the mirrored database is to be a backup image which you can use to restore the primary database.

**RELOCATE USING** *configFile*

Specifies that the database files are to be relocated based on the information listed in the *configFile* before the database is initialized as a snapshot, standby, or mirror. The format of *configFile* is described in “db2relocatedb - Relocate database” on page 1142.

### Usage notes

Do not issue the db2 connect to *database-alias* operation before issuing the db2inidb *database\_alias* 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 partitioned database environments, the **db2inidb** command must be issued 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, if 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.

## db2inidb - Initialize a mirrored database

If a standby database or mirrored database is being initialized, the specified *configFile* file is 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.

When performing an initialization of a split mirror database taken from an HADR primary or standby, use the **STANDBY** parameter if one of the following apply:

- The new database is going to act in an HADR pair and the HADR configuration settings of the new pair are not identical to the settings of the original pair.
- The database is to be initialized as a stand-alone database.

In DB2 pureScale environments, you can issue the **db2inidb** command from any member and have to issue the command only once.

## db2inspf - Format inspect results

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 for a table or a table space, and errors, warnings, and summary can be specified either alone or in any combination thereof.

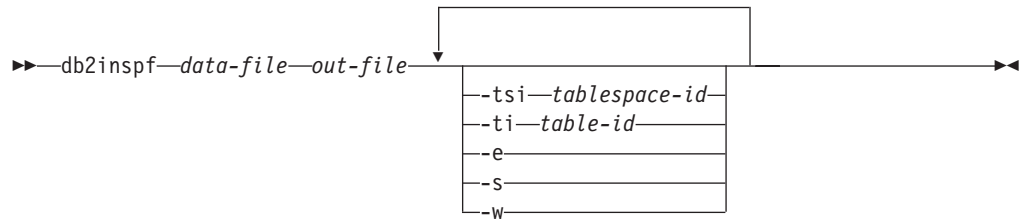
### Authorization

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** *tablespace-id*

Table space ID. Format out only for tables in this table space.

**-ti** *table-id*

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.

### Examples

To format all errors, warnings and summaries from the data file `tbschk.log`, execute the following query:

```
db2inspf tbschk.log tbschk_esw.txt -e -s -w
```



## db2iprune - Reduce installation image size

tmpdir/db2iprune\_username.log. On Windows operating systems, the log file db2iprune.log is written to the destination directory.

**-t** *trace\_file*

(On Linux and UNIX operating systems only.) Turns on the debug mode. The debug information is written to the file name specified.

**-? | -h** Displays the usage information.

### Examples

On Windows operating systems, to prune an IBM Data Server Client image where the input file is located in c:\db2client.prn, the DB2 **setup.exe** file is located in d:\, and you want your pruned IBM Data Server Client image to be copied to the e:\compact\_client directory, you would enter the following command at the command prompt:

```
db2iprune.exe -r c:\db2client.prn -p d:\ -o e:\compact_client
```

On Linux and UNIX operating systems, to prune a IBM Data Server Client image where the input file is located in /tmp/db2client.prn, the DB2 source files are located on /mnt/cdrom, and you want your pruned IBM Data Server Client image to be copied to the /compact\_client directory, you would enter the following command at the command prompt:

```
db2iprune -r /tmp/db2client.prn /mnt/cdrom -o /compact_client
```

### Usage notes

The **db2iprune** command and sample input files are provided on the installation DVD:

#### On Windows operating systems

```
dvd_drive:\db2\Windows\utilities\db2iprune
```

#### On Linux and UNIX operating systems

```
product_image/db2/platform/utilities/db2iprune
```

The **db2iprune** command is not supported in a DB2 pureScale environment.

---

### db2isetup - Start instance creation interface

Starts the DB2 Instance Setup wizard, which is a graphical user interface (GUI) tool for creating, configuring, and extending instances.

#### Authorization

For root installations, root user authority is required on the system where the command is issued. For non-root installations, you must log on with the user ID that owns the non-root installation.

Root user authority is required on the system where the command is issued.

#### Required connection

None

#### Command syntax

```
db2isetup [-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*

Writes the log to the file name specified. For root installations, the path and filename default to `/tmp/db2isetup.log`. For non-root installations, the default log file is `/tmp/db2isetup_<userID>.log`, where *userID* represents the user ID that owns the non-root installation.

**-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. If you plan to use the GUI interface, do not specify this parameter.

**-? | -h**

Output usage information.

#### Usage notes

1. The DB2 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 database components, do system setup tasks such as DAS creation or configuration, and setup, configure and extend instances. The DB2 Instance Setup wizard provides the same functionality as the DB2 Setup wizard except it does not allow the installation of DB2 database components.

## db2isetup - Start instance creation interface

2. The DB2 Instance Setup wizard installs all the required components for the IBM DB2 pureScale Feature on any remote hosts if those components were not installed during the instance creation or extension.
3. The executable file for this command is located in the *DB2DIR/instance* directory. It is available in a typical install, but not in a compact install.
4. **db2isetup** runs on all supported Linux and UNIX operating systems.
5. On AIX 6.1 (or higher), when running this command from a shared DB2 copy in a system workload partition (WPAR) global environment, this command must be run as the root user. .
6. In a DB2 pureScale environment:
  - a. The DB2 pureScale instance must be fully stopped on all hosts before being adding or dropping a member or cluster caching facility (CF.)
  - b. In silent mode, existing hosts of a DB2 pureScale instance can be used in the following cases (these do not apply to the DB2 Setup wizard):
    - To add a second CF to the DB2 pureScale instance, update one of the existing members into both a CF and member host, or add a new host as a CF.
    - To add another member to the DB2 pureScale instance, update the existing CF into both a CF and member host, or add a new host as a member.
    - You cannot use the **db2isetup** command to drop a member or CF.
  - c. There must always be at least one active CF host in the DB2 pureScale instance.

## db2iupdt - Update instances

The **db2iupdt** command can be used to update an instance to a higher level within a release, update an instance other than a DB2 pureScale instance to a DB2 pureScale instance, or scale a DB2 pureScale instance.

Before running the **db2iupdt** command, you must first stop the instance and all processes that are running for the instance. When using this command to update an instance other than a DB2 pureScale instance, add a member or a cluster caching facility, ensure that you have reviewed the prerequisites and the pre-installation checklist so that your instance and hosts are compliant.

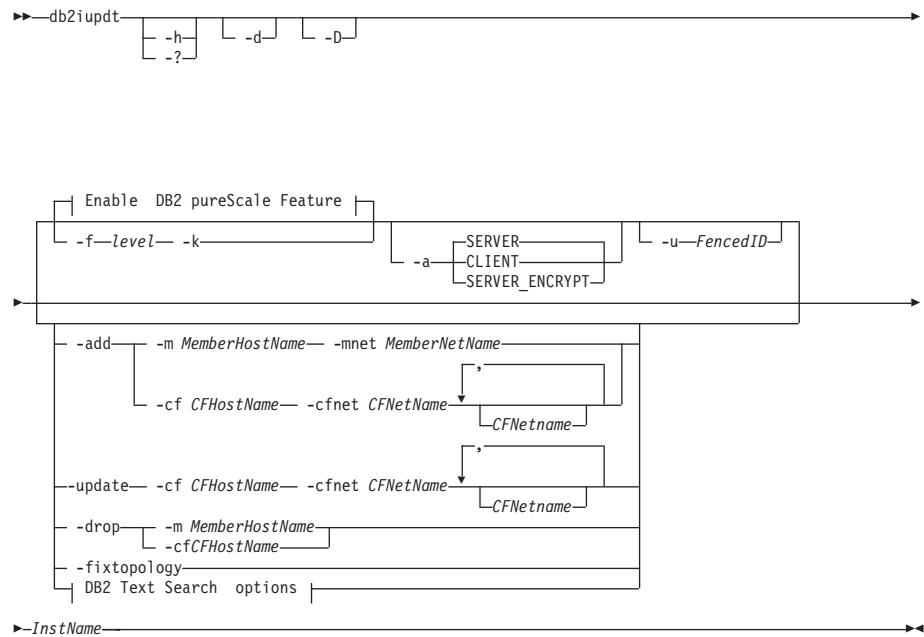
**Note:** In a DB2 pureScale instance, you cannot make changes to the resource model without having a *configurational quorum*, meaning that there is an absolute majority nodes online. In a two-host setup, you cannot use the **db2iupdt** command if one of the hosts is offline.

### Authorization

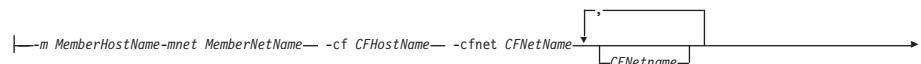
Root user authority on UNIX and Linux operating systems or Local Administrator on Windows operating systems.

### Command syntax

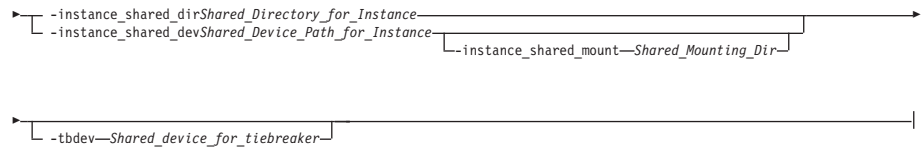
For UNIX and Linux operating systems



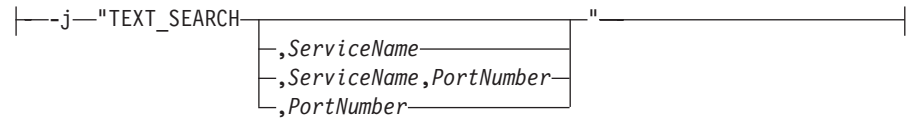
### Enable DB2 pureScale Feature:



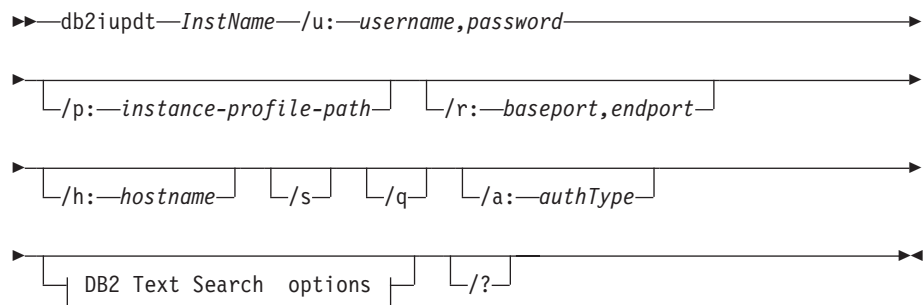




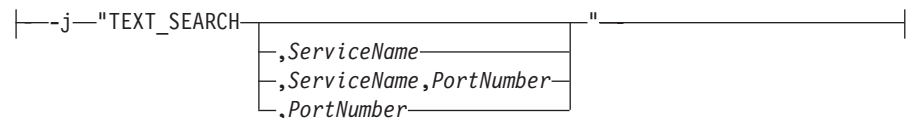
**DB2 Text Search options:**



**For Windows operating systems**



**DB2 Text Search options:**



**Command parameters**

**For UNIX and Linux operating systems**

- h | -? Displays the usage information.
- a *AuthType* Specifies the authentication type (SERVER, SERVER\_ENCRYPT or CLIENT) for the instance. The default is SERVER.
- d Turns on debug mode.
- k Keeps the current instance type during the update.
- D This parameter has been deprecated and might be removed in a future release. This parameter is replaced by the **-f level** parameter. Moves an instance from a higher code level on one path to a lower code level installed on another path.
- f **level** Updates an instance from a higher DB2 version instance type to a lower DB2 version instance type for compatibility.

## db2iupdt - Update instances

- add -m** *MemberHostName:Netname1* | **-cf** *CFHostName:Netname2*  
Specifies the host (member or cluster caching facility) to be added to a DB2 pureScale instance.
- To specify which type of host is to be added, use **-m** for member, or **-cf** for cluster caching facility. You must select one, not both.
- The **db2iupdt -add** command must be run from a host that is already part of the DB2 pureScale instance.
- The **-m** *MemberHostName:NetName1* and **-cf** *CFHostName:NetName2* syntax is deprecated and might be discontinued in a future release. Use the syntax **-add -m** *MemberHostName* **-mnet** *MemberNetName* or **-add -cf** *CFHostName* **-cfnet** *CFNetName* syntax instead.
- add -m** *MemberHostName* **-mnet** *MemberNetName* | **-add -cf** *CFHostName* **-cfnet** *CFNetName*  
Specifies the cluster interconnect netname, which is the hostname of the interconnect used for high speed communication between DB2 members and cluster caching facility. The Netname of each host you want to add has to be on the same subnet as any existing members or CFs in the instance.
- To specify which type of host is to be added, use **-m** *MemberHostName* **-mnet** *MemberNetName* for a member, or **-cf** *CFHostName* **-cfnet** *CFNetName* for a cluster caching facility. You must select one, not both.
- The **db2iupdt -add** command must be run from a host that is already part of the DB2 pureScale instance.
- add** Specifies the hostname and cluster interconnect netname or netnames of the host to be added to the DB2 pureScale Feature instance. The **db2iupdt -add** command must be run from a host that is already part of the DB2 pureScale instance.
- m** *MemberHostName* **-mnet** *MemberNetName*  
The host with hostname *MemberHostName* is added to the DB2 pureScale Feature instance with the cluster interconnect netname *MemberNetName*.
- cf** *CFHostName* **-cfnet** *CFNetName*  
The host with hostname *CFHostName* is added to the DB2 pureScale Feature instance as a cluster caching facility with the cluster interconnect netname *CFNetName*. If *CFHostName* has multiple cluster interconnect network adapter ports, you can supply a comma delimited list for *CFNetName* to separate each cluster interconnect netname.
- drop -m** *MemberHostName* | **-cf** *CFHostName*  
Specifies the host (member or cluster caching facility) to be dropped from a DB2 pureScale instance. To specify which type of host to be dropped, use the **-m** option for a member, or **-cf** option for a cluster caching facility. You must select one, not both. This parameter cannot be used to drop the last member and the last CF from a DB2 pureScale instance. This parameter should not be used with the **-add** parameter.
- instance\_shared\_dev** *Shared\_Device\_Path\_for\_Instance*  
Specifies a shared disk device path required to set up a DB2 pureScale instance to hold instance shared files and default

database path. For example, the device path /dev/hdisk1. The shared directory must be accessible on all the hosts for the DB2 pureScale instance. The value of this parameter cannot have the same value as the **-tbdev** parameter. This parameter and **-instance\_shared\_dir** are mutually exclusive.

This parameter is only required if you are updating an instance other than a DB2 pureScale instance to a DB2 pureScale instance.

**-instance\_shared\_mount** *Shared\_Mounting\_Dir*

Specifies the mount point for a new IBM General Parallel File System ( GPFS) file system. The specified path must be a new and empty path that is not nested inside an existing GPFS file system.

**-instance\_shared\_dir** *Shared\_Directory\_for\_Instance*

Specifies the directory in a shared file system (GPFS) required to set up a DB2 pureScale instance to hold instance shared files and default database path. For example, /sharedfs. The disk must be accessible on all the hosts for the DB2 pureScale instance. The value of this parameter cannot have the same value as the **-tbdev** parameter. This parameter and **-instance\_shared\_dev** are mutually exclusive.

This parameter is only required if you are updating an instance other than a DB2 pureScale instance to a DB2 pureScale instance.

**-tbdev** *Shared\_device\_for\_tiebreaker*

Specifies a shared device path that will act as a tiebreaker in the DB2 pureScale environment to ensure that the integrity of the data is maintained. The value of this parameter cannot have the same value as either the **-instance\_shared\_dev** parameter or the **-instance\_shared\_dir** parameter. This parameter is required when the DB2 cluster services tiebreaker is created for the first time, or if updating an instance other than a DB2 pureScale instance to a DB2 pureScale instance. This parameter is invalid if a DB2 cluster services Peer Domain exists.

**-j "TEXT\_SEARCH"**

Configures the DB2 Text Search server with generated default values for service name and TCP/IP port number. This parameter cannot be used if the instance type is client or dsf.

**-j "TEXT\_SEARCH, servicename"**

Configures the DB2 Text Search server with the provided service name and an automatically generated port number. If the service name has a port number assigned in the services file, it uses the assigned port number.

**-j "TEXT\_SEARCH, servicename, portnumber"**

Configures the DB2 Text Search server with the provided service name and port number.

**-j "TEXT\_SEARCH, portnumber"**

Configures the DB2 Text Search server with a default service name and the provided port number. Valid port numbers must be within the 1024 - 65535 range.

**-u** *Fenced ID*

Specifies the name of the user ID under which fenced user-defined

## db2iupdt - Update instances

functions and fenced stored procedures will run. This parameter is only needed when converting an instance from a client instance to a non-client instance type. To determine the current instance type, refer to the node type parameter in the output from a **GET DBM CFG** command. If an instance is already a non-client instance, or if an instance is a client instance and is staying as a client instance (for example, by using the **-k** parameter), the **-u** parameter is not needed. The **-u** parameter can change the fenced user for an existing instance.

### **-fixtopology**

Used to manually correct a failed add or drop operation. For an add operation, this parameter will roll back any changes to return to the previous topology. For a drop operation, this parameter will complete the drop operation. This parameter cannot be used in combination with any other parameters, except **-d**.

*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.

**/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.

**/j "TEXT\_SEARCH"**

Configures the DB2 Text Search server with generated default values for service name and TCP/IP port number. This parameter cannot be used if the instance type is client.

**/j "TEXT\_SEARCH, servicename"**

Configures the DB2 Text Search server with the provided service name and an automatically generated port number. If the service name has a port number assigned in the services file, it uses the assigned port number.

```
/j "TEXT_SEARCH, servicename, portnumber"
```

Configures the DB2 Text Search server with the provided service name and port number.

```
/j "TEXT_SEARCH, portnumber"
```

Configures the DB2 Text Search server with a default service name and the provided port number. Valid port numbers must be within the 1024 - 65535 range.

```
/?
```

Displays usage information for the **db2iupdt** command.

## Example

### For UNIX and Linux operating systems

A db2inst2 instance is associated to a DB2 copy of DB2 database product installed at *DB2DIR1*. You have another copy of a DB2 database product on the same computer at *DB2DIR2* for the same version of the DB2 database product that is installed on *DB2DIR1*. To update the instance to run from the DB2 copy installed at *DB2DIR1* to the DB2 copy installed at *DB2DIR2*, issue the following command:

```
DB2DIR2/instance/db2iupdt db2inst2
```

If the DB2 copy installed at *DB2DIR2* is at level lower than the DB2 copy installed at *DB2DIR1*, issue:

```
DB2DIR2/instance/db2iupdt -D db2inst2
```

### Update an instance to a higher level within a release

To update a DB2 instance to a higher level or from one DB2 installation path to another, enter a command such as the following:

```
DB2DIR/instance/db2iupdt db2inst1
```

where *DB2DIR* represents the installation location of your DB2 copy. If this command is run from a DB2 pureScale Feature copy, the existing db2inst1 must have an instance type of dsf. If the db2inst1 instance is a DB2 pureScale instance, this example can update it from one level to a different level of DB2 ESE with the DB2 pureScale Feature. This example does not apply to updating an ese type instance to a DB2 pureScale instance. The next example outlines this procedure.

### Update for an instance other than a DB2 pureScale instance to a DB2 pureScale instance

To update an instance to a DB2 pureScale instance:

```
DB2DIR/instance/db2iupdt
-cf host2
-cfnet host2-ib0
-m host1
-mnet host1-ib0
-instance_shared_dev /dev/hdisk1
-tbdev /dev/hdisk2
-u db2fenc1
db2inst1
```

where *DB2DIR* represents the installation location of your DB2 copy.

This command also uses */dev/hdisk1* to create a shared file system to store instance shared files and sets up */dev/hdisk2* as the shared device path that will act as a tiebreaker. The value of the **-tbdev** parameter must be different from the value of the **-instance\_shared\_dev** parameter.

## db2iupdt - Update instances

### Scale a DB2 pureScale instance (by using **db2iupdt -add** or **db2iupdt -drop**)

The following examples apply to a DB2 pureScale environment:

- Update a DB2 pureScale instance to add a member. To add a member called `host1` with a netname of `host1-ib0` to the DB2 pureScale instance `db2sdin1` enter a command such as the following:

```
DB2DIR/instance/db2iupdt -d -add -m host1 -mnet host1-ib0 db2sdin1
```

where *DB2DIR* represents the installation location of your DB2 copy.

- Update a DB2 pureScale instance to add a second cluster caching facility. To add a cluster caching facility called `host2` with a netname of `host2-ib0` to the DB2 pureScale instance `db2sdin1` enter a command such as the following:

```
DB2DIR/instance/db2iupdt -d -add -cf host2 -cfnet host2-ib0 db2sdin1
```

where *DB2DIR* represents the installation location of your DB2 copy.

- Drop a member from a DB2 pureScale instance. To drop a member called `host1` from the DB2 pureScale instance `db2sdin1` enter a command such as the following:

```
DB2DIR/instance/db2iupdt -d -drop -m host1 db2sdin1
```

where *DB2DIR* represents the installation location of your DB2 copy. If `host1` does not have a CF role in the same instance, the command must be run from a host other than `host1`.

## Usage notes

### For all supported operating systems

- If you use the **db2iupdt** command to update a DB2 instance from one DB2 copy to another DB2 copy of the same version of DB2 database product, the DB2 Global Profile Variables defined in an old DB2 copy installation path will not be updated over to the new installation location. The DB2 Instance Profile Variables specific to the instance will be carried over after the instance is updated.

### For UNIX and Linux operating systems

- Only DB2DB2 Enterprise Server Edition can be updated by using the **db2iupdt** command.
- If you change the member topology, for example by adding or dropping a member, you must take an offline backup before you can access the database. If you attempt to access the database before taking an offline backup, the database is placed in a backup pending state.  
You can add multiple members or drop multiple members without having to take a backup after each change. For example, if you add three members, you have to take a backup only after you completed all of the add operations. However, if you add two members and then drop a member, or if you drop two members and then add a member, you must take a backup before you can perform any additional member topology changes.
- The **db2iupdt** command is located in the *DB2DIR*/instance directory, where *DB2DIR* is the location where the current version of the DB2 database product is installed.
- If you want to update a non-root instance, refer to the **db2nrupdt** non-root-installed instance update command. The **db2iupdt** does not support updating of non-root instances.

- If you are using the **su** command instead of the **login** command to become the root user, you must issue the **su** command with the **-** option to indicate that the process environment is to be set as if you had logged in to the system with the **login** command.
- You must not source the DB2 instance environment for the root user. Running **db2iupdt** when you sourced the DB2 instance environment is not supported.
- On AIX 6.1 (or higher), when running this command from a shared DB2 copy in a system workload partition (WPAR) global environment, this command must be run as the root user. WPAR is not supported in a DB2 pureScale environment.
- When **db2iupdt** is run to update an instance to a higher level within a release, routines and libraries are copied from each member to a shared location. If there is a library with the same name but different content on each host, the library content in the shared location is that of the last host that ran the **db2iupdt** command.

### For Windows operating systems

- The **db2iupdt** command is located in the **DB2PATH\bin** directory, where **DB2PATH** is the location where the current version of the DB2 database product is installed.
- The instance is updated to the DB2 copy from which you issue the **db2iupdt** command. However, 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 the instance update. Use the **db2iupgrade** command instead to upgrade to the current release from a previous release.

## db2iupgrade - Upgrade instance

Upgrades an instance to a DB2 copy of the current release from a DB2 copy of a previous release. The DB2 copy from where you are running the **db2iupgrade** command must support instance upgrade from the DB2 copy that you want to upgrade.

On Linux and UNIX operating systems, this command is located in the *DB2DIR/instance* directory, where *DB2DIR* represents the installation location where the new release of the DB2 database system is installed. This command does not support instance upgrade for a non-root installation.

On Windows operating systems, this command is located in the **DB2PATH\bin** directory, where **DB2PATH** is the location where the DB2 copy is installed. 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 the upgrade.

### Authorization

Root user authority on Linux and UNIX operating systems or Local Administrator on Windows operating systems.

### Command syntax

#### For Linux and UNIX operating systems

```

db2iupgrade [-d] [-k]
             [-j "TEXT_SEARCH" [,servicename] [,portnumber]]
             [-a AuthType] [-u FencedID] InstName

```

#### For Windows operating systems

```

db2iupgrade InstName /u: username,password
             [/p: instance-profile-path] [/q] [/a: authType]
             [/j "TEXT_SEARCH" [,servicename] [,portnumber]] [/?]

```

### Command parameters

#### For Linux and UNIX operating systems

- d** Turns debug mode on. Use this option only when instructed by DB2 database support.
- k** Keeps the pre-upgrade instance type if it is supported in the DB2



copy from where you are running the **db2iupgrade** command. If this parameter is not specified, the instance type is upgraded to the default instance type supported.

### **-j "TEXT\_SEARCH"**

Configures the DB2 Text Search server using generated default values for service name and TCP/IP port number. This parameter cannot be used if the instance type is client.

### **-j "TEXT\_SEARCH, servicename"**

Configures the DB2 Text Search server using the provided service name and an automatically generated port number. If the service name has a port number assigned in the services file, it uses the assigned port number.

### **-j "TEXT\_SEARCH, servicename, portnumber"**

Configures the DB2 Text Search server using the provided service name and port number.

### **-j "TEXT\_SEARCH, portnumber"**

Configures the DB2 Text Search server using a default service name and the provided port number. Valid port numbers must be within the 1024 - 65535 range.

### **-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 required when upgrading a DB2 client instance to a DB2 server instance.

### *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 upgrading a partitioned instance.

### **/p:instance-profile-path**

Specifies the new instance profile path for the upgraded instance.

### **/q**

Issues the **db2iupgrade** command in quiet mode.

### **/a:authType**

Specifies the authentication type (SERVER, CLIENT, or SERVER\_ENCRYPT) for the instance.

### **/j "TEXT\_SEARCH"**

Configures the DB2 Text Search server using generated default values for service name and TCP/IP port number. This parameter cannot be used if the instance type is client.

### **/j "TEXT\_SEARCH, servicename"**

## db2iupgrade - Upgrade instance

Configures the DB2 Text Search server using the provided service name and an automatically generated port number. If the service name has a port number assigned in the services file, it uses the assigned port number.

`/j "TEXT_SEARCH, servicename, portnumber"`

Configures the DB2 Text Search server using the provided service name and port number.

`/j "TEXT_SEARCH, portnumber"`

Configures the DB2 Text Search server using a default service name and the provided port number. Valid port numbers must be within the 1024 - 65535 range.

`/?` Displays usage information for the **db2iupgrade** command.

### Usage notes

Only DB2 Enterprise Server Edition instances (instance type *ese*) can be upgraded using the **db2iupgrade** command.

The **db2iupgrade** command calls the **db2ckupgrade** command with the **-not1** parameter, and specifies `update.log` as the log file for **db2ckupgrade**. The default log file created for **db2iupgrade** is `/tmp/db2ckupgrade.log.processID`. Verify that local databases are ready for upgrade before upgrading the instance. The **-not1** parameter disables the check for type-1 indexes. The log file is created in the instance home directory for Linux and UNIX operating systems or in the current directory for Windows operating systems. The instance upgrade will not continue if the **db2ckupgrade** command returns any errors.

For partitioned database environments, run the **db2ckupgrade** command before you issue the **db2iupgrade** command. The **db2ckupgrade** command will check all partitions and will return errors found in any partition. If you do not check whether all database partitions are ready for upgrade, subsequent database upgrades could fail even though the instance upgrade was successful. See **db2ckupgrade** for details.

### For Linux and UNIX operating systems

- If you use the **db2iupgrade** command to upgrade 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 upgraded over to the new installation location. The DB2 Instance Profile Variables specific to the instance to be upgraded will be carried over after the instance is upgraded.
- If you are using the **su** command instead of the **login** command to become the root user, you must issue the **su** command with the **-** option to indicate that the process environment is to be set as if you had logged in to the system using the **login** command.
- You must not source the DB2 instance environment for the root user. Running the **db2iupgrade** command when you sourced the DB2 instance environment is not supported.
- On AIX 6.1 (or higher), when running this command from a shared DB2 copy in a system workload partition (WPAR) global environment, this command must be run as the root user. WPAR is not supported in a DB2 pureScale environment.

## db2jdbcbind - DB2 JDBC package binder

This utility is used to bind or rebind the JDBC packages to a DB2 database.

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

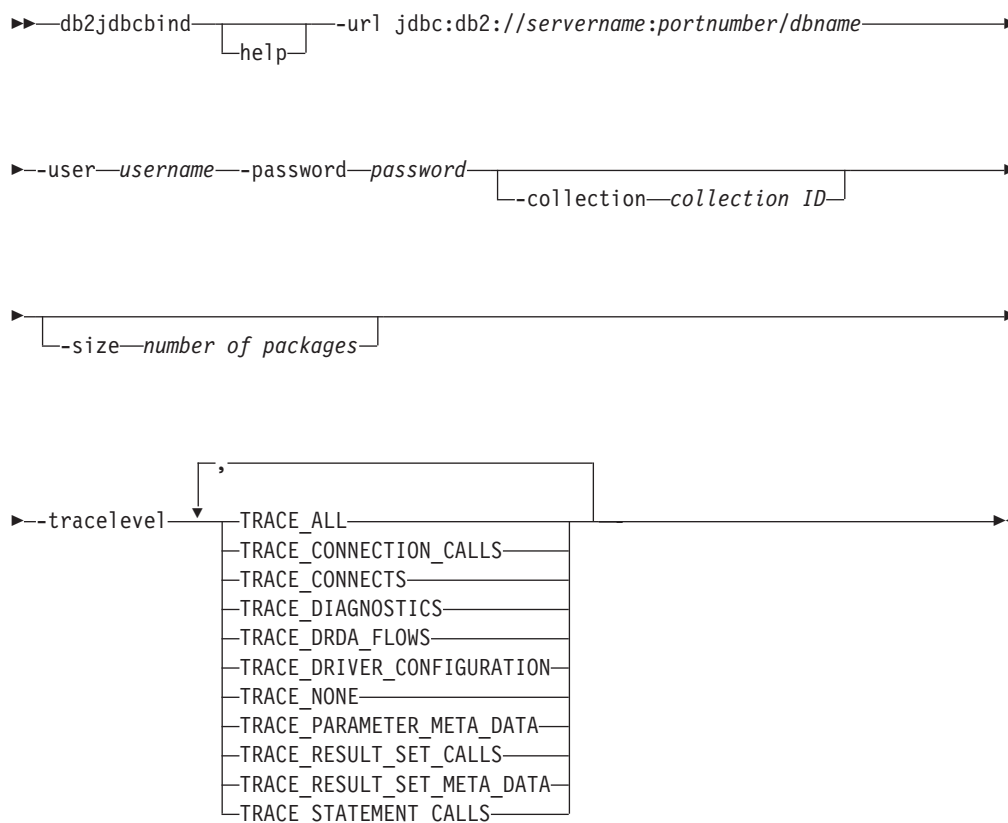
One of the following authorities:

- 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



### Command parameters

`-help` Displays help information, all other options are ignored.

## db2jdbcbind - DB2 JDBC package binder

- url** *jdbc:db2://servername:portnumber/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 userDN -w password
```

### Command parameters

**-u** *userDN*

Specifies the LDAP user's Distinguished Name to be used when accessing the LDAP directory. As shown in the following example, the Distinguished name has several parts: the user ID, such as *jdoe*, 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=jdoe,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.

### 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—▶

#### Command parameters

None

#### 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".
```

#### Usage notes

The information output by the command includes Release, Level, and various informational tokens.

## db2licm - License management tool

Performs basic license functions such as adding, removing, listing, or modifying licenses and policies installed on the local system.

**Note:** Under the processor Value Unit (PVU) licensing structure, each processor core will be assigned a specific number of Value Units. You must acquire the total number of processor Value Units for each processor core on which the software programs are deployed. IBM continues to define a processor to be each processor core on a chip. For example, a dual-core chip contains two processor cores.

Each software program has a unique price per Value Unit. To determine the total cost of deploying an individual software program, you multiply the program price per Value Unit by the total number of processor Value Units required.

### Authorization

On Windows operating systems:

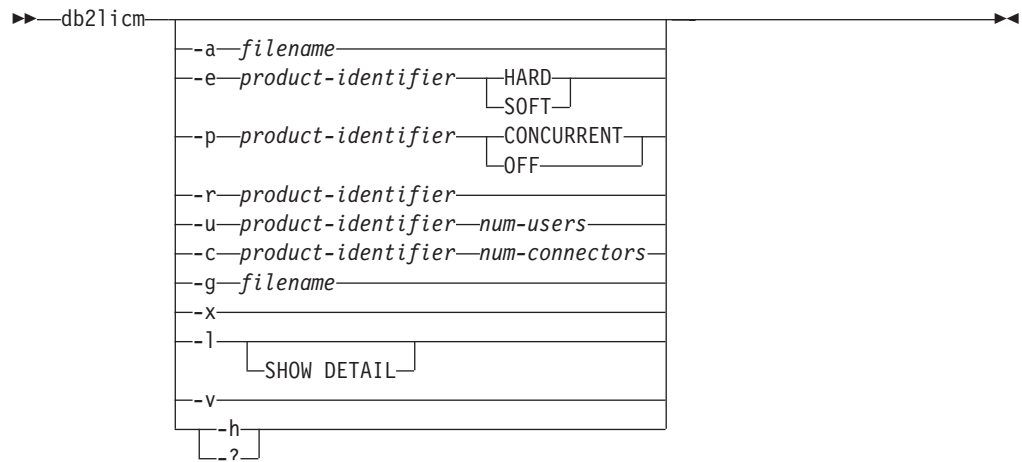
- You must belong to the local Administrators or Power Users group to use the **-a**, **-r**, or **-x** command parameters.
- SYSADM authority is required to use the **-c**, **-e**, **-p**, **-r**, or **-u** command parameters.

On UNIX and Linux operating systems, no authority is required.

### 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.

## db2licm - License management tool

- 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.
- e *product-identifier*  
Updates the enforcement policy on the system. Valid values are:
  - HARD** Specifies that unlicensed requests will not be allowed.
  - SOFT** Specifies that unlicensed requests will be logged but not restricted.
- g *filename*  
Generates compliance report. Specify file name where output is to be stored.
- h | -?  
Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.
- l  
Lists all the products with available license information, including the product identifier.
  - SHOW DETAIL**  
Specify to view detailed information about licensed features (if any).
- p *product-identifier*  
Updates the license policy type to use on the system.
  - CONCURRENT**  
Specify for concurrent user policy.
  - OFF** Specify to turn off all policies.
- r *product-identifier*  
Removes the license for a product. To get the product identifier 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.
- v  
Displays version information.
- x  
Resets license compliance information for the purposes of license compliance report.

### Examples

```
db2licm -a db2ese_u.lic
db2licm -p db2consv concurrent
db2licm -r db2ese
db2licm -u db2wse 10
db2licm -e db2ese SOFT
```

Output example listing all the products with available license information, including the product identifier:

```
C:\Program Files\IBM\SQLLIB\BIN>db2licm -l
Product name:          "DB2 Enterprise Server Edition"
License type:          "Trial"
Expiry date:           "08/31/2009"
Product identifier:    "db2ese"
Version information:   "10.1"
```



## db2licm - License management tool

**Note:** In Linux the `db2licm -l` command writes the following informational message to `/var/log/messages`:

```
kernel: Program db2licm tried to access /dev/mem between 1f0000000f0000->10100000000.
```

### 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 complete 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



### db2look - DB2 statistics and DDL extraction tool

Extracts the Data Definition Language (DDL) statements that are required to reproduce the database objects of a production database on a test database. The **db2look** command generates the DDL statements by object type. Note that this command ignores all objects under SYSTOOLS schema except user-defined functions and stored procedures.

It is often advantageous to have a test system that contains a subset of the data of a production system, but access plans selected for such a test system are not necessarily the same as those that would be selected for the production system. However, using the **db2look** tool, you can create a test system with access plans that are similar to those that would be used on the production system. You can use this tool to generate the UPDATE statements that are required to replicate the catalog statistics on the objects in a production database on a test database. You can also use this tool to generate **UPDATE DATABASE CONFIGURATION**, **UPDATE DATABASE MANAGER CONFIGURATION**, and **db2set** commands so that the values of query optimizer-related configuration parameters and registry variables on a test database match those of a production database.

You should check the DDL statements that are generated by the **db2look** command because they might not 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 DATABASE** command.

#### Authorization

SELECT privilege on the system catalog tables.

In some cases, such as generating table space container DDL, you will require one of the following authorities:

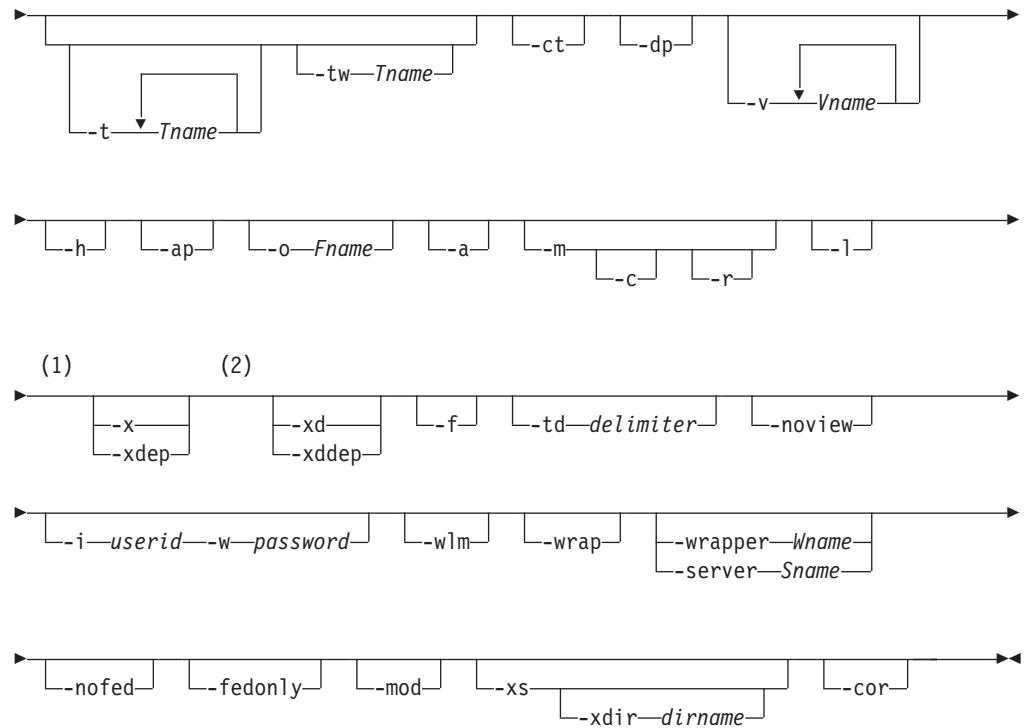
- SYSADM
- SYSCtrl
- SYSMAINT
- SYSMON
- DBADM
- EXECUTE privilege on the ADMIN\_GET\_STORAGE\_PATHS table function

#### Required connection

None

#### Command syntax

```
▶▶ db2look --d DBname [-e] [-u creator] [-z schema]
```



**Notes:**

- 1 You cannot specify both the **-x** parameter and **-xdep** parameter
- 2 You cannot specify both the **-xd** parameter and **-xddep** parameter

**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 database. If the *DBname* is a DB2 for z/OS database, the **db2look** command generates the following statements for OS/390 and z/OS objects:

- DDL statements for tables, indexes, views, and user-defined distinct types
- UPDATE statistics statements for tables, columns, column distributions, and indexes

These DDL and UPDATE statistics statements are applicable to a DB2 Database for Linux, UNIX, and Windows database and not to a DB2 for z/OS database. These statements are useful if you want to extract OS/390 and z/OS objects and re-create them in a DB2 Database for Linux, UNIX, and Windows database.

**-e** Extracts DDL statements for the following database objects:

- Aliases
- Audit policies
- Check constraints
- Function mappings
- Function templates

## db2look - DB2 statistics and DDL extraction tool

- Global variables
- Indexes (including partitioned indexes on partitioned tables)
- Index specifications
- Materialized query tables (MQTs)
- Nicknames
- Primary key, referential integrity, and check constraints
- Referential integrity constraints
- Roles
- Schemas
- Security labels
- Security label components
- Security policies
- Sequences
- Servers
- Stored procedures
- Tables

**Note:** Values from column STATISTICS\_PROFILE in the SYSIBM.SYSTABLES catalog table are not included.

- Triggers
- Trusted contexts
- Type mappings
- User mappings
- User-defined distinct types
- User-defined functions
- User-defined methods
- User-defined structured types
- User-defined transforms
- Views
- Wrappers

If you use DDL statements that are generated by the **db2look** command to re-create a user-defined function, the source code that the function references (the EXTERNAL NAME clause, for example) must be available for the function to be usable.

### **-u** *creator*

Generates DDL statements for objects that were created with the specified creator ID. Limits output to objects that were created with the specified creator ID. The output does not include any inoperative objects. To display inoperative objects, use the **-a** parameter. If you specify the **-a** parameter, the **-u** parameter is ignored.

### **-z** *schema*

Generates DDL statements for objects that have the specified schema name. Limits output to objects that have the specified schema name. The output does not include any inoperative objects. To display inoperative objects, use the **-a** parameter. If you do not specify the **-z** parameter, objects with all schema names are extracted. If you specify the **-a** parameter, the **-z** parameter is ignored. This parameter is also ignored for federated DDL statements.

**-t** *Tname1 Tname2 ... TnameN*

Generates DDL statements for the specified tables and their dependent objects. Limits output to the tables that are specified in the table list and generates the DDL statements for all dependent objects of all user specified tables. The maximum number of tables is 30. Specify the list as follows:

- Separate table names by a blank space.
- Enclose case-sensitive names and double-byte character set (DBCS) names with the backslash (\) and double quotation marks (" ") (for example, \`MyTable` \").
- Enclose multiword table names with the backslash and two sets of double quotation marks (for example, \`"My Table"`) to prevent the pairing from being evaluated word-by-word by the command line processor (CLP). If you use only one set of double quotation marks (for example, `My Table`), all words are converted into uppercase, and the **db2look** command looks for an uppercase table name (for example, `MY TABLE`).

If you specify the **-t** parameter with the **-l** parameter, partitioned tables are supported.

You can use two-part table names of the format *schema.table* to fully qualify a table name without using the **-z schema** parameter. Use a two-part table name when the table has dependent objects that are in a different schema than that of the table and you require DDL statements to be generated for the dependent objects. If you use the **-z schema** parameter to specify the schema, the parameter excludes dependent objects that do not have the same parent schema, thereby preventing the generation of DDL statements for the dependent objects.

**-tw** *Tname*

Generates DDL statements for tables with names that match the pattern that you specify with *Tname* and generates the DDL statements for all dependent objects of those tables. *Tname* must 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. When **-tw** is specified, the **-t** option is ignored.

You can use two-part table names of the format *schema.table* to fully qualify a table name without using the **-z schema** parameter. Use a two-part table name when the table has dependent objects that are in a different schema than that of the table and you require DDL statements to be generated for the dependent objects. If you use the **-z schema** parameter to specify the schema, the parameter excludes dependent objects that do not have the same parent schema, thereby preventing the generation of DDL statements for the dependent objects.

- ct** Generates DDL statements by object creation time. The object DDL statements might not be displayed in correct dependency order. If you specify the **-ct** parameter, the **db2look** command supports only the following additional parameters: **-e**, **-a**, **-u**, **-z**, **-t**, **-tw**, **-v**, **-l**, **-noview**, and **-wlm**. If you use the **-ct** parameter with the **-z** and **-t** parameters, the **db2look** command generates the required UPDATE statements to replicate the statistics on tables, statistical views, columns, and indexes.
- dp** Generates a DROP statement before a CREATE statement. The DROP statement might not work if there is an object that depends on the dropped object. For example, you cannot drop a schema if there is a table that depends on the schema, and you cannot drop a user-defined type or

## db2look - DB2 statistics and DDL extraction tool

user-defined function if there is a type, function, trigger, or table that depends on that user-defined type or user-defined function. For typed tables, the DROP TABLE HIERARCHY statement is generated for the root table only. A DROP statement is not generated for indexes, primary and foreign keys, and constraints because they are always dropped when the table is dropped. You cannot drop a table that has the RESTRICT ON DROP attribute.

**-v** *Vname1 Vname2 ... VnameN*

Generates DDL statements for the specified views, but not for their dependent objects. The maximum number of views is 30. The rules governing case-sensitive, DBCS, and multiword table names also apply to view names. If you specify the **-t** parameter, the **-v** parameter is ignored.

You can use a two-part view name of the format *schema.view* to fully qualify a view.

**-h** Display help information. If you specify this parameter, all other parameters are ignored.

**-ap** Generates the AUDIT USING statements that are required to associate audit policies with other database objects.

**-o** *Fname*

Writes the output to the *Fname* file. If you do not specify an extension, the .sql extension is used. If you do not specify this parameter, output is written to standard output.

**-a** Generates DDL statements for objects that were created by any user, including inoperative objects. For example, if you specify this parameter with the **-e** parameter, DDL statements are extracted for all objects in the database. If you specify this parameter with the **-m** parameter, UPDATE statistics statements are extracted for all user-created tables and indexes in the database.

If you do not specify either the **-u** or the **-a** parameter, the USER environment variable is used. On UNIX operating systems, you do not have to explicitly set this variable. On Windows operating systems, however, there is no default value for the USER environment variable. Therefore, you must set a user variable in the SYSTEM variables or issue the **set USER=username** command for the session.

**-m** Generates the UPDATE statements that are required to replicate the statistics on tables, statistical views, columns, and indexes. Using the **-m** parameter is referred to as running in mimic mode.

**-c** If you specify this option, the **db2look** command does not generate COMMIT, CONNECT, and CONNECT RESET statements. The default action is to generate these statements. This option is ignored unless you also specify the **-m** or **-e** parameter.

**-r** If you specify this option with the **-m** parameter, the **db2look** command does not generate the **RUNSTATS** command. The default action is to generate the **RUNSTATS** command.

**Important:** If you intend to run the command processor script that is created using the **db2look** command with the **-m** parameter against another database (for example, to make the catalog statistics of the test database match those in production), both databases must use the same code set and territory.



- l Generates DDL statements for the following database objects:
  - User-defined table spaces
  - User-defined storage groups
  - User-defined database partition groups
  - User-defined buffer pools
- x Generates authorization DDL statements such as GRANT statements.

The supported authorizations include the following ones:

- Columns: UPDATE, REFERENCES
- Databases: ACCESSCTRL, BINDADD, CONNECT, CREATETAB, CREATE\_EXTERNAL\_ROUTINE, CREATE\_NOT\_FENCED\_ROUTINE, DATAACCESS, DBADM, EXPLAIN, IMPLICIT\_SCHEMA, LOAD, QUIESCE\_CONNECT, SECADM, SQLADM, WLMADM
- Exemptions
- Global variables
- Indexes: CONTROL
- Packages: CONTROL, BIND, EXECUTE
- Roles
- Schemas: CREATEIN, DROPIN, ALTERIN
- Security labels
- Sequences: USAGE, ALTER
- Stored procedures: EXECUTE
- Tables: ALTER, SELECT, INSERT, DELETE, UPDATE, INDEX, REFERENCE, CONTROL
- Views: SELECT, INSERT, DELETE, UPDATE, CONTROL
- User-defined functions (UDFs): EXECUTE
- User-defined methods: EXECUTE
- Table spaces: USE
- Workloads: USAGE

**Note:** This parameter does not generate authorization DDL statements for dependent objects when used with either the **-t** or **-tw** parameter. Use the **-xdep** parameter to generate authorization DDL statements for parent and dependent objects.

- xdep Generates authorization DDL statements, for example, GRANT statements, for parent and dependent objects. This parameter is ignored if either the **-t** or **-tw** parameter is not specified. The supported authorizations include the following ones:
  - Columns: UPDATE, REFERENCES
  - Indexes: CONTROL
  - Stored procedures: EXECUTE
  - Tables: ALTER, SELECT, INSERT, DELETE, UPDATE, INDEX, REFERENCE, CONTROL
  - Table spaces: USE
  - User-defined functions (UDFs): EXECUTE
  - User-defined methods: EXECUTE
  - Views: SELECT, INSERT, DELETE, UPDATE, CONTROL
- xd Generates authorization DDL statements, including authorization DDL

## db2look - DB2 statistics and DDL extraction tool

statements for objects whose authorizations were granted by SYSIBM at object creation time. It does not generate the authorization DDLs for system catalog tables and catalog views.

**Note:** This parameter does not generate authorization DDL statements for dependent objects when used with either the **-t** or **-tw** parameter. Use the **-xddep** parameter to generate authorization DDL statements for parent and dependent objects.

### **-xddep**

Generates all authorization DDL statements for parent and dependent objects, including authorization DDL statements for objects whose authorizations were granted by SYSIBM at object creation time. This parameter is ignored if either the **-t** or **-tw** parameter is not specified.

### **-f**

Extracts the configuration parameters and registry variables that affect the query optimizer.

### **-td** *delimiter*

Specifies the statement delimiter for SQL statements that are generated by the **db2look** command. The default delimiter is the semicolon (;). Use this parameter if you specify the **-e** parameter because the extracted objects might contain triggers or SQL routines.

### **-noview**

Specifies that CREATE VIEW DDL statements will not be extracted.

### **-i** *userid*

Specifies the user ID that the **db2look** command uses to log on to a remote system. When you specify this parameter and the **-w** parameter, the **db2look** command can run against a database on a remote system. The local and remote database must use the same DB2 version.

### **-w** *password*

Specifies the password that the **db2look** command uses to log on to a remote system. When you specify this parameter and the **-i** parameter, the **db2look** command can run against a database on a remote system. The local and remote database must use the same DB2 version.

### **-wlm**

Generates WLM-specific DDL output, which can serve to generate CREATE and ALTER statements for the following items:

- Histograms
- Service classes
- Thresholds
- WLM event monitors
- Workloads
- Work action sets
- Work class sets

### **-wrap**

Generates obfuscated versions of DDL statements for routines, triggers, views, and PL/SQL packages.

### **-wrapper** *Wname*

Generates DDL statements for federated objects that apply to the specified wrapper. The federated DDL statements that might be generated include the following ones:

- CREATE FUNCTION ... AS TEMPLATE
- CREATE FUNCTION MAPPING

- CREATE INDEX SPECIFICATION
- CREATE NICKNAME
- CREATE SERVER
- CREATE TYPE MAPPING
- CREATE USER MAPPING
- CREATE WRAPPER
- GRANT (privileges to nicknames, servers, indexes)

An error is returned if you do not specify a wrapper name or specify more than one.

**-server** *Sname*

Generates DDL statements for federated objects that apply to the specified server. The federated DDL statements that might be generated include the following ones:

- CREATE FUNCTION ... AS TEMPLATE
- CREATE FUNCTION MAPPING
- CREATE INDEX SPECIFICATION
- CREATE NICKNAME
- CREATE SERVER
- CREATE TYPE MAPPING
- CREATE USER MAPPING
- CREATE WRAPPER
- GRANT (privileges to nicknames, servers, indexes)

An error is returned if you do not specify a server name or specify more than one.

**-nofed** Specifies that no federated DDL statements will be generated. If you specify this parameter, the **-wrapper** and **-server** parameters are ignored.

**-fedonly**

Specifies that only federated DDL statements will be generated.

**-mod** Generates DDL statements for each module, and for all of the objects that are defined in each module.

**-xs** Exports all files that are necessary to register XML schemas and DTDs at the target database and generates appropriate commands for registering them. The set of XSR objects that is exported is controlled by the **-u**, **-z**, and **-a** parameters.

**-xdir** *dirname*

Exports XML-related files into the specified path. If you do not specify this parameter, all XML-related files are exported into the current directory.

**-cor** Generates DDL statements with the CREATE OR REPLACE clause, regardless of whether or not the statements originally contained that clause.

## Examples

The following examples show how to use the **db2look** command:

- Generate the DDL statements for objects created by user `walid` in database `DEPARTMENT`. The output is sent to the `db2look.sql` file.

```
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 output is sent to the `db2look.sql` file.

```
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 the `db2look.sql` file.

```
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 output is sent to the `db2look.sql` file.

```
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 output is sent to the `db2look.sql` file.

```
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 output is sent to the `db2look.sql` file.

```
db2look -d department -l -o db2look.sql
```

- Generate the UPDATE statements for optimizer-related database and database manager configuration parameters and the **db2set** commands for optimizer-related registry variables in database `DEPARTMENT`. The output is sent to the `db2look.sql` file.

```
db2look -d department -f -o db2look.sql
```

- Generate the **db2set** commands for optimizer-related registry variables and the following statements for database `DEPARTMENT`:

- The DDL statements for all database objects
- The UPDATE statements to replicate the statistics on all tables and indexes
- The GRANT authorization statements
- The UPDATE statements for optimizer-related database and database manager configuration parameters
- The **db2set** commands for optimizer-related registry variables
- The DDL statements for all user-defined database partition groups, buffer pools, and table spaces

The output is sent to the `db2look.sql` file.

```
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 that were created by the original creator. (In this case, the authorizations were granted by `SYSIBM` at object creation time.) The output is sent to the `db2look.sql` file.

```
db2look -d department -xd -o db2look.sql
```

- Generate the DDL statements for objects created by all users in the database `DEPARTMENT`. The output is sent to the `db2look.sql` file.

```
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 output is sent to the `db2look.sql` file.

```
db2look -d department -e -noview -o db2look.sql
```

## db2look - DB2 statistics and DDL extraction tool

- Generate the DDL statements for objects in database DEPARTMENT related to specified tables. The output is sent to the db2look.sql file.  

```
db2look -d department -e -t tab1 "\"My Tab1E2\" \" -o db2look.sql
```
- 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 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 output is sent to the out.sql file.  

```
db2look -d feddepart -e -nofed -o out
```
- Generate the DDL statements for objects that have schema name valid in the database DEPARTMENT. The files required to register any included XML schemas and DTDs are exported to the current directory. The output is sent to the db2look.sql file.  

```
db2look -d department -z valid -e -xs -o db2look.sql
```
- Generate the DDL statements for objects that were created by all users in the database DEPARTMENT. The files that are required to register any included XML schemas and DTDs are exported to the /home/ofer/ofer/ directory. The output is sent to standard output.  

```
db2look -d department -a -e -xs -xdir /home/ofer/ofer/
```
- Generate only WLM-specific DDL statements for database DEPARTMENT:  

```
db2look -d department -wlm
```
- Generate the DDL statements for all objects in database DEPARTMENT:  

```
db2look -d department -wlm -e -l
```
- Generate the DDL statements for both the parent table TAB1 in schema TABLES and the dependent view of TAB1 that is called VIEW1 in the VIEWS schema in database DB1. The output is sent to the db2look.sql file.  

```
db2look -d DB1 -t TABLES.TAB1 -e -o db2look.sql
```
- Generate the DDL statements and authorization DDL statements for the parent table TAB1 in schema TABLES and the dependent view of TAB1 that is called VIEW1 in the VIEWS schema in database DB1. The output is sent to the db2look.sql file.  

```
db2look -d DB1 -t TABLES.TAB1 -e -xdep -o db2look.sql
```
- Generate the RUNSTATS DDL statements on the TABLE1 table in mimic mode. The output is sent to the db2look.sql file. Not all available RUNSTATS clauses of the command are supported.  

```
db2look -d DB1 -t TABLE1 -m -e -o db2look.sql
```

### Usage notes

On Windows operating systems, you must issue the **db2look** command from a DB2 command window.

By default, the instance owner has the EXECUTE privilege on **db2look** packages. For other users to run the **db2look** command, the instance owner has to grant the EXECUTE privilege on **db2look** packages. To determine the **db2look** package names, the **db2bfd** command can be used as follows:

## db2look - DB2 statistics and DDL extraction tool

```
cd ../sql1lib/bnd
db2bfd -b db2look.bnd
db2bfd -b db21kfun.bnd
db2bfd -b db21ksp.bnd
```

To create DDL statements for federated objects, you must enable the use of federated systems in the database manager configuration. After the **db2look** command generates the script file, you must set the **federated** configuration parameter to YES before running the script. The following **db2look** command parameters are supported in a federated environment:

- ap**  
Generates AUDIT USING statements.
- e** Generates DDL statements for federated objects.
- f** Extracts federated-related information from the database manager configuration.
- m** Extracts statistics for nicknames.
- x** Generates GRANT statements to grant privileges on federated objects.
- xd**  
Generates DDL statements to add system-granted privileges to federated objects.
- wlm**  
Generates WLM-specific DDL statements.

If the nickname column and the remote table column are of different data types, then the **db2look** command will generate an ALTER COLUMN statement for the nickname column.

You must modify the output script to add the remote passwords for the CREATE USER MAPPING statements.

You must 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, DBCS, and multi-word table and view names must be enclosed by both a backslash and double quotation marks. For example:

```
\ "My Table"
```

If a multibyte character set (MBCS) or double-byte character set (DBCS) name is not enclosed by the backward slash and double quotation delimiter and if it contains the same byte as the lowercase character, it will be converted into uppercase and **db2look** will look for a database object with the converted name. As a result, the DDL statement will not be extracted.

- 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 for i , or DB2 Server for VSE & VM.
- The **-tw** option is only supported via the CLP.

If you try to generate DDL statements on systems with a partitioned database environment, a warning message is displayed in place of the DDL statements for table spaces that are on inactive database partitions. To ensure that correct DDL statements are produced for all table spaces, you must activate all database partitions.

The **db2look** command is supported only on DB2 Version 9.5 or later databases, but you can issue the command from earlier DB2 clients. For example, issuing this command from a Version 9.1 client for a Version 9.5 database is supported, but issuing the command from a Version 9.5 client for a Version 9.1 database is not supported.

When you invoke the db2look utility, the **db2look** command generates the DDL statements for any object created using an uncommitted transaction.

When you extract a DDL statement for a security label component of type array, the extracted statement might not generate a component whose internal representation (encoding of elements in that array) matches that of the component in the database for which you issued the **db2look** command. This mismatch can happen if you altered the security label component by adding one or more elements to it. In such cases, data that you extract from one table and moved to another table that you created from **db2look** output will not have corresponding security label values, and the protection of the new table might be compromised.

### **Related information:**

Nickname column and index names

Changing applications for migration

---

### db2ls - List installed DB2 products and features

Lists the DB2 products and features installed on your Linux and UNIX systems, including the DB2 HTML documentation.

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.

The **db2ls** command can be found both in the installation media and in a DB2 install copy on the system. The **db2ls** command can be run from either location. The **db2ls** command can be run from the installation media for all products except IBM Data Server Driver Package.

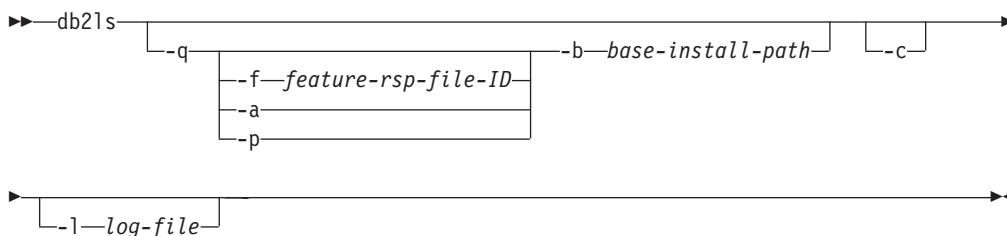
#### Authorization

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 nonzero, 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.



## db2ls - List installed DB2 products and features

- 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.
- l *log-file*  
Trace log file 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/V10.1`
- To see all DB2 database features installed to a particular path, issue:  
`db2ls -q -a -b /opt/ibm/ese/V10.1`
- To check whether a specific DB2 database feature is installed or not, issue:  
`db2ls -q -b /opt/ibm/ese/V10.1 -f feature`

### Usage notes

- You cannot use the **db2ls** command on Windows operating systems.
- If the root has write permission in `/usr/local/bin` or is able to create `/usr/local/bin`, the symbolic link `/usr/local/bin/db2ls` will be created which points to `DB2DIR/install/db2ls` for the first installation of DB2 Version 9 or later version installed on the system. The root will update the link pointing to the highest version and level of DB2 installed on the system, if multiple copies of DB2 are installed.  
A non-root installation will not create or change the `/usr/local/bin/db2ls`. In that case, to run **db2ls**, you have to do one of two things:
  - Add `inst_home/sqllib/install` to the user's path. Then you can run **db2ls** as the non-root user.
  - Pass in the exact path of the command, for example `inst_home/sqllib/install/db2ls`.
- The **db2ls** command is the only method to query a DB2 product at Version 9 or later. 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.
- Different feature listings are obtained depending upon the root versus non-root method of DB2 installation and the user running the command.

Without the **-q** option:

- For any user, other than the non-root-install instance user, the command displays all copies installed by the root user.
- For the non-root-install instance user, the command displays all DB2 copies installed by the root user plus the non-root copy owned by the non-root user.

With the **-q** option:

- If userA wants to know if userB has DB2 installed, userA can run `db2ls -q -b $userBHomeDir/sqllib`. If userA has access permission, then the DB2 features installed by userB will be displayed, otherwise, an error message will be returned indicating that access permission was denied.
- If you run `db2ls -q` without the **-b** option, the installed features in the install path where **db2ls** belongs are displayed.
- If the directory is read-only, the **db2ls** command cannot be linked to from the `/usr/local/bin` directory. If you are running in a system workload partitions

## **db2ls - List installed DB2 products and features**

(WPARs) you can use the **db2ls** command located in the installation image root directory to query a list of installed copies.

## db2move - Database movement tool

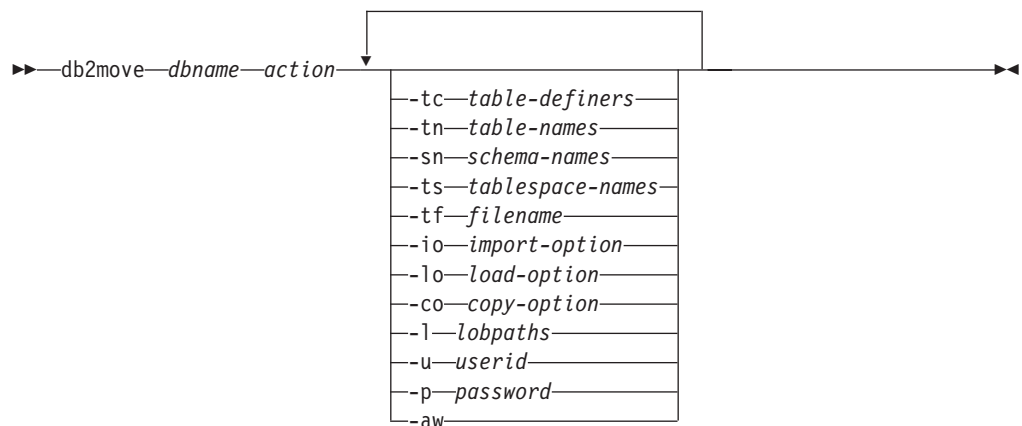
This tool, when used in the EXPORT, IMPORT, or LOAD mode, facilitates the movement of large numbers of tables between DB2 databases located on workstations. When used in the COPY mode, this tool facilitates the duplication of a schema.

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.

### 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 schemas into a target database. The target database must be a local database. Use the **-sn** option to specify one or more

## db2move - Database movement tool

schemas. See the **-co** option for COPY specific options. Use the **-tn** or **-tf** option to filter tables in LOAD\_ONLY mode. You must use a table space named SYSTOOLSPACE when either the ADMIN\_COPY\_SCHEMA() stored procedure is used, or when the **db2move** utility is used with the **-COPY** option.

See the following section 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 with the EXPORT action, only those tables whose names match those in the specified string are exported. 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. Table names should be listed unqualified and the **-sn** option should be used to filter schemas.

For export, an asterisk (\*) can be used as a wildcard character that can be placed anywhere in the string.

If specified with the COPY action, the **-co "MODE" LOAD\_ONLY** *copy-option* must also be specified, and only those tables specified will be repopulated on the target database. The table names should be listed with their schema qualifier in the format "schema"."table".

### **-sn** *schema-names*

The default for EXPORT is all schemas (not for COPY).

If specified, only those tables whose schema names match 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 characters 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*

If specified with EXPORT action, only those tables whose names match exactly those in the specified file are exported. If not specified, the default is to use all user tables. The tables should be listed one per line, and each table should be fully qualified. Wildcard characters are not allowed in the strings. Here is an example of the contents of a file:

```
"SCHEMA1"."TABLE NAME1"
"SCHEMA NAME77"."TABLE155"
```

If specified with the COPY action, the **-co "MODE" LOAD\_ONLY** *copy-option* must also be specified, and only those tables specified in the file will be repopulated on the target database. The table names should be listed with their schema qualifier in the format "schema"."table".

**-io** *import-option*

The default is REPLACE\_CREATE. See "IMPORT command options CREATE and REPLACE\_CREATE are deprecated" for limitations of import create function.

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, userid and password. (The source database userid and 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 is not specified, then **db2move** will prompt for the password information. The reason for prompting is for security reasons discussed in the following section. **TARGET\_DB** is a mandatory option for the COPY action. The **TARGET\_DB** cannot be the same as the source database and must be a local 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

## db2move - Database movement tool

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. The LOAD\_ONLY mode requires inputting at least one table using the **-tn** or **-tf** option.

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. For example: 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-dependent 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.

If a target schema already exists, the utility will fail. Use the ADMIN\_DROP\_SCHEMA procedure to drop the schema and all objects associated with that schema.

### “NONRECOVERABLE”

This option allows the user to override the default behavior of the load to be done with COPY-NO. With the default behavior, the

user will be forced to take backups of each table space that was loaded into. When specifying this **NONRECOVERABLE** keyword, the user will not be forced to take backups of the table spaces 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 is an optional option available to the COPY action.

**"TABLESPACE\_MAP"**

The user may specify table space name mappings to be used instead of the table spaces from the source system during a copy. This will be an array of table space mappings surrounded by brackets. For example, `tablespace_map ((TS1, TS2), (TS3, TS4))`. This would mean that all objects from table space TS1 will be copied into table space TS2 on the target database and objects from table space TS3 will be copied into table space TS4 on the target. In the case of `((T1, T2), (T2, T3))`, all objects found in T1 on the source database will be re-created in T2 on the target database and any objects found in T2 on the source database will be re-created in T3 on the target database. The default is to use the same table space name as from the source, in which case, the input mapping for this table space is not necessary. If the specified table space does not exist, the copy of the objects using that table space 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 table space should be chosen using the default table space selection algorithm. In this case, **db2move** will be able to choose any available table space to be used as the target. The `SYS_ANY` keyword can be used for all table spaces, example: `tablespace_map SYS_ANY`. In addition, the user can specify specific mappings for some table spaces, and the default table space selection algorithm for the remaining. For example, `tablespace_map ((TS1, TS2), (TS3, TS4), SYS_ANY)`. This indicates that table space TS1 is mapped to TS2, TS3 is mapped to TS4, but the remaining table spaces will be using a default table space target. The `SYS_ANY` keyword is being used since it's not possible to have a table space starting with "SYS".

This is an optional option available to the COPY action.

**"PARALLEL" *number-of-threads***

Specify this option to have the load operations for the tables in the schema(s) spread across a number of threads. The value range for *number-of-threads* is 0-16

- If PARALLEL is not specified, no threads are used and the load operations are performed serially.
- If PARALLEL is specified without a number of threads, the **db2move** utility will choose an appropriate value.

## db2move - Database movement tool

- If PARALLEL is specified and *number-of-threads* is provided, the specified number of threads is used. If *number-of-threads* is 0 or 1, the load operation is performed serially.
- The maximum value that can be specified for *number-of-threads* is 16.

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 (such as data truncation) the user might want 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 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 table space 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

- When copying one or more schemas into a target database the schemas must be independent of each other. If not, some of the objects might not be copied successfully into the target database
- Loading data into tables containing XML columns is only supported for the **LOAD** and not for the **COPY** action. 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.
- A **db2move EXPORT**, followed by a **db2move IMPORT** or **db2move LOAD**, facilitates the movement of table data. It is necessary to manually move all other database objects associated with the tables (such as aliases, views, or triggers) as well as objects that these tables may depend on (such as user-defined types or user-defined functions).
- If the **IMPORT** action with the **CREATE** or **REPLACE\_CREATE** option is used to create the tables on the target database (both options are deprecated and may be removed in a future release), then the limitations outlined in “Imported table re-creation” 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 whether the errors might be the result of the limitations on table creation.
- Tables that contain GENERATED ALWAYS identity columns cannot be imported or loaded using **db2move**. You can, however, manually import or load these tables. For more information, see “Identity column load considerations” or “Identity column import considerations”.
- When export, import, or load APIs are called by **db2move**, the **FileTypeMod** parameter is set to lobsinfile. That is, LOB data is kept in files that are 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 using **db2move LOAD** and **logarchmeth1** is enabled for the database (the database is recoverable):
  - If the **NONRECOVERABLE** option is not specified, then **db2move** will invoke the db2Load API using the default **COPY NO** option, and the table spaces where the loaded tables reside are placed in the Backup Pending state upon

## db2move - Database movement tool

completion of the utility (a full database or table space backup is required to take the table spaces out of the Backup Pending state).

- If the **NONRECOVERABLE** option is specified, the table spaces are not placed in backup-pending state, however if rollforward recovery is performed later, the table is marked inaccessible and it must be dropped. For more information about load recoverability options, see “Options for improving load performance”.
- Performance for the **db2move** command with the **IMPORT** or **LOAD** actions can be improved by altering the default buffer pool, **IBMDEFAULTBP**, and by updating the configuration parameters **sortheap**, **util\_heap\_sz**, **logfilsiz**, and **logprimary**.
- When running data movement utilities such as **export** and **db2move**, the query compiler might determine that the underlying query will run more efficiently against an MQT than the base table or tables. In this case, the query will execute against a refresh deferred MQT, and the result of the utilities might not accurately represent the data in the underlying table.
- The **db2move** command is not available with DB2 clients. If you issue the **db2move** command from a client machine, you will receive a db2move is not recognized as an internal or external command, operable program or batch file error message. To avoid this issue, you can issue the **db2move** command directly on the server.

### Files Required/Generated When Using EXPORT

- Input: None.
- Output:

#### **EXPORT.out**

The summarized result of the EXPORT action.

#### **db2move.lst**

The list of original table names, their corresponding PC/IXF file names (tabnnn.ixf), and message file names (tabnnn.msg). This list, the exported PC/IXF files, and LOB files (tabnnnc.yyy) are used as input to the **db2move** IMPORT or LOAD action.

#### **tabnnn.ixf**

The exported PC/IXF file of a specific table.

#### **tabnnn.msg**

The export message file of the corresponding table.

#### **tabnnnc.yyy**

The exported LOB files of a specific table.

*nnn* is the table number. *c* is a letter of the alphabet. *yyy* is a number ranging from 001 to 999.

These files are created only if the table being exported contains LOB data. If created, these LOB files are placed in the *lobpath* directories. There are a total of 26,000 possible names for the LOB files.

#### **system.msg**

The message file containing system messages for creating or deleting file or directory commands. This is only used if the action is EXPORT, and a LOB path is specified.

### Files Required/Generated When Using IMPORT

- Input:

**db2move.lst**

An output file from the EXPORT action.

**tab<sup>nnn</sup>.ixf**

An output file from the EXPORT action.

**tab<sup>nnnc</sup>.yyy**

An output file from the EXPORT action.

• Output:

**IMPORT.out**

The summarized result of the IMPORT action.

**tab<sup>nnn</sup>.msg**

The import message file of the corresponding table.

**Files Required/Generated When Using LOAD**

• Input:

**db2move.lst**

An output file from the EXPORT action.

**tab<sup>nnn</sup>.ixf**

An output file from the EXPORT action.

**tab<sup>nnnc</sup>.yyy**

An output file from the EXPORT action.

• Output:

**LOAD.out**

The summarized result of the LOAD action.

**tab<sup>nnn</sup>.msg**

The **LOAD** message file of the corresponding table.

**Files Required/Generated When Using COPY**

• Input: None

• Output:

**COPYSHEMA.msg**

An output file containing messages generated during the COPY operation.

**COPYSHEMA.err**

An output file containing an error message for each error encountered during the COPY operation, including DDL statements for each object which could not be re-created on the target database.

**LOADTABLE.msg**

An output file containing messages generated by each invocation of the Load utility (used to repopulate data on the target database).

**LOADTABLE.err**

An output file containing the names of tables that either encountered a failure during Load or still need to be populated on the target database. See the “Restarting a failed copy schema operation” topic for more details.

These files are timestamped and all files that are generated from one run will have the same timestamp.

## db2mq1sn - MQListener

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.

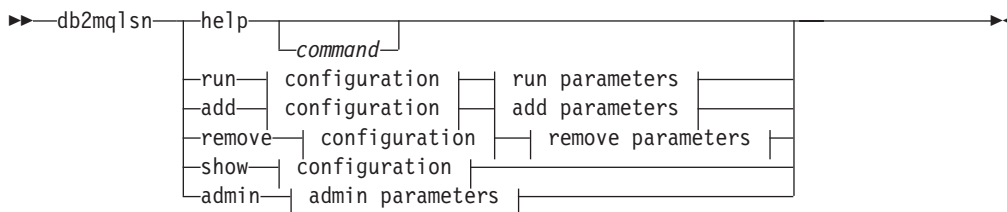
On Windows operating systems, this utility is located in the *DB2PATH\sql1lib\bin* directory, where *DB2PATH* is the location where the current version of the DB2 database product is installed.

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 **db2mq1sn 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 **db2mq1sn run** and **db2mq1sn admin** options, the user who executes the program must be able to open the appropriate MQ objects.
- To execute the **db2mq1sn run** option successfully, the dbUser specified in the **db2mq1sn 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-->>>
  
```

```

└─┬─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 **shutdown** 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 add -configDB sampleDB -config nightlies -inputQueue app3  
-procSchema imauser -procName proc3 -dbName aDB -dbUser imauser -dbPwd aSecret
```

```
db2mq1sn run -configDB -config nightlies
```

---

### 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] [-l user_name] [-p password]
```

#### Command parameters

##### -f *input\_file*

Specifies the input file to be used by the MSCS utility. If this parameter is specified, **db2mscs** utility will use filename as the input file, if this parameter is not specified, the **db2mscs** utility will try to use 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*.

##### -l *user\_name*

Specifies the user name of the domain account for the DB2 service. If you specify this parameter and the **DB2\_LOGON\_USERNAME** parameter in the DB2MSCS.CFG file is already specified, the value for the **DB2\_LOGON\_USERNAME** parameter is ignored. If neither of the parameters are specified, the DB2 service is created under the local administrator account on the remote machine.

If you are concerned about security, use the **-l** parameter. Do not specify the **DB2\_LOGON\_USERNAME** parameter.

##### -p *password*

Specifies the password of the domain account for the DB2 service. If you specify this parameter and the **DB2\_LOGON\_PASSWORD** parameter in the DB2MSCS.CFG file is already specified, the value for the **DB2\_LOGON\_PASSWORD** parameter is ignored. If neither of the parameters are specified, you will be prompted for a password.

If you are concerned about security, use the **-p** parameter. Do not specify the **DB2\_LOGON\_PASSWORD** parameter.

#### Usage notes

The **db2mscs** utility is a stand-alone 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 database product installation 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. It is required that **DB2\_INSTANCE** must be stopped before running **db2mscs**. The maximum length for this parameter is 8 bytes.

### **DAS\_INSTANCE**

The name of the DB2 Admin Server instance. Specify this parameter to 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. The maximum length for this parameter is 8 bytes.

### **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. The maximum length for this parameter is 15 bytes.

### **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. If this parameter is not specified, the DB2 service will be created under the local administrator account on the remote machine. The maximum length for this parameter is 256 bytes.

### **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. If **DB2\_LOGON\_USERNAME** is specified and **DB2\_LOGON\_PASSWORD** is not specified, you will be prompted for a password. The maximum length for this parameter is 256 bytes.

### **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. A MSCS group can be created within a MSCS Cluster. The maximum length for this parameter is 15 bytes.

### **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.

## db2mscs - Set up Windows failover utility

This parameter is required for a multi-partitioned database environment. The maximum length for this parameter is 15 bytes.

### **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. The maximum length for this parameter is 256 bytes.

### **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. The maximum length for this parameter is 15 bytes.

### **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. The maximum length for this parameter is 15 bytes.

### **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. The maximum length for this parameter is 256 bytes.

### **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, on which DB2 instance directory resides, in a partitioned database environment. The maximum length for this parameter is 256 bytes.

### **NETNAME\_VALUE**

The value for the Network Name resource. This parameter must be specified if the **NETNAME\_NAME** parameter is specified. The maximum length for this parameter is 256 bytes.

### **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. The maximum length for this parameter is 256 bytes.

### **SERVICE\_DISPLAY\_NAME**

The display name of the Generic Service resource. Specify this parameter if you want to create a Generic Service resource. The maximum length for this parameter is 256 bytes.

### **SERVICE\_NAME**

The service name of the Generic Service resource. This parameter must be specified if the **SERVICE\_DISPLAY\_NAME** parameter is specified. The maximum length for this parameter is 256 bytes.

### SERVICE\_STARTUP

Optional startup parameter for the Generic Resource service. The maximum length for this parameter is 256 bytes.

### 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 **db2mcs** 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. The maximum length for this parameter is 256 bytes.

### INSTPROF\_DISK

An optional parameter to specify an MSCS disk to contain the DB2 instance directory. If this parameter is not specified the **db2mcs** utility uses the first disk that belongs to the same group. The maximum length for this parameter is 256 bytes.

### 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. The maximum length for this parameter is 256 bytes.

### TARGET\_DRVMAP\_DISK

An optional parameter to specify the target MSCS disk for database drive mapping for a the multi-partitioned database environment. 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. The maximum length for this parameter is 256 bytes.

### 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.

---

### db2mtrk - Memory tracker

Provides complete report of memory status, for instances, databases, agents, and applications.

This command outputs the following memory pool allocation information:

- Currentsize
- 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)
- Application

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. Apart from when instance level information is returned, the command returns information only for that database partition. This command does not return information for remote servers.

#### Authorization

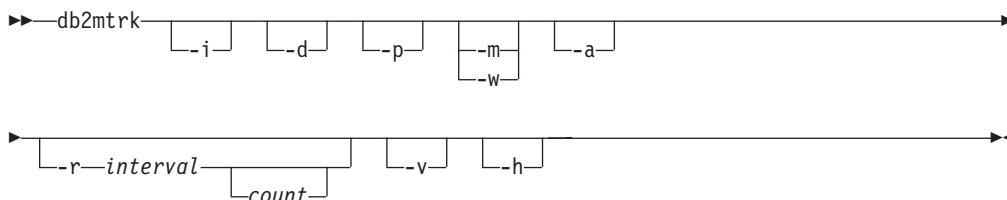
One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT
- SYSMON

#### 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.
- `-a` Show application memory usage.
- `-p` Deprecated. Show private memory.  
Replaced with `-a` parameter to show application memory usage.

- 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.
- 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` 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       apph (13)  appshrh
128.0K   64.0K    9.6M      4.8M     64.0K    256.0K
```

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       apph (26)  appshrh
128.0K   64.0K    9.6M      4.8M     64.0K    256.0K
```

The command `db2mtrk -a -i -d` displays the following output:

Tracking Memory on: 2007/01/15 at 11:30:38

Memory for instance

```
other    monh    fcmbp
11.5M    64.0K   640.0K
```

Memory for database: SAMPLE

```
utilh    pckcacheh  other    catcacheh  bph (1)  bph (S32K)  bph (S16K)
64.0K    1.0M      576.0K   448.0K    1.3M     832.0K     576.0K

bph (S8K) bph (S4K) shsorth  lockh     dbh       apph (12)  apph (11)
448.0K   384.0K   192.0K   320.0K    10.4M    64.0K     64.0K
```

## db2mtrk - Memory tracker

```
apph (10) apph (9) apph (8)
64.0K      64.0K      64.0K
```

Application Memory for database: SAMPLE

```
appshrh
256.0K
```

Memory for application 11

```
apph      other
64.0K     64.0K
```

Memory for application 10

```
apph      other
64.0K     64.0K
```

Memory for application 9

```
apph      other
64.0K     64.0K
```

Memory for application 8

```
apph      other
64.0K     448.0K
```

The command `db2mtrk -a -v -i -d` displays the following output:

Tracking Memory on: 2007/01/15 at 11:22:56

Memory for instance

```
Other Memory is of size 12058624 bytes
Database Monitor Heap is of size 65536 bytes
FCMBP Heap is of size 655360 bytes
Total: 12779520 bytes
```

Memory for database: SAMPLE

```
Backup/Restore/Util Heap is of size 65536 bytes
Package Cache is of size 1048576 bytes
Other Memory is of size 589824 bytes
Catalog Cache Heap is of size 458752 bytes
Buffer Pool Heap (1) is of size 1376256 bytes
Buffer Pool Heap (System 32k buffer pool) is of size 851968 bytes
Buffer Pool Heap (System 16k buffer pool) is of size 589824 bytes
Buffer Pool Heap (System 8k buffer pool) is of size 458752 bytes
Buffer Pool Heap (System 4k buffer pool) is of size 393216 bytes
Shared Sort Heap is of size 196608 bytes
Lock Manager Heap is of size 327680 bytes
Database Heap is of size 10944512 bytes
Application Heap (12) is of size 65536 bytes
Application Heap (11) is of size 65536 bytes
Application Heap (10) is of size 65536 bytes
Application Heap (9) is of size 65536 bytes
Application Heap (8) is of size 65536 bytes
Applications Shared Heap is of size 524288 bytes
Total: 18153472 bytes
```

Application Memory for database: SAMPLE

```
Applications Shared Heap is of size 524288 bytes
Total: 524288 bytes
```

Memory for application 11

Application Heap is of size 65536 bytes  
 Other Memory is of size 65536 bytes  
 Total: 131072 bytes

Memory for application 10

Application Heap is of size 65536 bytes  
 Other Memory is of size 65536 bytes  
 Total: 131072 bytes

Memory for application 9

Application Heap is of size 65536 bytes  
 Other Memory is of size 65536 bytes  
 Total: 131072 bytes

Memory for application 8

Application Heap is of size 65536 bytes  
 Other Memory is of size 458752 bytes  
 Total: 524288 bytes

Total: 1441792 bytes

## Usage notes

### Note:

1. When no flags are specified, usage is returned.
2. One of the **-d**, **-h**, **-i**, **-p** or **-a** flag must be specified.
3. When the **-p** parameter is specified, detailed private memory usage information is returned, grouped by agent ID.
4. When the **-a** parameter is specified, detailed application memory usage information is returned, grouped by application ID.
5. The "Other Memory" reported is the memory associated with the usage of operating the database management system.
6. 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.
7. 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.
8. For application heaps, the number specified in parentheses is the application ID.
9. 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.
10. The bufferpool heaps are always fully allocated so the memory tracker will report the same values for the current and maximum sizes of these heaps. If a

## db2mtrk - Memory tracker

bufferpool size is set to automatic then the current and maximum size of the bufferpool heap will be adjusted over time based on workload and available memory.



## db2nchg - Change database partition server configuration

Modifies database partition server configuration. This includes moving the database partition server 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.

This command can only be used if the database partition server is stopped.

This command is available on Windows operating systems only.

### Authorization

Local Administrator

### Command syntax

```

▶▶ db2nchg /n:—dbpartitionnum _____
                                   | /i:—instance_name |
_____
| /u:—username,password | | /p:—logical_port | | /h:—host_name |
_____
| /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

## db2nchg - Change database partition server configuration

partition server when there are multiple IP addresses on a machine. The network name or the IP address can be entered.

### 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
```

## db2ncrt - Add database partition server to an instance

Adds a database partition server 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 environment.

### 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:username,password**

Specifies the 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 /h:TEST
```

## db2ndrop - Drop database partition server from an instance

Drops a database partition server 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 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 dbpartitionnum** command should be used. This ensures that the database partition server is correctly removed from the partition database environment. 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 environment.

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.

---

### db2nrcfg - Non-root install configuration tool

Configuration tool used for non-root installations of DB2.

#### Authorization

Non-root ID who owns the non-root installation.

#### Required Connection

None

#### Command syntax

```
db2nrcfg [-a AuthType] [-d] [-p PortName] [-s InstType] [-j "TEXT_SEARCH ,portnumber"] [-h | -?]
```

#### Command parameters

- a *AuthType*  
Sets the authentication type (SERVER, CLIENT, or SERVER\_ENCRYPT) for the instance.
- d Turns debug mode ON.
- p *PortName*  
Sets the port name or port number to be used by this instance.
- s *InstType*  
Sets the type of instance to be created (wse, ese, or client).
- j "TEXT\_SEARCH"  
Configures the DB2 Text Search server using generated default values for service name and TCP/IP port number. This parameter cannot be used if the instance type is client.
  - j "TEXT\_SEARCH, *portnumber*"  
Configures the DB2 Text Search server using a default service name and the provided port number. Valid port numbers must be within the 1024 - 65535 range.
- h | -?  
Displays help information.

#### Usage notes

This command is automatically run by DB2 installer during non-root installation.

The **db2icrt**, **db2iupdt**, and **db2iupgrade** commands, that are available for root install, are not be available in non-root install.

## db2nrupdt - Non-root-installed instance update

Update tool used for instances created by non-root installations of DB2.

### Authorization

Non-root ID who owns the non-root installation.

### Required Connection

None

### Command syntax

```

db2nrupdt [-a AuthType] [-d] [-k] [-j "TEXT_SEARCH ,portnumber"] [-h] [-?]

```

### Command parameters

#### -a *AuthType*

Sets the authentication type (SERVER, CLIENT, or SERVER\_ENCRYPT) for the instance.

-d Turns debug mode ON.

-k

Keeps the current instance type during the update.

#### -j "TEXT\_SEARCH"

Configures the DB2 Text Search server using generated default values for service name and TCP/IP port number. This parameter cannot be used if the instance type is client.

#### -j "TEXT\_SEARCH, portnumber"

Configures the DB2 Text Search server using a default service name and the provided port number. Valid port numbers must be within the 1024 - 65535 range.

-h | -?

Displays help information.

### Usage notes

The **db2icrt**, **db2iupdt**, and **db2iupgrade** commands, that are used by root install, are not available in non-root install.

---

### db2nrupgrade - Upgrade non-root instance

Upgrades a non-root instance from a previous version of the DB2 database system to the current version of the DB2 copy from where you are running the **db2nrupgrade** command.

This command is only available on Linux and UNIX operating systems.

This command is located in the *DB2DIR/instance* directory, where *DB2DIR* represents the installation location where the new release of the DB2 database system is installed. This command does not support instance upgrade for a root installation.

#### Authorization

Non-root ID who owns the non-root installation copy.

#### Command syntax

```
db2nrupgrade [-d] [-a AuthType] [-b backup_dir] [-j "TEXT_SEARCH [,portnumber]"] [-h | -?]
```

#### Command parameters

- d** Turns on debug mode. 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.
- b backup\_dir**  
This parameter is mandatory. Specifies the directory where the configuration files from the old DB2 version are stored. The backup directory is in the **db2setup** log in the format of `sqllib_vVR` where *V* is the version number and *R* is the release number of the old copy. For example, if you have V9.5 installed and then install V9.7 using the **db2setup** command, you can find the name of the backup directory as `sqllib_v95` in the **db2setup** log file.
- j "TEXT\_SEARCH"**  
Configures the DB2 Text Search server using generated default values for service name and TCP/IP port number. This parameter cannot be used if the instance type is client.
  - j "TEXT\_SEARCH, portnumber"**  
Configures the DB2 Text Search server using a default service name and the provided port number. Valid port numbers must be within the 1024 - 65535 range.
- h | -?** Displays help information.

#### Usage notes

- This command is run automatically during the copy upgrade. You do not need to manually run this command unless the copy upgrade failed.



## 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 databases on the Solaris operating system and for 64-bit DB2 instances on HP-UX.

### Authorization

- On DB2 database for HP-UX, no authorization is required. To make the changes recommended by the **db2osconf** utility, you must have root user authority.
- On DB2 database for Solaris operating systems, you must have root user authority 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 for DB2 database systems on Solaris operating systems.
- f 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 Solaris operating systems, 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 Lists the current kernel parameters.
- m Overrides the amount of physical memory in GB. Normally, the **db2osconf** utility determines the amount of physical memory automatically. This option is available only for DB2 database systems on Solaris operating systems.
- n Overrides the number of CPUs on the system. Normally, the **db2osconf** utility determines the number of CPUs automatically. This option is available only for DB2 database systems on Solaris operating systems.

## db2osconf - Utility for kernel parameter values

- p 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 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 Provides recommendations for `semsys:seminfo_semume` and `shmsys:shminfo_shmseg` kernel parameter values. This option is available only for DB2 database systems on Solaris operating systems. For multithreaded 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 multithreaded 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 section 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 multithreaded application is a local application and has *X* number of connections to DB2 databases, then that application (process) will need *X* number of shared memory segments and *X* number of the semaphore undo structures to communicate with the DB2 database system. 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 disproportionately low, the recommendations will be based on the lower of the two.

## Example

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
```

## db2osconf - Utility for kernel parameter values

```
set msgsys:msginfo_msgmni = 2560
set msgsys:msginfo_msgtql = 2560
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 database 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, the DB2 database manager 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 the DB2 database manager 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 4 GB of RAM and 4 CPUs, the amount of memory for the recommended kernel parameters is 4.67 MB or 0.11%. This small fraction of memory used for the kernel parameters should be acceptable given the benefits.

On Solaris operating systems, 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
```

## db2pd - Monitor and troubleshoot DB2 database

Retrieves information from the DB2 database system memory sets.

### Authorization

One of the following authority levels is required:

- The SYSADM authority level.
- The SYSCTRL authority level.
- The SYSMANT authority level.
- The SYSMON authority level.

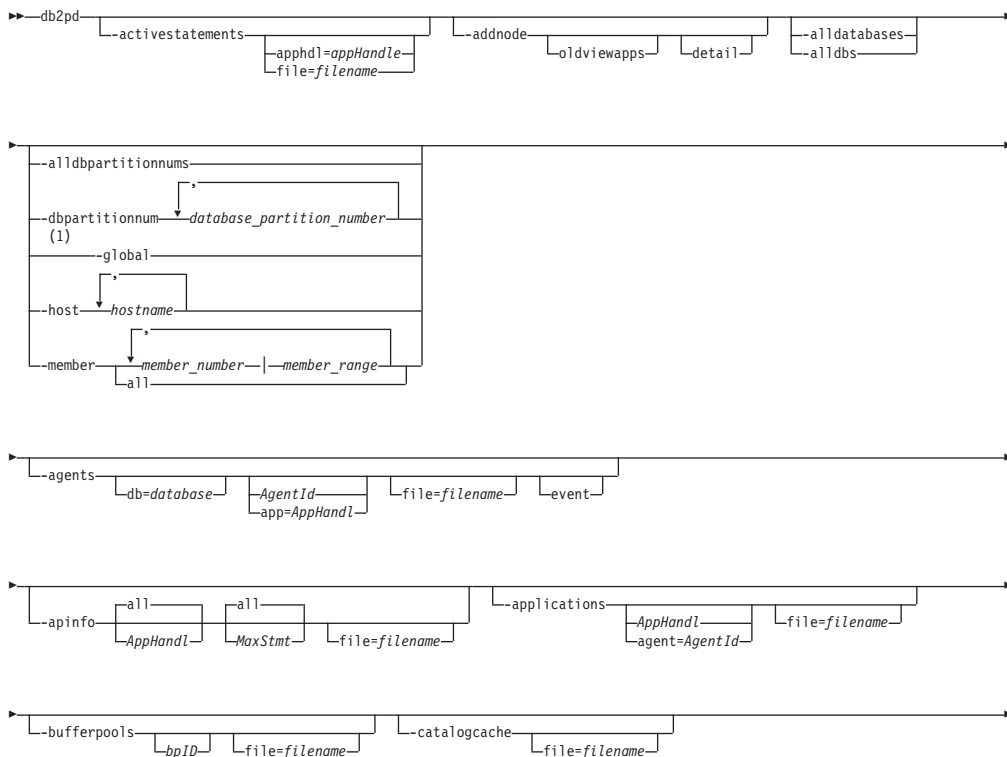
When the SYSMON authority level is granted, the following options are not available:

- **dump**
- **memblocks**
- **stack**

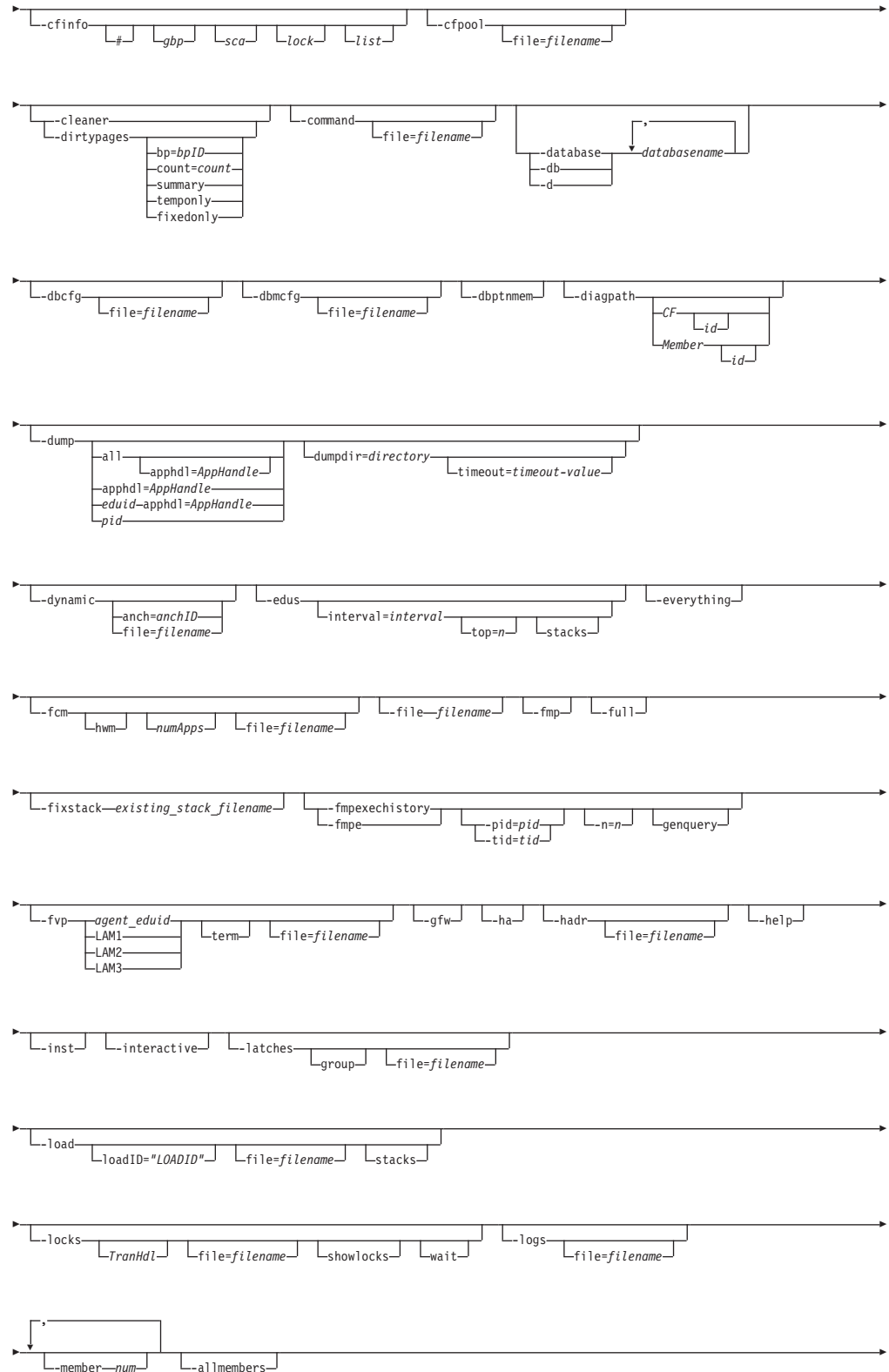
### Required connection

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.

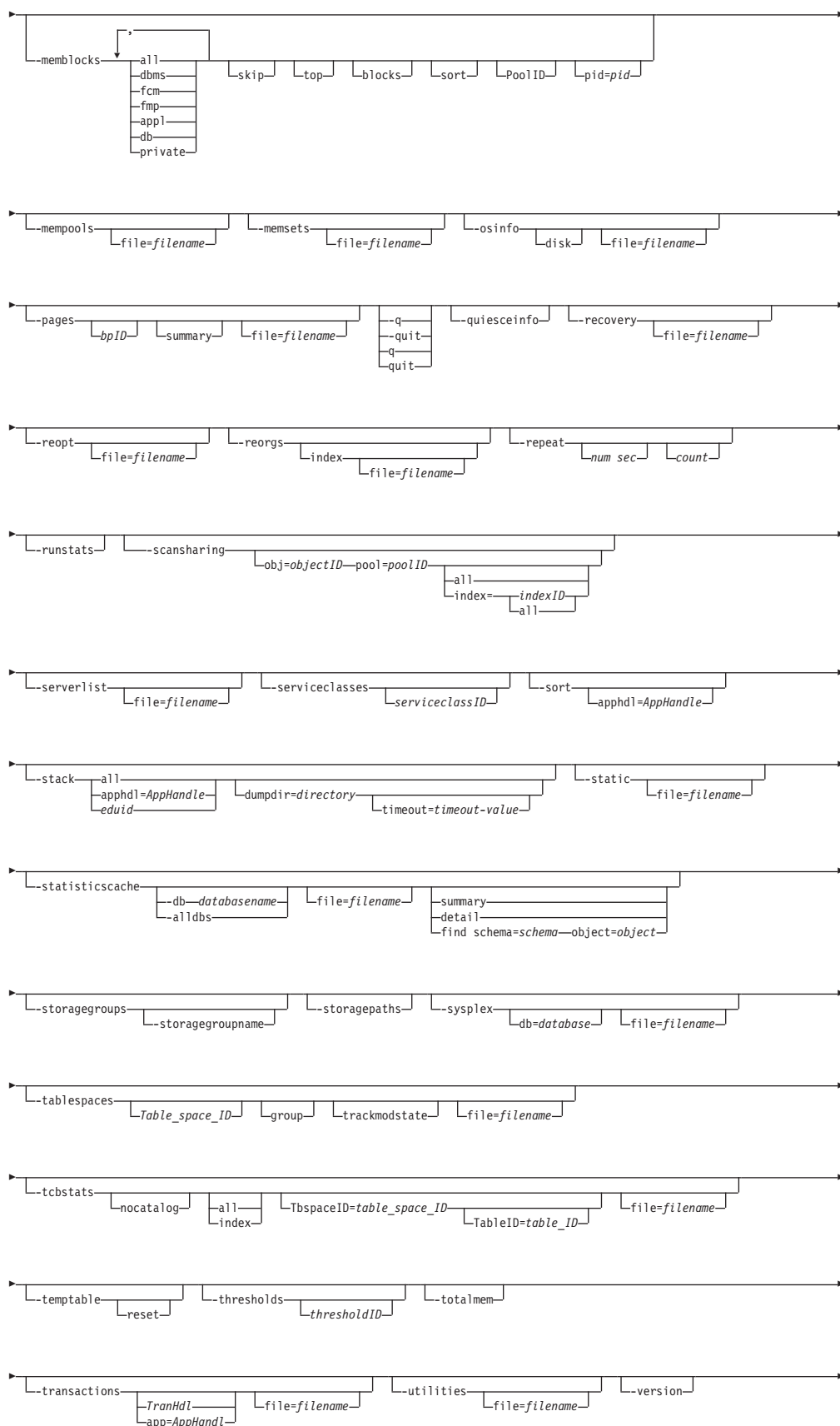
### Command syntax

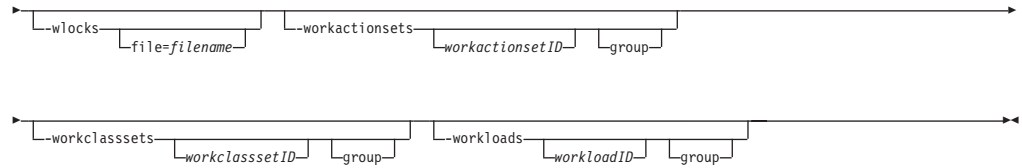


## db2pd - Monitor and troubleshoot DB2 database



## db2pd - Monitor and troubleshoot DB2 database





### Notes:

- 1 The **-global** parameter has been deprecated. You can use the **-member all** parameter options to obtain information globally.

## Command parameters

### **-activestatements**

Returns information about the active statement list.

**apphdl**=*appHandle*

If an application handle is specified, information is returned about that particular application.

**file**=*filename*

Sends the **-activestatements** output to a specified file.

See the **-activestatements** usage notes.

### **-addnode**

Returns progress information about the add database partition server operation. This parameter only returns information when issued on the database partition server that is being added. The progress information is persistent on the new database partition server until it is restarted. If issued on an existing database partition server, this parameter returns no information.

See Sample output of the **db2pd -addnode** command.

### **-alldatabases** | **-alldb**s

Specifies that the command attaches to all memory sets of all the databases.

### **-alldbpartitionnums**

Specifies that this command is to run on all active database partition servers on the local host. This parameter reports only information from database partition servers on the same physical machine that **db2pd** is being run on.

### **-allmembers**

Specifies that this command is to run on all active members for a DB2 pureScale environment. **db2pd** will only report information from database members on the same physical machine that **db2pd** is being run on.

### **-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 command parameter with the **-inst** parameter, if you have chosen a database that you want scope output for.

## db2pd - Monitor and troubleshoot DB2 database

### event

This option returns metrics for the event being processed by the agent. The metrics returned include the last time that the event was changed, the state of the event, the type of event, the object of the event, and the event object name.

See the agents usage notes.

### -apinfo

Displays the detailed information about applications including the execution of dynamic SQL statements of the current unit of work (UOW), if it is applicable.

#### *AppHandl*

If an application handle is specified, information is returned about that particular application. The default is to display information for all applications running at that partition.

#### *MaxStmt*

If a number of maximum statements is specified, the information for the most recent of SQL statements, equalling the maximum number specified, is returned. The default is to display information for all the executed SQL statements.

#### **file=filename**

Sends the **-apinfo** output to a specified file.

See Sample output of the **db2pd -apinfo** command.

**Note:** To capture the past history of a unit of work (UOW) including the SQL statement text for the applications, activate a deadlock event monitor using the statement history clause. For example, use one of the following statements:

```
create event monitor testit for deadlocks with details history write to file path global
create event monitor testit for deadlocks with details history write to table
```

The CREATE EVENT MONITOR statement has additional options, such as the ability to specify the name of the table space and table into which data will be written. For details, see the CREATE EVENT MONITOR statement description. The event monitor with statement history capability affects all applications and increases the monitor heap usage by the DB2 database manager.

See the **-apinfo** usage notes.

### -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. See the **-applications** usage notes.

### -bufferpools

Returns information about the buffer pools. If a bufferpool ID is specified, information is returned about the bufferpool. See the **-bufferpools** usage notes.



## db2pd - Monitor and troubleshoot DB2 database

### **-catalogcache**

Returns information about the catalog cache, which maintains an in-memory version of statistics.

See Sample output of the **db2pd -catalogcache** command.

The output for SYSTABLES can have multiple entries for the same table (see DEPT in the output shown previously). Multiple entries correspond to a different version of the statistics for the same table. The usage lock name will be unique among the entries for the same object and soft invalid entries will be marked with an 'S'. See the **-catalogcache** usage notes.

### **-cfinfo**

Dumps CF information that can be useful when diagnosing performance and other issues. You can specify which specific structure you want information to be dumped for by using any of the following sub-options: **gbp**, **sca**, **lock**, or **list**. For example, running **db2pd -cfinfo 2 sca** will dump the SCA structure information from CF #2.

### **-cfpool**

Displays a listing of each CF connection pool entry on the current member and its status, including whether it is being used or not, the DB2 Engine Dispatchable Unit (EDU) that is using it, and the function it is being used for.

The **cfpool** option can be used to monitor command connections and to display Host Channel Adapter (HCA) port mapping information. You can use this information to validate that load balancing between HCA ports is behaving as expected. You can also use this information verify that HCA failover is working as expected (for example, draining connections from an offline connection, or reestablishing connections after the port comes back online).

In addition, information about the cluster interconnect netname of the HCA port to which the XI and lock notification connections are established is included in the output from db2pd when you use the **cfpool** option.

### **-cleaner**

Dumps page cleaner related information from a database. This option must be preceded by an active database by specifying the **-database** or **-db** option with the proper active database name.

See Sample output of the **-cleaner** option.

### **-command filename**

Specifies to read and execute the **db2pd** command options that are specified in the file.

### **-database | -db | -d databasename**

Specifies that the command attaches to the database memory sets of the specified database. Specify the database name, not the alias name.

**-dbcfg** Returns the settings of the database configuration parameters. See the **-dbcfg** usage notes.

### **-dbmcfg**

Returns the settings of the database manager configuration parameters.

Specify this option with the **-inst** command parameter, if you have chosen a database for which you want scope output. See the **-dbmcfg** usage notes.

## db2pd - Monitor and troubleshoot DB2 database

### **-dbpartitionnum** *number*

Specifies that the command is to run on the specified local or remote database partition server.

### **-dbptnmem**

Lists database partition memory statistics.

### **-diagpath**

Returns the fully resolved split diagnostic path. In a DB2 pureScale environment, this returns the diagnostic path for all members and CFs.

#### **CF | member**

Returns the diagnostic path for either the members or CFs. If you do not specify a *id*, then the diagnostic log paths for all members or CFs are returned.

### **-dirtypages**

Dumps the dirty pages from each bufferpool in the database. This option must be preceded by an active database by specifying the **-database** or **-db** option with the proper active database name.

**bpID** Specify this option to dump dirty pages from the specified bufferpool.

**count** Specify this option to dump the first *count* number of dirty pages in each bufferpool.

#### **summary**

Specify this option to dump recovery related information of each bufferpool.

#### **temponly**

Specify this option to dump temporary dirty pages from each bufferpool.

#### **fixedonly**

Specify this option to dump dirty pages that are fixed from each bufferpool.

See Sample output of the **-dirtypages** option.

**-dump** Produces stack trace and binary dump files in the **diagpath** directory. Only available on UNIX operating systems.

- Specify with the **all** command parameter to produce stack trace files and binary dump files for all agents in the current database partition.
- Specify with the **all** parameter and an **apphdl=appHandle** parameter to return all EDUs associated with the specified *appHandle* application.
- Specify with an EDU ID of *eduid* and an **apphdl=appHandle** parameter to return information about the specified EDU if it is associated the *appHandle* application.
- Specify with an **apphdl=appHandle** parameter to return just the EDU of the coordinator agent for the *appHandle* application.
- Specify with the *pid* option to produce a stack trace file and binary dump file for a specific agent.

You can specify the following parameters with the parameters mentioned previously:

#### **dumpdir=directory**

Specifies a directory where the stack files are to be redirected. An

## db2pd - Monitor and troubleshoot DB2 database

absolute path must be specified and the specified directory must exist. This option is available for UNIX and Linux operating systems only.

### **timeout=timeout-value**

Specifies the time in seconds during which the stack files are redirected to the directory specified.

### **-dynamic**

Returns information about the execution of dynamic SQL. See the **-dynamic** usage notes.

### **anch=anchID**

If an anchor identifier is specified, information is returned about that particular dynamic SQL.

### **file=filename**

Sends the **-dynamic** output to a specified file.

**-edus** Lists all EDUs in the instance. In the case of a sustained trap, specifying this option outputs the EDU Name indicating that the EDU is suspended.

### **interval=interval**

Only available on UNIX operating systems. If an interval is specified, two snapshots of the EDUs are taken, *interval* seconds apart. Two new columns are then included in the list of EDUs: USR DELTA which displays the delta of CPU user time across the *interval*, and SYS DELTA which displays the delta of CPU system time across the *interval*. If an EDU is added part way through the *interval* it is included in the list, with the delta calculated from when it was added. If an EDU is deleted part way through the *interval* it is not included in the list at all.

### **top=n**

Specifies *n* EDUs that are to be displayed, where *n* is an integer value. The EDUs that take up the max CPU time during interval specified are displayed first.

### **stacks**

Dumps the stacks for the EDUs that are displayed.

See the **-edus** usage notes. See also Sample output of the **db2pd -edus** command.

### **-everything**

Runs all options for all databases on all database partition servers that are local to the server.

### **-fcm**

Returns information about the fast communication manager.

- Specify this parameter with the **-inst** parameter, if you have chosen a database for which you want scope output.
- Specify this parameter with the **hwm** parameter, 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 parameter with the *numApps* option, to limit the maximum number of applications that the **db2pd** command reports in the current and HWM consumption statistics.

See the **-fcm** usage notes.

## db2pd - Monitor and troubleshoot DB2 database

- file** *filename*  
Specifies to write the output to the specified file.
- fmp** Returns information about the process in which the fenced routines are executed. See the **-fmp** usage notes.
- fixstack** *existing\_stack\_filename*  
Reads an existing stack file and generates a new file in the same location, with the same file name, but with an additional `.fmt` file extension. The new `.fmt` file generated will have improved symbol details on some of the frames in the stack trace if the library where the symbol is defined is available when running this command.

**Note:** Only applicable on Linux operating systems.

- fmpexechistory | -fmpe**  
Displays the fenced routine history that attempted to be loaded and executed. Note that this parameter is available starting in Fix Pack 1.
  - pid=pid**  
Displays detailed thread information about a specific fenced process ID. If none is specified, detailed information for all processes is displayed. For thread-safe FMP processes, there will be one execution history list per thread, and threads are presented in three groups: Active, Pooled, Forced. For non thread-safe FMP processes, only one execution history list per process is displayed.
  - tid=tid**  
Displays historical details for a thread-safe routine using a specific thread ID. For non thread-safe routine, the thread ID value will be 1.
  - n=n** Use this option to specify the number of routine execution history that is to be displayed for each FMP process. The maximum value is 128. If not specified, only the last routine history is returned by default.
  - genquery**  
Generates a select query that will return the routine schema, module, name and specific name according to the routine unique ID.  
  
See the **-fmpexechistory | -fmpe** usage notes.
- full** Specifies that all output is expanded to its maximum length. If not specified, output is truncated to save space on the display.
- fvp** Displays fenced vendor process information and allows the termination of a fenced vendor process in situations where it is not responding. This applies to backup, restore, prune history, load, load copy (roll forward) and Log Manager, where a vendor media device is being used.

**Note:** The **-database** *database* command parameter must be used in conjunction with this parameter in order to connect to the right memory set to gather the information.

- agent\_eduid**  
Displays the fenced vendor process information for a DB2 EDU ID of a backup, restore, prune history, load or load copy (roll forward) agent.
- LAM1** Displays fenced vendor process information for **logarchmeth1**.

## db2pd - Monitor and troubleshoot DB2 database

- LAM2** Displays fenced vendor process information for **logarchmeth2**.
- LAM3** Displays fenced vendor process information for the special case where the current log archive method configuration parameter is not set to **VENDOR**, and so a fenced vendor process needs to be created temporarily, during **ROLLFORWARD DATABASE**, to retrieve logs from a previous vendor archiving method.
- term** On top of displaying fenced vendor process information, this option also terminates the fenced vendor process specified.

**Note:** This has no affect on Windows operating systems.

### **-global**

Specifies that **db2pd** will also be run on remote hosts. If the **-file** parameter is specified, a single file consisting of all the individual files from remote host will be created on the computer from where you issued the **db2pd** command.

**Note:** This command parameter is available in DB2 Version 9.8 Fix Pack 3 and later fix packs. This parameter is deprecated in DB2 Version 9.7 Fix Pack 4 and later fix packs.

### **-dbp** *database\_partition\_number*

Specifies that **db2pd** will be run on the remote host of the specified database partition. If no database partition is specified with the **-global** option, **db2pd** will run only on the host where member resides.

- gfw** Returns a list of event monitors that are currently active or were deactivated for some reason. It also returns statistics and information about the targets into which event monitors write data for each fast writer independent coordinator.
- ha** Reports high availability statistics.
- hadr** Reports high availability disaster recovery (HADR) information. Descriptions of some of the reported element can be found in the high availability disaster recovery section of the *Database Monitoring Guide and Reference*.
- See the **-hadr** usage notes.

### **-h | -help**

Displays the online help information.

### **-host** *hostname*

Specifies the host or hosts on which the command is issued. The command is issued for all members that reside on the host. If this option is not specified, the command is issued on the local host. If multiple hosts are specified, all host names must be valid for the command to complete. This option cannot be specified together with the **-member** option.

- inst** Returns all instance-scope information.

### **-interactive**

Specifies to override the values specified for the **DB2PDOPT** environment variable when running the **db2pd** command.

### **-latches**

Reports all latch holders and all latch waiters.

**group** Just prints the list of holders followed by the list of waiters.

## db2pd - Monitor and troubleshoot DB2 database

**file=filename**

Sends **-latches** output to *filename*.

See the **-latches** usage notes.

**-load** Displays all load EDU information. This parameter can be combined with the **-host** or **-member** parameter to display host or member specific load EDU information. Requires an active database to be specified.

**loadID="LOADID"**

Displays all load EDUs working for the specific load operation specified by *LOADID*. If the *LOADID* specified does not exist, no information is displayed.

**file=filename**

Redirects the output, excluding stack files, to the specified file.

**stacks** Dumps the stack traces for the load EDUs displayed in the **diagpath** directory. If this option is used with the **loadID** option, the stacks are dumped for the load EDUs working for the specified load operation. This option is available for UNIX and Linux operating systems only.

See the **-load** usage notes.

**-locks** Returns information about the locks.

Specify a transaction handle to obtain information about the locks that are held by a specific transaction.

Specify with the **showlocks** command parameter 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** command parameter to return locks in a wait state and the owners of those locks.

See the **-locks** usage notes.

**-logs** Returns information about the log files. See the **-logs** usage notes. See also Sample output of the **db2pd -logs** command.

This information can also be obtained by running the `MON_GET_TRANSACTION_LOG` table function.

**-member** *member\_number* | *member\_range*

Specifies the member or members on which the command is issued. If this option is not specified, the command is issued on the current member. Multiple members can be specified as a comma separated list of *member\_number* (*member1*, *member2*), or using *member\_range*, where *member\_range* is a range of members (*member1*-*member3*), or using any combination of the first two methods. This option cannot be specified together with the **-host** option.

**all** Specifies that the command is issued on all members, including members on remote hosts.

**-memblocks**

Returns information about the memory sets. Certain memory sets are returned based on the scope that the **-memblocks** parameter is used in:

- If this parameter is issued with the **-inst** and **-alldbs** parameters, information about the **dbms**, **fcm**, **fmp**, **apl**, and **db** memory sets is

## db2pd - Monitor and troubleshoot DB2 database

returned. The following command returns information about the instance-scope and database-scope memory sets for all databases:

```
db2pd -inst -alldbs -memblocks
```

- If this parameter is issued with the **-inst** and **-db** parameters, information about the **dbms**, **fc**, **fmp**, **app1**, and **db** memory sets is returned for the specified database. The following command returns information about the instance-scope and database-scope memory sets for the database sample:

```
db2pd -inst -db sample -memblocks
```

- If this parameter is issued within a database scope (**-db**), information about the **app1** and **db** memory sets is returned. The following command returns information about the database-scope memory sets for the database sample:

```
db2pd -db sample -memblocks
```

- If this parameter is issued on its own with the **db2pd** command, it returns information about the instance-scope memory sets, which include the **dbms**, **fc**, and **fmp** memory sets. This is the default behavior. The following command returns the same information as `db2pd -inst -memblocks` which returns information about the instance-scope memory sets:

```
db2pd -memblocks
```

- If this parameter is issued with any of the following parameter options, the information returned is only that options memory set. The following command returns information only on the **fmp** memory set:

```
db2pd -memblocks -fmp
```

### dbms

Only report memory blocks in the database manager system memory set. This memory set is a part of instance-scope memory sets.

**fc** Only report memory blocks in the fast communication manager memory set. This memory set is a part of instance-scope memory sets.

**fmp** Only report memory blocks in the fenced mode process memory set. This memory set is a part of instance-scope memory sets.

**app1** Only report memory blocks in the application memory set. This memory set is a part of database-scope memory sets.

**db** Only report memory blocks in the database memory set. This memory set is a part of database-scope memory sets.

**all** Report memory blocks from all memory sets. This includes memory blocks from instance-scope (**-inst**) memory sets, and, on Windows operating systems only, the private memory set.

**Note:** The database scope (**-db** or **-alldbs**) must be specified to include memory blocks from database-scope memory sets (database and application memory sets).

**top** Report the top memory consumers for each set.

**blocks** Report the memory blocks for each set.

**sort** Report the sorted memory blocks for each pool in each set.

**PoolID** Report memory blocks from a specific pool.

## db2pd - Monitor and troubleshoot DB2 database

**pid=pid**

Report memory blocks from a specific process id (for UNIX operating systems only).

**private**

Report memory blocks from the private memory set (for Windows operating systems only).

**skipfreedwithpool | skip**

Report memory blocks skipping those to be freed with the pool.

See the **-memblocks** usage notes.

**-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. See the **-mempools** usage notes.

**-memsets**

Returns information about the memory sets.

Specify this command parameter with the **-inst** command parameter to include all the instance-scope information in the returned information. See the **-memsets** usage notes.

**-osinfo**

Returns operating system information. If a disk path is specified, information about the disk will be printed. See the **-osinfo** usage notes.

**-pages** Returns information about the buffer pool pages.

*bpID* If bufferpool ID is specified, only pages from the specified bufferpool are returned.

**summary**

If this option is specified, only the summary information section will be displayed.

See the **-pages** usage notes. See also Sample output of the **db2pd -pages** command.

**-q | -quit | q | quit**

Quit. When the **db2pd** keyword alone is issued, **db2pd** runs in interactive mode. The **quit** command causes an exit from this mode back to the standard command prompt.

**-quiesceinfo**

Specifies the current quiesce status of the instance and database.

**-recovery**

Returns information about recovery activity. See the **-recovery** usage notes.

**-reopt** Returns information about cached SQL statements that were re-optimized using the **REOPT ONCE** option. See the **-reopt** usage notes.

**-reorgs**

Returns information about table and data partition reorganization. When the **index** parameter is added to the command, index reorganization information is returned along with the table and data partition reorganization information.



## db2pd - Monitor and troubleshoot DB2 database

**Note:** Starting with DB2 Version 9.8 Fix Pack 3, the **db2pd -reorgs index** command reports the index reorg statistics for partitioned indexes in addition to the index reorg statistics for non-partitioned indexes reported since DB2 V9.8 Fix Pack 3.

See the **-reorgs** usage notes. See also Sample output of the **db2pd -reorgs index** command.

### **-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.

### **-runstats**

Returns information about the status of the RUNSTATS utility on table and associated indexes. The status of table statistics collection is displayed first, followed by the status of index statistics collection. Sample output of the **-runstats** option:

```
db2pd -runstats
```

```
Table Runstats Information:
```

```
Retrieval Time: 08/13/2009 20:38:20
TbspaceID: 2      TableID: 4
Schema: SCHEMA   TableName: TABLE
Status: Completed Access: Allow write
Sampling: No     Sampling Rate: -
Start Time: 08/13/2009 20:38:16 End Time: 08/13/2009 20:38:17
Total Duration: 00:00:01
Cur Count: 0          Max Count: 0
```

```
Index Runstats Information:
```

```
Retrieval Time: 08/13/2009 20:38:20
TbspaceID: 2      TableID: 4
Schema: SCHEMA   TableName: TABLE
Status: Completed Access: Allow write
Start Time: 08/13/2009 20:38:17 End Time: 08/13/2009 20:38:18
Total Duration: 00:00:01
Prev Index Duration [1]: 00:00:01
Prev Index Duration [2]: -
Prev Index Duration [3]: -
Cur Index Start: 08/13/2009 20:38:18
Cur Index: 2          Max Index: 2      Index ID: 2
Cur Count: 0          Max Count: 0
```

### **-scansharing**

Returns scan sharing information about all tables that have table or block index scan sharing in the specified database.

**obj=objectID pool=poolID**

Returns scan sharing information about the specified table.

**all** Returns scan sharing information for all tables. For each table, table or range scan sharing information is returned. In addition, for MDC tables, block index scan sharing information is returned.

**index=**

## db2pd - Monitor and troubleshoot DB2 database

*indexID*

Returns scan sharing information for the specified table, and block index scan sharing information for the specified block index.

**all** Returns table scan sharing information for the specified table, and block index scan sharing information for all block indexes.

See Sample output of the **db2pd -scansharing** command.

See the **-scansharing** usage notes.

### **-serverlist**

Returns information about which members are available for use and the relative load of each of those members.

There are instances where no output is returned for one or more databases:

- No active databases exist
- The specified database is not active
- The specified database is active, but the server list has not yet been cached
- The **db2pd** command is run in an environment that is not a DB2 pureScale environment
- No remote client has connected to the database

See the **-serverlist** usage notes.

See Sample output of the **db2pd -serverlist** command.

### **-serviceclasses** *serviceclassID*

Returns information about the service classes for a database. *serviceclassID* is an optional parameter to retrieve information for one specific service class. If *serviceclassID* is not specified, information for all service classes is retrieved.

See the **-serviceclasses** usage notes. See also Sample output of the **db2pd -serviceclasses** command.

**-sort** Starting with Fix Pack 5, this option returns information about the application sort operation. If an application handle ID is specified, information is returned about the sort operation for the specified application.

See the **-sort** usage notes.

### **-stack** **all** | **apphd1=appHandle** | *eduid*

In case of an engine hang the stack trace file can be used to get information about the DB2 state. This command produces stack trace files in the **diagpath** directory with the naming convention *pid.tid.node.stack.txt* in the UNIX and Linux operating systems. In the Windows operating systems the EDU's will dump to a stack trace file *pid.tid.stack.bin*. Note that **-stack all** is the only option supported on the Windows operating system.

**all** Specify this option to produce stack trace files for all processes in the current database partition.

**Note:** The **all** option can require a sufficient amount of memory for it to function properly.

### **apphd1=appHandle**

Specify this option to produce a stack trace file for just the application handle equal to *appHandle*. This option is available for UNIX and Linux operating systems only.

### *eduid*

Limit output to only EDU with a specified ID. Formatted events and data attached will be dumped to the relevant *pid.tid/EDUID.node.trap.txt* trap files in the db2dump directory. This option is available for UNIX and Linux operating systems only.

Event stack will be output in the following order:

Last event (on the top of event stack)

- Event type and short description
- Customer impact
- Object identifier
- ECF ID, probe
- Top event header
- Top event qualifiers (if any)
- Top event data (if present)

First event (on the bottom of event stack)

- Event type and short description
- Customer impact
- Object identifier
- ECF ID, probe
- Bottom event header
- Bottom event qualifiers (if any)
- Bottom event data (if present)

In the preceding list, ECF ID is ECF identifier (will be formatted as *product, component, function*) and probe is a line of code or some unique number (for a function).

Event flow (recorded event “history”) will be output in the following order:

First event record

- Event type and short description
- Customer impact
- Object identifier
- ECF ID, probe
- Event header
- Object data (if not a string or integer)

Last event record

- Event type and short description
- Customer impact
- Object identifier
- ECF ID, probe
- Event header
- Object data (if not a string or integer)

## db2pd - Monitor and troubleshoot DB2 database

### **dumpdir=directory**

Specifies a directory where the stack files are to be redirected. An absolute path must be specified and the specified directory must exist. This option is available for UNIX and Linux operating systems only.

### **timeout=timeout-value**

Specifies the time in seconds during which the stack files are redirected to the directory specified.

### **-static**

Returns information about the execution of static SQL and packages. See the **-static** usage notes.

### **-statisticscache**

Returns information about the statistics cache at the database level.

#### **summary**

Summarizes statistics cache. To dump the statistics cache summary for database *sample*, issue the following command:

```
db2pd -db sample -statisticscache summary
```

**detail** Specify this option to dump detailed statistics information stored in the statistics cache for all tables with the latest statistics collected by real-time statistics gathering. To dump detailed statistics information stored in the statistics cache for all the databases, issue the following command:

```
db2pd -statisticscache detail -alldbs
```

#### **find schema=schema object=object**

Specify this option to dump the detailed statistics information for a specific table with schema as schema name and object as table name. To dump detailed statistics information for table *USER1.T1* of database *sample*, issue the following command:

```
db2pd -db sample -statisticscache find schema=USER1 object=T1
```

See the **-statisticscache** usage notes.

### **-storagegroups**

Returns information about the storage groups defined for the database.

Specify with the *Storagegroup ID* command parameter to display the information about a specific storage group and its paths.

See the **-storagegroups** usage notes. See also Sample output of the **db2pd -storagegroups** command.

### **-storagepaths**

Returns information about the automatic storage paths for all storage groups defined for the database. Unlike the **storagegroups** parameter, this parameter does not accept storage group ID as input.

See the **-storagepaths** usage notes. See also Sample output of the **db2pd -storagepaths** command.

### **-sysplex**

Returns information about the list of servers associated with the database alias indicated by the **db** parameter. If the **-database** command parameter is not specified, information is returned for all databases.

Specify this command parameter with the **-inst** command parameter, if you have chosen a database for which you want scope output.

See the **-sysplex** usage notes.

### **-tablespaces**

Returns information about the table spaces.

Specify with the **group** command parameter to display the information about the containers of a table space grouped with the table space.

Specify with the *Table\_space\_ID* command parameter to display the information about a specific table space and its containers.

Specify with the **trackmodstate** command parameter to display the state of a table space with respect to the last or next backup. This parameter requires the **trackmod** configuration parameter to be set to Yes.

See the **-tablespaces** usage notes. See also Sample output of the **db2pd -tablespaces** command.

### **-tcbstats**

Returns information about tables and indexes. The total number of updates on tables, the UDI and real-time statistics UDI counters (RTSUDI), are returned as well.

**TbpaceID**=*table\_space\_ID*

Specify this option to display the information about a specific table space.

**TableID**=*table\_ID*

Specify this option to display the information about a specific table. The **TbpaceID** option is required when using the **TableID** option.

### **nocatalog**

Specify this option to display table and index information relating to all non-catalog tables.

See the **-tcbstats** usage notes.

### **-temptable**

By default, returns the following information about temporary tables:

- **Number of Temp Tables** The total number of temporary tables created and dropped since the database manager was started or since the last reset of the counters.
- **Comp Eligible Temps** Temporary tables that the data base manager has determined is eligible for compression based on these three properties: *query type*, *minimum row size*, and *minimum expected table size*.
- **Compressed Temps** The total number of temporary tables that were actually compressed. This implies that the table has enough data so that a compression dictionary is created for the temporary table.
- **Total Stored Temp Bytes** The total number of actual row data for temporary tables that is stored on disk. This can be from both compressed and non-compressed temporary tables.
- **Total Bytes Saved** The total bytes saved by compressing rows.
- **Total Compressed Rows** A cumulative count of the number of rows that saved at least one byte using compression.
- **Total Temp Table Rows** The total number of rows inserted into all the temporary tables, whether they are compressed or not. Not all rows inserted into a compressed temporary table are necessarily compressed.

## db2pd - Monitor and troubleshoot DB2 database

### **reset**

Specify this option to reset all counters to zero.

See the **-temptable** usage notes. See also Sample output of the **db2pd -temptable** command.

### **-thresholds** *thresholdID*

Returns information about thresholds. *thresholdID* is optional, but specifying a threshold ID returns information about a specific threshold. If *thresholdID* is not specified, information for all enabled and disabled thresholds is retrieved.

See the **-thresholds** usage notes. See Sample output of the **db2pd -thresholds** command.

### **-totalmem**

Returns information about total amount of memory allocated on a DB2 host, specifically:

- the amount of reserved restart light memory preallocated on the host
- the total memory consumption by the host's resident members and guest members

The **-totalmem** option only reports information about the current host being accessed.

### **-transactions**

Returns information about active transactions. If a transaction handle is specified, information is returned about that transaction handle. If an application handle is specified, information is returned about the application handle of the transaction. See the **-transactions** usage notes.

### **-utilities**

Reports utility information. Descriptions of each reported element can be found in the utilities section of the *Database Monitoring Guide and Reference*.

See the **-utilities** usage notes.

### **-v | -version**

Displays the current version and service level of the installed DB2 database product.

### **-wlocks**

Displays the owner and waiter information for each lock being waited on. In the Sample output of the **db2pd -wlocks** command, the lock status (Sts) value of G designates the owner of the lock, while a Sts value of W designates the waiter of that lock.

**file=filename**

Sends the **-wlocks** output to a specified file.

See the **-wlocks** usage notes.

### **-workactionsets** *workactionsetID*

Returns information about all enabled work action sets and all enabled work actions in these sets.

**group** Returns the same information grouped by work action set.

See the **-workactionsets** usage notes.

## db2pd - Monitor and troubleshoot DB2 database

### **-workclassesets** *workclasssetID*

Returns information about all work class sets that are referenced by an enabled work action set and all work classes in the work class sets.

**group** Returns the same information grouped by work class set.

See Sample output of the **db2pd -workclassesets** command. See the **-workclassesets** usage notes.

### **-workloads** *workloadID*

Returns the list of workload definitions, user privilege holders, and local partition workload statistics in memory at the time the command is run.

**group** Returns the same information grouped by workload.

See Sample output of the **db2pd -workloads** command.

See the **-workloads** usage notes.

## 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
```

Use the **db2pd -fvp** command, from the command line, in the following way to obtain fenced vendor process state information:

### For Log Manager:

- A database named SAMPLE has **logarchmeth1** set to TSM. At any time issue:

```
db2pd -db sample -fvp lam1
```

The resulting output is as follows:

```
-----  
Fenced Vendor Process State Information:  
-----
```

```
Log Manager:
```

```
-----  
LOGARCHMETH1 available.
```

## db2pd - Monitor and troubleshoot DB2 database

```
Vendor EDU is available and running.  
  startTime: 1155581841 20060814145721  
  function: sqluvint
```

This tells you that the fenced vendor process is running in the vendor function sqluvint since August 14, 2006 14:57. Now, if you feel that this has been running too long, or you have determined that this process has hung waiting for TSM resources, you can terminate the fenced vendor process by issuing:

```
db2pd -db sample -fvp lam1 term
```

The resulting output is as follows:

```
-----  
Fenced Vendor Process State Information:  
-----
```

```
Log Manager:  
-----
```

```
LOGARCHMETH1 available.
```

```
Vendor EDU is available and running.  
  startTime: 1155581841 20060814145721  
  function: sqluvint
```

This fenced vendor process has been sent a signal to terminate.

This shows you the same information as the previous output, but also lets you know that the terminate request has been sent. After waiting a few moments, you should notice that the request has taken affect.

- If the fenced vendor process is running, but not running in vendor code, you will see this for a regular display request:

```
-----  
Fenced Vendor Process State Information:  
-----
```

```
Log Manager:  
-----
```

```
LOGARCHMETH1 available.
```

```
Vendor EDU is available and running.  
No vendor code being run.
```

### For Backup:

**Note:** It should be noted that the **FORCE APPLICATION** command can be used as an alternative to what is described in the following section.

- A database named SAMPLE is being backed up to TSM using 2 sessions. You need to find out the backup agent EDU ID, which can be found through **db2pd -edus** or the DB2 diagnostics log. Once found, one can issue:

```
db2pd -db sample -fvp 149
```

The resulting output is as follows:

```
-----  
Fenced Vendor Process State Information:  
-----
```

```
Backup:  
-----
```

```
Media Controller(s):  
-----
```

```
  EDU ID: 504  
mediaSession: 1  
mediaSeqNum: 0  
Vendor EDU is available and running.  
  startTime: 1155583315 20060814152155
```



```
function: sqluvint
```

```
EDU ID: 505
mediaSession: 2
mediaSeqNum: 0
Vendor EDU is available and running.
No vendor code being run.
```

This says that DB2 Media Controller 0 (EDU ID: 504) is in vendor code, while DB2 Media Controller 1 (EDU ID: 505) has a fenced vendor process, but is not running vendor code. Now, if you feel that this has been running too long, or you have determined that this process has hung waiting for TSM resources, you can terminate the fenced vendor process by issuing:

```
db2pd -db sample -fvp 149 term
```

The resulting output is as follows:

```
-----
Fenced Vendor Process State Information:
-----
```

```
Backup:
-----
```

```
Media Controller(s):
-----
```

```
EDU ID: 504
mediaSession: 1
mediaSeqNum: 0
Vendor EDU is available and running.
  startTime: 1155583315 20060814152155
  function: sqluvint
This fenced vendor process has been sent a signal to terminate.
```

```
EDU ID: 505
mediaSession: 2
mediaSeqNum: 0
Vendor EDU is available and running.
No vendor code being run.
This fenced vendor process has been sent a signal to terminate.
```

This tells you the same information as the previous output, but notes that both fenced vendor processes have been sent terminate requests and will be terminated shortly.

## Usage notes

The following sections describe the output produced by the different **db2pd** parameters.

- **-activestatements**
- **-agents**
- **-apinfo**
- **-applications**
- **-bufferpools**
- **-catalogcache**
- **-dbcfg**
- **-dbmcfg**
- **-dynamic**
- **-edus**
- **-fcm**
- **-fmp**
- **-fmpexechistory** | **-fmpe**
- **-hadr**
- **-latches**

## db2pd - Monitor and troubleshoot DB2 database

- -load
- -locks
- -logs
- -memblocks
- -mempools
- -memsets
- -osinfo
- -pages
- -recovery
- -reopt
- -reorgs
- -scansharing
- -serviceclasses
- -sort
- -static
- -statisticscache
- -storagegroups
- -storagepaths
- -sysplex
- -tablespaces
- -tcbstats
- -temptable
- -thresholds
- -transactions
- -utilities
- -wlocks
- -workactionsets
- -workclasssets
- -workloads

### **-activestatement** parameter

For the **-activestatement** parameter, the following information is returned:

**Address**

Address of the current activity.

**AppHandle**

Application handle.

**UOW-ID** UOW-ID at the start of execution.

**StmtID** The activity ID of the statement within the UOW-ID.

**AnchID** The anchor ID of the statement.

**StmtUID**

The unique ID of the statement within the anchor.

**EffISO** Effective isolation level.

**EffLockTO**

Effective lock timeout at start.

**EffDegree**

Effective SMP parallelism degree at start.

**StartTime**

The start time of when the statement was executed.

**LastRefTime**

Last application reference time.

**-agents parameter**

For the **-agents** parameter, the following information is returned:

**AppHandle**

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.

**LkTm0t** The lock timeout setting for the agent.

**LastApplID**

The outbound application ID that the pooled agent serviced last.

**LastPooled**

The timestamp when the agent was pooled.

If the event option is specified with the **-agents** parameter, the following, additional information is returned. You use this information to determine whether an agent continues to process the same task or whether the agent moves onto new tasks over time.

**AGENT\_STATE\_LAST\_UPDATE\_TIME(Tick Value)**

The last time that the event, being processed by the agent, was changed. The event currently processed by the agent is identified by the EVENT\_STATE, EVENT\_TYPE, EVENT\_OBJECT, and EVENT\_OBJECT\_NAME columns.

**EVENT\_STATE**

The state of the event last processed by this agent. The possible values are EXECUTING and IDLE.

**EVENT\_TYPE**

The type of event last processed by this agent. The possible values are ACQUIRE, PROCESS, and WAIT.

**EVENT\_OBJECT**

The object of the event last processed by this agent. The possible values are

## db2pd - Monitor and troubleshoot DB2 database

COMP\_DICT\_BUILD, IMPLICIT\_REBIND, INDEX\_RECREATE, LOCK, LOCK\_ESCALATION, QP\_QUEUE, REMOTE\_REQUEST, REQUEST, ROUTINE, and WLM\_QUEUE.

### EVENT\_OBJECT\_NAME

The event object name. If the value of EVENT\_OBJECT is LOCK, the value of this column is the name of the lock that the agent is waiting on. If the value of EVENT\_OBJECT is WLM\_QUEUE, the value of the column is the name of the WLM threshold that the agent is queued on. Otherwise, the value is NULL.

The possible combinations of EVENT\_STATE, EVENT\_TYPE, EVENT\_OBJECT and EVENT\_OBJECT\_NAME column values are listed in the following table:

Table 52. Possible combinations for EVENT\_STATE, EVENT\_TYPE, EVENT\_OBJECT and EVENT\_OBJECT\_NAME column values

Event description	EVENT_STATE value	EVENT_TYPE value	EVENT_OBJECT value	EVENT_OBJECT_NAME value
Acquire lock	IDLE	ACQUIRE	LOCK	Lock name
Escalate lock	EXECUTING	PROCESS	LOCK_ESCALATION	NULL
Process request	EXECUTING	PROCESS	REQUEST	NULL
Wait for a new request	IDLE	WAIT	REQUEST	NULL
Wait for a request to be processed at a remote partition	IDLE	WAIT	REMOTE_REQUEST	NULL
Wait on a WLM threshold queue	IDLE	WAIT	WLM_QUEUE	Threshold name
Process a routine	EXECUTING	PROCESS	ROUTINE	NULL
Re-create an index	EXECUTING	PROCESS	INDEX_RECREATE	NULL
Build compression dictionary	EXECUTING	PROCESS	COMP_DICT_BLD	NULL
Implicit rebind	EXECUTING	PROCESS	IMPLICIT_REBIND	NULL

### -apinfo parameter

For the **-apinfo** parameter, the following information is returned:

#### AppHandl

The application handle, including the node and the index.

#### Application PID

The process ID for the application.

#### Application Node Name

The name of the application node.

#### IP Address

The IP address from which the database connection was established.

#### Connection Start Time

The time stamp at which the application connection started.

#### Client User ID

The client user ID.

**System Auth ID**

This is the system authorization ID of the connection.

**Coordinator EDU ID**

The EDU ID of the coordinator agent for the application.

**Coordinator Partition**

The partition number of the coordinator agent for the application.

**Number of Agents**

The number of agents that are working on behalf of the application.

**Locks timeout value**

The lock timeout value for the application.

**Locks Escalation**

The locks escalation flag indicates whether the lock, used by the application, has been escalated.

**Workload ID**

Workload identifier.

**Workload Occurrence ID**

Workload occurrence identifier.

**Trusted Context**

The name of the trusted context associated with the connection if the connection is either an implicit trusted connection or an explicit trusted connection.

**Connection Trust Type**

The connection trust type. This is one of: non-trusted, implicit trusted, or explicit trusted connection.

**Role Inherited**

This is the role inherited through a trusted connection, if any.

**Application Status**

The status of the application.

**Application Name**

The name of the application.

**Application ID**

The application ID. This value is the same as the **appl\_id** monitor element data. For detailed information about how to interpret this value, see the "appl\_id - Application ID monitor element".

**UOW-ID** The ID of the current UOW of the application.

**Activity ID**

The activity ID within the UOW.

**Package Schema**

The package schema.

**Package Name**

The package name.

**Package Version**

The package version.

**Section Number**

The section number of the SQL statement.

## db2pd - Monitor and troubleshoot DB2 database

### SQL Type

The type of SQL: dynamic or static.

### Isolation

The isolation mode set for the application.

### Statement Type

The type of statement operation, such as: DML, DDL.

### Statement

The SQL statement.

### ClientUserID

Client userid for the transaction, which is the same as `tpmon_client_userid` (TP Monitor Client User ID monitor element).

### ClientWrkstnName

Client workstation name for the transaction, which is the same as `tpmon_client_wrkstn` (TP Monitor Client Workstation Name monitor element).

### ClientAppName

Client application name driving the transaction, which is the same as `tpmon_client_app` (TP Monitor Client Application monitor element).

### ClientAcctng

Accounting string of the client driving the transaction, which is the same as `tpmon_acc_str` (TP Monitor Client Accounting String monitor element).

See Sample output of the `db2pd -apinfo` command.

## -applications parameter

For the `-applications` parameter, the following information is returned:

### AppHandle

The application handle, including the node and the index.

### NumAgents

The number of agents that are working on behalf of the application.

### CoordPid

The process ID of the coordinator agent for the application.

**Status** The status of the application.

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

### ClientIPAddress

The IP address from which the database connection was established.

### EncryptionLvl

The data stream encryption used by the connection. This is one of NONE, LOW or HIGH. NONE implies that no data stream encryption is being used. LOW implies that the database server **authentication** type is set to DATA\_ENCRYPT. HIGH implies that SSL is being used.

### SystemAuthID

This is the system authorization ID of the connection.

**ConnTrustType**

The connection trust type. This is one of: non-trusted, implicit trusted connection, or explicit trusted connection.

**TrustedContext**

The name of the trusted context associated with the connection if the connection is either an implicit trusted connection or an explicit trusted connection.

**RoleInherited**

This is the role inherited through a trusted connection, if any.

### **-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.

## db2pd - Monitor and troubleshoot DB2 database

### **Automatic**

Shows the self-tuning automatic status. "True" means self-tuning for this buffer pool is enabled. "False" means self-tuning for this buffer pool is not enabled.

### **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.

**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

## db2pd - Monitor and troubleshoot DB2 database

index 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.

### **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.

## **-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).

**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).
- S (soft invalid. Catalog cache entries become *soft invalid* when statistics have been updated by real-time statistics collection. These catalog cache entries may still be used by a database agent, but they are not valid for use by a new catalog cache request. Once the soft invalid entry is no longer in use, it will be removed. New catalog cache requests will use the valid entry.)

**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.

## db2pd - Monitor and troubleshoot DB2 database

### **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*).

**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.

### SYSROLEAUTH:

**AuthID** The authorization identifier (*authid*).

**AuthType**  
The authorization type.

**Roleid** The role identifier if the authorization identifier is a role.

**LastRefID**  
The last process identifier to reference the cache entry.

**CatalogCache LoadingLock**  
The name of the catalog cache loading lock for the cache entry.

### TABLESPACES:

**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.

## db2pd - Monitor and troubleshoot DB2 database

- Sts** The status of the entry. The possible values are:
- V (valid).
  - I (invalid).
  - S (soft invalid. Catalog cache entries become *soft invalid* when statistics have been updated by real-time statistics collection. These catalog cache entries may still be used by a database agent, but they are not valid for use by a new catalog cache request. Once the soft invalid entry is no longer in use, it will be removed. New catalog cache requests will use the valid entry.)

See Sample output of the **db2pd -catalogcache** command.

### -dbcfg parameter

For the **-dbcfg** parameter, the current values of the database configuration parameters are returned.

### -dbmcfg parameter

For the **-dbmcfg** parameter, current values of the database manager configuration parameters are returned.

### -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.

**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.

**Val** The variation valid flag. The following are possible values:

**Y** Object is valid.

**N** Object is invalid.

**X** Object is inoperative.

**?** Object needs revalidation.

#### **Insert Time**

The time at which the variation was inserted into the package cache.

#### **Sect Size**

The length of section data.

## -edus parameter

For the **-edus** parameter, the following information is returned:

**EDU ID** The unique identifier for the engine dispatchable unit (EDU). Except on Linux operating systems, the EDU ID is mapped to the thread ID. On Linux operating system the EDU ID is a DB2 generated unique identifier.

**TID** Thread identifier. Except on Linux operating systems, the thread ID is the unique identifier for the specific thread. On Linux operating systems, this is a DB2 generated unique identifier.

#### **Kernel TID**

A unique identifier for the operating system kernel thread in service.

## db2pd - Monitor and troubleshoot DB2 database

### EDU Name

DB2 specific name for the EDU.

**USR** Total CPU user time consumed by the EDU.

**SYS** Total CPU system time consumed by the EDU.

### USR Delta

Indicates the delta of the CPU user time across a specified time interval.

### SYS Delta

Indicates the delta of the CPU system time across a specified time interval.

See Sample output of the **db2pd -edus** command.

## -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.

#### Max Buffers

Maximum number of buffers that can be allocated based on the amount of virtual memory reserved when the instance was started.

#### 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.

#### Max Channels

Maximum number of channels that can be allocated based on the amount of virtual memory reserved when the instance was started.

#### Total Sessions

Total number of sessions, including all free and in-use ones.

#### 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 `db2nodes.cfg` file 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 `db2nodes.cfg` file. This might indicate an error.
- **Unknown:** The database partition server is in an unknown state. This indicates an error.

### **Buffers Current Consumption**

#### **AppHand1**

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**

#### **AppHand1**

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**

#### **AppHand1**

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**

#### **AppHand1**

The application handle, including the node and the index.

#### **TimeStamp**

A unique identifier for the usage of an application handle.

#### **Channels Used**

The high-watermark number of channels used by an application since the start of the instance.

### -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
  - 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.

### -fmpexechistory | -fmpe parameter

For the **-fmpexechistory** | **-fmpe** parameter, the following information is returned:

#### 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
  - 0x00000004 - Used to run federated wrappers

## db2pd - Monitor and troubleshoot DB2 database

- 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 ipcs for the process have been removed
- 0x00000200 - .NET runtime initialized
- 0x00000400 - JVM initialized for debugging
- 0x00000800 - Termination flag
- ActiveThrd - Number of active threads running in the FMP process.
- PooledThrd - Number of pooled threads held by the FMP process.
- ForcedThrd - Number of forced threads generated by the FMP process.
- Active - Active state of the FMP process. Values are Yes or No.

### Active Threads:

- EduPid - EDU process ID that this thread is working.
- ThreadId - Active thread ID.
- RoutineID - The routine identifier.
- Timestamp - A unique identifier for the usage of an application handle.

### Pooled Threads:

- ThreadId - Pooled thread ID.
- RoutineID - The routine identifier.
- Timestamp - A unique identifier for the usage of an application handle.

### Forced Threads:

- ThreadId - Forced thread ID.
- RoutineID - The routine identifier.
- Timestamp - A unique identifier for the usage of an application handle.

See Sample output of the **db2pd -catalogcache** command.

## -hadr parameter

For the **-hadr** parameter, information related to high availability disaster recovery is returned. Only information relevant to the current settings is shown, so for example if reads on standby is not enabled, information about the replay-only window is not shown.

### HADR\_ROLE

The current HADR role of the local database. Possible values are:

- PRIMARY
- STANDBY

### REPLAY\_TYPE

The type of HADR replication of the database. The possible value is:

- PHYSICAL

### HADR\_SYNCMODE

The current HADR synchronization mode of the local database. Possible values are:

- ASYNC
- NEARSYNC

## db2pd - Monitor and troubleshoot DB2 database

- SUPERASYNC
- SYNC

**Note:** In multiple standby mode, the HADR\_SYNCMODE value of the standby is shown as an empty string (a zero-length string) until the primary connects to the standby database.

### STANDBY\_ID

The identifier for all the standbys in the current setup. This value has meaning only when the command is issued on the primary. If you issue it on a standby, it always returns 0 because, even in multiple standby mode, other standbys are not visible to each other. The 1 identifier is always assigned to the standby in single standby mode, whereas in multiple standby mode, 1 indicates the principal standby.

### LOG\_STREAM\_ID

The identifier for the log stream that is being shipped from the primary database.

### HADR\_STATE

The current HADR state of the database. Possible values are:

- DISCONNECTED
- DISCONNECTED\_PEER
- LOCAL\_CATCHUP
- PEER
- REMOTE\_CATCHUP
- REMOTE\_CATCHUP\_PENDING

### PRIMARY\_MEMBER\_HOST

The local host, indicated by the **hadr\_local\_host** configuration parameter, of the primary member processing the log stream.

### PRIMARY\_INSTANCE

The instance name of the primary member processing the log stream.

### PRIMARY\_MEMBER

The primary member processing the log stream.

### STANDBY\_MEMBER\_HOST

The local host, indicated by the **hadr\_local\_host** configuration parameter, of the standby member processing the log stream.

### STANDBY\_INSTANCE

The instance name of the standby member processing the log stream.

### STANDBY\_MEMBER

The standby member processing the log stream.

### HADR\_CONNECT\_STATUS

The current HADR connection status of the database. Possible values are:

- CONGESTED
- CONNECTED
- DISCONNECTED

### HADR\_CONNECT\_STATUS\_TIME

The time when the current HADR connection status began. Depending on the HADR\_CONNECT\_STATUS value, the HADR\_CONNECT\_STATUS\_TIME value indicates:

- Congestion start time

- Connection start time
- Disconnection time

### **HEARTBEAT\_INTERVAL**

The heartbeat interval in seconds, which is computed from various factors such as the values of the **hadr\_timeout** and **hadr\_peer\_window** configuration parameters. The HEARTBEAT\_INTERVAL element indicates how often the primary and standby exchange monitor information.

### **HADR\_TIMEOUT**

The time, in seconds, by which an HADR database must receive a message from its partner database. After this period of time, an HADR database server considers that the connection between the databases has failed and disconnects.

### **TIME\_SINCE\_LAST\_RECV**

The time, in seconds, that has elapsed since the last message was received, so the larger the number, the longer the delay in message delivery. When the TIME\_SINCE\_LAST\_RECV value equals the HADR\_TIMEOUT value, the connection between the databases is closed.

### **PEER\_WAIT\_LIMIT**

The length of time, in seconds, that the primary database waits before breaking out of peer state if logging is blocked waiting for HADR log shipping to the standby. A value of 0 indicates no timeout.

### **LOG\_HADR\_WAIT\_CUR**

The length of time, in seconds, that the logger has been waiting on an HADR log shipping request. A value of 0 is returned if the logger is not waiting. When the wait time reaches the value that is returned in the PEER\_WAIT\_LIMIT field, HADR breaks out of peer state to unblock the primary database.

### **LOG\_HADR\_WAIT\_RECENT\_AVG**

The average time, in seconds, for each log flush.

### **LOG\_HADR\_WAIT\_ACCUMULATED**

The accumulated time, in seconds, that the logger has spent waiting for HADR to ship logs.

### **LOG\_HADR\_WAITS\_COUNT**

The total count of HADR wait events in the logger. The count is incremented every time the logger initiates a wait on HADR log shipping, even if the wait returns immediately. As a result, this count is effectively the number of log flushes while the databases are in peer state.

### **SOCK\_SEND\_BUF\_REQUESTED,ACTUAL**

- The requested socket send buffer size (SOCK\_SEND\_BUF\_REQUESTED), in bytes. A value of 0 indicates no request (the system default is used).
- The actual socket send buffer size (SOCK\_SEND\_BUF\_ACTUAL), in bytes.

### **SOCK\_RECV\_BUF\_REQUESTED,ACTUAL**

- The requested socket receive buffer size (SOCK\_RECV\_BUF\_REQUESTED), in bytes. A value of 0 indicates no request (the system default is used).
- The actual socket receive buffer size (SOCK\_RECV\_BUF\_ACTUAL), in bytes.

### **PRIMARY\_LOG\_FILE, PAGE, POS**

## db2pd - Monitor and troubleshoot DB2 database

- The name of the current log file of the log stream on the primary database (PRIMARY\_LOG\_FILE).
- The page number in the current log file indicating the current log position on the primary HADR database. The page number is relative to its position in the log file. For example, page 0 is the beginning of the file (PRIMARY\_LOG\_PAGE).
- The current receive log position (byte offset) of the log stream on the primary database (PRIMARY\_LOG\_POS).

### STANDBY\_LOG\_FILE, PAGE, POS

- The name of the log file corresponding to the standby receive log position on the log stream (STANDBY\_LOG\_FILE).
- The page number (relative to its position in the log file) corresponding to the standby receive log position (STANDBY\_LOG\_PAGE).
- The current log position of the standby HADR database (STANDBY\_LOG\_POS).

### HADR\_LOG\_GAP

The running average, in bytes, of the gap between the PRIMARY\_LOG\_POS value and STANDBY\_LOG\_POS value.

### STANDBY\_REPLAY\_LOG\_FILE, PAGE, POS

- The name of the log file corresponding to the standby replay log position on the log stream (STANDBY\_REPLAY\_LOG\_FILE).
- The page number in the standby replay log file corresponding to the standby replay log position (STANDBY\_REPLAY\_LOG\_PAGE). The page number is relative to its position in the log file. For example, page 0 is the beginning of the file.
- The byte offset of the standby replay log position on the log stream (STANDBY\_REPLAY\_LOG\_POS).

### STANDBY\_RECV\_REPLAY\_GAP

The average, in bytes, of the gap between the standby log receive position and the standby log replay position. If the value of this gap reaches the combined value of the standby's receive buffer size and the standby's spool limit, the standby stops receiving logs and blocks the primary if it is in peer state.

### PRIMARY\_LOG\_TIME

The latest transaction timestamp of the log stream on the primary database.

### STANDBY\_LOG\_TIME

The latest transaction timestamp of received logs on the log stream on the standby database.

### STANDBY\_REPLAY\_LOG\_TIME

The transaction timestamp of logs being replayed on the standby database.

### STANDBY\_RECV\_BUF\_SIZE

The standby receive buffer size, in pages.

### STANDBY\_RECV\_BUF\_PERCENT

The percentage of standby log receive buffer that is currently being used. Even if this value is 100, indicating that the receive buffer is full, the standby can continue to receive logs if you enabled log spooling.

### **STANDBY\_SPOOL\_LIMIT**

The maximum number of pages to spool. A value of 0 indicates that log spooling is disabled; a value of -1 indicates that there is no limit.

### **PEER\_WINDOW**

The value of the **hadr\_peer\_window** database configuration parameter.

### **READS\_ON\_STANDBY\_ENABLED**

An indicator of whether the HADR reads on standby feature is enabled. Possible values are:

- Y
- N

### **STANDBY\_REPLAY\_ONLY\_WINDOW\_ACTIVE**

An indicator of whether the replay-only window (caused by DDL or maintenance-operation replay) is in progress on the standby, meaning that readers are not allowed on the standby. Possible values are:

- Y
- N

### **PEER\_WINDOW\_END**

The point in time until which the primary database stays in peer or disconnected peer state, as long as the primary database is active. The field is displayed only if you enabled a peer window.

### **STANDBY\_REPLAY\_DELAY**

Indicates the value of the **hadr\_replay\_delay** database configuration parameter.

### **TAKEOVER\_APP\_REMAINING\_PRIMARY**

The current number of applications still to be forced off the primary during a non-forced takeover. This field is displayed only if there is a non-forced takeover in progress.

### **TAKEOVER\_APP\_REMAINING\_STANDBY**

The current number of applications still to be forced off the read-enabled standby during a takeover. This field is displayed only if there is a takeover in progress.

### **STANDBY\_REPLAY\_ONLY\_WINDOW\_START**

The time at which the current replay-only window became active. This field is displayed only if there is an active replay-only window on the read-enabled standby.

### **STANDBY\_REPLAY\_ONLY\_WINDOW\_TRAN\_COUNT**

The total number of existing uncommitted DDL or maintenance transactions that have been executed so far in the current replay-only window. This field is displayed only if there is an active replay-only window on the read-enabled standby.

## **-latches parameter**

For the **-latches** parameter, the following information is returned:

### **Address**

Address of the holding latch in the virtual address space.

**Holder** The EDU ID of the EDU that is holding the latch.

**Waiter** The EDU ID of the EDU waiting for the latch.

## db2pd - Monitor and troubleshoot DB2 database

### Filename

The source file name where the latch is obtained.

### LOC

The line of code in the file indicated by the file name where the latch is obtained.

### LatchType

The identity of the latch being held.

## -load parameter

For the **-load** parameter, the following information is returned:

**LoadID** The ID of a specific load operation.

**EDU ID** The unique identifier for the engine dispatchable unit (EDU). Except on Linux operating systems, the EDU ID is mapped to the thread ID. On Linux operating system the EDU ID is a DB2 generated unique identifier.

### EDU Name

DB2 specific name for the EDU.

### TableName

The name of the table.

### SchemaName

The schema that qualifies the table name.

### AppHandle

The application handle, including the node and the index.

### Application ID

The application ID. This value is the same as the **appl\_id** monitor element data.

### StartTime

The date and time when the load operation was originally invoked.

### LoadPhase

The phase that the load operation is currently in.

## -locks parameter

For the **-locks** parameter, the following information is returned:

### TranHandle

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:

- IS
- IX
- S
- SIX
- X
- IN
- Z
- U
- NS
- 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.

**rrIID** The IID of the index through which the RR lock (0x10 attribute shown previously) was acquired. Possible values are:

- 0 Not related to a single, specific index (or not an RR lock).
- <>0 The specific index IID used to acquire the lock.

## db2pd - Monitor and troubleshoot DB2 database

### -logs parameter

For the **-logs** parameter, the following information is returned:

**Current Log Number**

The number of the current active log.

**Pages Written**

The current page being written in the current log.

**Cur Commit Disk Log Reads**

The number of times the currently committed version of a row was retrieved via a log read from disk (versus log buffer).

**Cur Commit Total Log Reads**

The total number of times the currently committed version of a row was retrieved from the logs (log buffer and disk).

**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.

**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.

**StartLSN**

The starting log sequence number.

**StartLSO**

The first LSO of the log file.

**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.

**Log Chain ID**

The identifier of the log chain number

**Current LSN**

The current log sequence number (LSN)

**Current LSO**

The current LSO.

See Sample output of the **db2pd -logs** command.

### -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.

### -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.

**SecondId**  
The second memory pool identifier to distinguish between multiple memory pools of the same type.

**Overhead**  
The internal usage information required for the pool structures.

**LogSz** The current total of pool memory requests.

**LogHWM** The logical size high water mark.

**PhySz** The physical memory required for logical size.

**PhyHWM** The largest physical size reached during processing.

**CfgSize**  
The configured size of the memory pool.

**Bnd** Specifies whether the memory pool has a fixed upper limit.

**BlkCnt** The current number of allocated blocks in the memory pool.

**CfgParm**  
The configuration parameter that declares the size of the pool being reported.

### -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 operating 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.

**HWM(Kb)**  
Maximum memory ever allocated to memory pools.

## db2pd - Monitor and troubleshoot DB2 database

### **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.

### **CmtRt (Kb)**

The largest contiguous area of committed memory that is available.

### **DcmtRt (Kb)**

The largest contiguous area of uncommitted memory that is available.

### **HoldRt (Kb)**

The largest contiguous area of committed memory that is available for volatile requests.

### **Sngl**

The number of pre-allocated regions that are available for faster allocation.

## **-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.

### **Cores/Socket**

Number of cores per socket

**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.

## db2pd - Monitor and troubleshoot DB2 database

### **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.

**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 semaphore identifiers.

**SemMns** System-wide maximum number of semaphores on system.

## db2pd - Monitor and troubleshoot DB2 database

- SemMnu** System-wide maximum number of undo structures on system.
- SemMs1** 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 preceding 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.

### -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.

## db2pd - Monitor and troubleshoot DB2 database

### **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 BlockMap.

**Dirty** Indicates if the page is dirty. Possible values are Y and N. In the summary information section of the pages output, the value indicates the number of dirty pages.

### **Permanent**

In the summary information section of the pages output, the value indicates the number of PERMANENT pages.

### **Temporary**

In the summary information section of the pages output, the value indicates the number of TEMPORARY pages.

### **Prefetched**

Indicates if the page has been prefetched. Possible values are Y and N.

See Sample output of the **db2pd -pages** command.

## **-recovery parameter**

For the **-recovery** parameter, the following information is returned:

### **Database State**

The state of the catalog partition in partitioned database environments if the database catalog partition fails. If the database catalog partition fails, the CATALOGNODEFAIL state is returned. Otherwise, no information is returned. This state can be displayed from any database partition.

### **Recovery Status**

The internal recovery status.

### **Current Log**

The current log being used by the recovery operation.

### **Current LSN**

The current log sequence number.

### **Current LRI**

The current LRI.

### **Current LSO**

The current LSO.

### **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.

**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**.

**Dynamic SQL Environments**

See the **-dynamic**.

**Dynamic SQL Variations**

See the **-dynamic**.

**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.

## db2pd - Monitor and troubleshoot DB2 database

**OrderNum**

Ordinal number of the variable that was used to reoptimize of the SQL statement

**SQLZType**

The variable type.

**CodPg** The variable code page.

**NullID** 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.

### -reorgs parameter

For the **-reorgs** parameter, the following information is returned:

**Index Reorg Stats:****Retrieval time**

Retrieval time of this set of index reorg statistics information.

**TabSpaceID**

The table space identifier.

**TableID**

The table identifier.

**Schema** Table schema.

**TableName**

The name of the table.

**MaxPartition**

Total number of partitions for the table being processed. For partition-level reorg, **MaxPartition** will always have a value of 1 since only a single partition is being reorganized. This field is only displayed for partitioned indexes.

**PartitionID**

The data partition identifier for the partition being processed. This field is only displayed for partitioned indexes.

**Access** Access level, possible values are:

- Allow none
- Allow read
- Allow write

**Status** Current reorg status, one of:

- In Progress (operation is in progress)
- Completed (operation has completed successfully)
- Stopped (operation has stopped due to error or interrupt)

**Start time**

Start time of this reorg session.

**End time**

End time of this reorg session.

**Total duration**

Total duration time of this reorg session.

**Prev Index Duration**

Reorg duration of the previous (completed) index.

**Cur Index Start**

Reorg start time of the current (in progress) index.

**Cur Index**

Sequence number of the current (in progress) index.

**Max Index**

Total number of indexes being monitored. This is not the same as total number of indexes on the table, because some system-generated indexes are not monitored.

**Index ID**

Index ID of the current (in progress) index.

**Cur Phase**

Sequence number of the current phase. Enclosed within the braces is the name of the current phase, one of:

- Scan (the table is being scanned and sorted one data page at a time)
- Build (the index is being built from the sorted input one row at a time)
- Catchup (transactions that occurred while building the index are being replayed; only seen for index reorgs where access level is allow write)

**Max Phase**

Total number of phases for the current (in-progress) index; differs for different types of indexes.

**CurCount**

Units of work processed so far. Unit has a different meaning for each reorg phase, as follows:

- Scan phase: number of data pages scanned
- Build phase: number of rows processed
- Catchup: number of transaction log records replayed

**MaxCount**

Total number of units for the current phase (see CurCount for explanation on units).

**Total Row Count**

Total number of rows processed. May or may not show up depending on the phase and index type.

See Sample output of the **db2pd -reorgs index** command.

**Table Reorg Stats:****Address**

A hexadecimal value.

**TableName**

The name of the table.

**Start** The time that the table reorganization started.

**End** The time that the table reorganization ended.

## db2pd - Monitor and troubleshoot DB2 database

### **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.

**PartID** The data partition identifier. One row is returned for each data partition, 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.

**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.

### -scansharing parameter

For the **-scansharing** parameter, the following fields are returned, specific to the headings:

#### Individual shared scan

- Agent ID
- Application ID
- ScanMode (prewrap or postwrap)
- IsScanWrappable
- Scan speed
- Time spent getting throttled
- Relative location of scan in pages within group (for block index scans).  
Absolute location of scan in pages (for table and range scans)
- Predicted speed category (SLOW or FAST)
- Remaining pages to process (accurate for table and range scans). For block index scans, the optimizer estimate is returned instead.

See Sample output of the **db2pd -scansharing** command.

#### Sharing set

- Table space ID
- Table ID
- Scan object (0 for table scans or ID of block index)
- Number of groups
- Sharing set footprint in pages
- Table size in pages (for table scans and block index scans on nonpartitioned tables, and for range scans on partitioned tables; for block index scans on partitioned tables the value is unknown)
- Fast scan speed (speed at which FAST scans are going)
- Slow scan speed (speed at which SLOW scans are going)

#### Sharing group

- Number of scans in the group
- Group footprint (in number of pages)

### -serverlist parameter

For the **-serverlist** parameter, the following information is returned:

**Time** The time when the server list was cached

#### Database Name

The name of the database

**Count** The number of entries in the server list

#### Hostname

The TCP/IP hostname of a member

#### Non-SSL Port

The non-SSL port that a member is listening on for client connections

## db2pd - Monitor and troubleshoot DB2 database

### SSL Port

The SSL TCP/IP port that a member is listening on for client connections

### Priority

The relative load of a member, also known as the weight. A member (A) having a higher value compared with another member (B) indicates to the client that more work should be directed at member A.

## **-serviceclasses parameter**

For the **-serviceclasses** parameter, the following fields are returned, specific to the headings:

### Service class fields:

- Service Class Name: Name of service class
- Service Class ID: System generated ID of service class
- Service Class Type: Type of service class: superclass or subclass
- Service Class State (Effective and Catalog): State of service class: enabled or disabled
- Effective Agent Priority and Catalog Agent Priority: Effective agent priority setting for service class that maps to priority recorded in SYSCAT.SERVICECLASSES
- Effective Prefetch Priority and Catalog Prefetch Priority: Effective prefetch priority setting for service class that maps to priority recorded in SYSCAT.SERVICECLASSES
- Effective Bufferpool Priority and Catalog Bufferpool Priority: Effective buffer pool priority setting for service class that maps to priority recorded in SYSCAT.SERVICECLASSES
- Effective Outbound Correlator and Catalog Outbound Correlator: Effective outbound correlator setting for service class that maps to correlator recorded in SYSCAT.SERVICECLASSES
- Last Statistics Reset Time: Timestamp of last statistics reset for service class

### Service superclass fields:

- Default Subclass ID: Service class ID of Default Subclass
- Work Action Set ID: ID of work action set associated with service superclass
- Collect Request Metrics: Setting of COLLECT REQUEST METRICS option for service class
- Num Connections: Current number of coordinator and remote connections in service superclass
- Num Coordinator Connections: Current number of coordinator connections in service superclass
- Coordinator Connections HWM: High water mark for coordinator connections since last statistics reset
- Associated Workload Occurrences (WLO): List of workload occurrences currently in service superclass

### Service subclass fields:

- Parent Superclass ID: Service class ID of parent superclass
- Collect Activity Opt: Setting of COLLECT ACTIVITY DATA option for service subclass

## db2pd - Monitor and troubleshoot DB2 database

- Collect Aggr Activity Opt: Setting of COLLECT AGGREGATE ACTIVITY option for service subclass
- Collect Aggr Request Opt: Setting of COLLECT AGGREGATE REQUEST option for service subclass
- Act Lifetime Histogram Template ID: ID of Activity Lifetime Histogram Template
- Act Queue Time Histogram Template ID: ID of Activity Queue Time Histogram Template
- Act Execute Time Histogram Template ID: ID of Activity Execute Time Histogram Template
- Act Estimated Cost Histogram Template ID: ID of Activity Estimated Cost Histogram Template
- Act Interarrival Time Histogram Template ID: ID of Activity Interarrival Time Histogram Template
- Request Execute Time Histogram Template ID: ID of Request Execute Time Histogram Template
- Access Count: Current number of activities in service subclass
- Activities HWM: High water mark for activities since last statistics reset, counting both activities that entered the system through this subclass and activities that you remap into this subclass by a REMAP ACTIVITY threshold action.
- Activities Completed: Total number of activities completed since last statistics reset. If you remap an activity to a different subclass with a REMAP ACTIVITY action before it completes, then this activity counts only toward the total of the subclass it completes in.
- Activities Rejected: Total number of activities rejected since last statistics reset
- Activities Aborted: Total number of activities aborted since last statistics reset. If you remap an activity to a different subclass with a REMAP ACTIVITY action before it aborts, then this activity counts only toward the total of the subclass it aborts in.
- Associated Agents: List of agent currently working in service subclass
- Associated Non-agent threads: List of non-agent entities currently working in service subclass

See Sample output of the **db2pd -serviceclasses** command.

### **-sort parameter**

For the **-sort** parameter, the following information is returned:

#### **App1Hand1**

The application handle, including the node and the index.

**SortCB** The address of a sort control block

#### **MaxRowSize**

The sum of the maximum length of all columns of the row being sorted

#### **EstNumRows**

The Optimizer estimated number of rows that will be inserted into the sort

#### **EstAvgRowSize**

The Optimizer estimated average length of the rows being sorted

## db2pd - Monitor and troubleshoot DB2 database

### **NumSMPSorts**

The number of concurrent subagents processing this sort

### **NumSpills**

The total number of times this sort has spilled to disk

### **KeySpec**

A description of the type and length of each column being sorted

### **SortheapMem**

The number of KB of sortheap memory reserved and allocated by this sort

### **NumSpilledRows**

The total number of rows spilled to disk for this sort

### **NumBufferedRows**

The total number of rows inserted into this sort since the last time it spilled

## **-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.

### -statisticscache parameter

For the **-statisticscache** parameter, the following information is returned:

**Current Size**  
The current number of bytes used in the statistics cache.

**Address**  
The address of the entry in the statistics cache.

**Schema** The schema qualifier for the table.

**Name** The name of the table.

**LastRefID**  
The last process identifier that referenced the table.

**LastStatsTime**  
The time for the latest statistics collection for the table.

**Sts** The status of the entry. The possible values are:

- V (valid).
- I (invalid).

For additional details about the returned information using the **-statisticscache** command parameter, see the topic “Catalog statistics tables” in *Troubleshooting and Tuning Database Performance*

### -storagegroups parameter and storagepaths parameter

Both the **-storagegroups** parameter and the **-storagepaths** parameter return the following information:

#### Storage Group Configuration:

**SGID** Storage group identifier.

## db2pd - Monitor and troubleshoot DB2 database

**Deflt** Indicates if the storage group is the current designated default storage group.

**DataTag**

An identifying tag used to uniquely identify and group data.

**Name** Name of the storage group.

**Storage Group Statistics:**

**SGID** Storage group identifier.

**State** State of the storage group. One of the following values:

- 0x0000000000000000 - SQLB\_STORAGEGROUP\_STATE\_NORMAL
- 0x0000000000000001 - SQLB\_STORAGEGROUP\_ALTER\_PENDING
- 0x0000000000000002 - SQLB\_STORAGEGROUP\_SKIP\_ALTERS
- 0x0000000000000004 - SQLB\_STORAGEGROUP\_KEEP\_ON\_DISK\_PATHS
- 0x0000000000000008 - SQLB\_STORAGEGROUP\_REDEFINE\_CONTAINERS
- 0x0000000000000010 - SQLB\_STORAGEGROUP\_CREATE\_PENDING
- 0x0000000000000020 - SQLB\_STORAGEGROUP\_DROP\_PENDING
- 0x0000000000000040 - SQLB\_STORAGEGROUP\_RENAME\_PENDING

**NumPaths**

Number of storage paths defined in this storage group.

**NumDropPen**

Number of storage paths in the Drop Pending state.

**Storage Group Paths:**

**SGID** Storage group identifier.

**PathID** Storage path identifier.

**PathState**

Current state of the storage path: NotInUse, InUse, or DropPending.

**PathName**

Name of an automatic storage path defined for the database. If the path contains a database partition expression, it is included, in parentheses, after the expanded path.

See Sample output of the **dp2pd -storagegroups** command and **db2pd -storagepaths** command.

### -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.

### -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

#### 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 TABLESPACE or ALTER TABLESPACE time. The possible values are:

- Yes
- No

#### NumCnts

The number of containers owned by a table space.

## db2pd - Monitor and troubleshoot DB2 database

### 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.

### Max HWM

The maximum HWM for the table space since the instance was started.

### State

- 0x00000000 - NORMAL
- 0x00000001 - QUIESCED: SHARE
- 0x00000002 - QUIESCED: UPDATE
- 0x00000004 - QUIESCED: EXCLUSIVE
- 0x00000008 - LOAD PENDING
- 0x00000010 - DELETE PENDING
- 0x00000020 - BACKUP PENDING
- 0x00000040 - ROLLFORWARD IN PROGRESS
- 0x00000080 - ROLLFORWARD PENDING
- 0x00000100 - RESTORE PENDING
- 0x00000200 - DISABLE PENDING

## db2pd - Monitor and troubleshoot DB2 database

- 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.

### PathsDropped

For automatic storage table spaces, specifies whether one or more containers reside on a storage path that has been dropped. The possible values are:

- Yes
- No

### TrackmodState

The modification status of a table space with respect to the last or next backup. The possible values are:

- Clean - No modifications have occurred in the table space since the previous backup. If an incremental or delta backup is executed at this time, no data pages from this table space would be backed up.
- Dirty - Table space contains data that needs to be picked up by the next backup.
- InIncremental - Table space contains modifications that were copied into an incremental backup. This state is in a Dirty state relative to a full backup such that a future incremental backup needs to include some pages from this pool. This state is also in a Clean state such that a future delta backup does not need to include any pages from this pool.
- ReadFull - The latest table space modification state change was caused by a dirty table space being read by a full backup that might not have completed successfully, or is currently in progress.
- ReadIncremental - The latest table space modification state change was caused by a dirty table space being read by an incremental backup that might not have completed successfully, or is currently in progress.
- n/a - The **trackmod** configuration parameter is set to No. Therefore, no table space modification status information is available.

### Table space Autoresize Statistics:

**Id** The table space ID.

## db2pd - Monitor and troubleshoot DB2 database

**AS** Indicates whether or not the table space is using automatic storage. The possible values are:

- Yes
- No

**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**

If the value of this parameter is -1, the database manager automatically determines an appropriate value. 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 Storage Statistics:**

**Id** The table space ID.

### **DataTag**

An identifying tag used to uniquely identify and group data.

### **Rebalance**

Indicates if a rebalance is active.

**SGID** For automatic storage managed table spaces, indicates the storage group the table space is associated with.

### **SourceSGID**

For automatic storage managed table spaces that are changing storage group association, indicates the source storage group the table space was associated with.

### **Table space Containers:**

## db2pd - Monitor and troubleshoot DB2 database

**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.

**PathID** For automatic storage table spaces, the identifier of the storage path on which the container resides.

See Sample output of the **db2pd -tablespaces** command.

### -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).

## db2pd - Monitor and troubleshoot DB2 database

- 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.

### **SchemaNm**

The schema that qualifies the table name.

**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 the background statistics collection process or manually using the **RUNSTATS** command.

**RTSUDI** 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 by real-time statistics gathering, background statistics collection process, or manual **RUNSTATS**.

### **PgReorgs**

The number of page reorganizations performed.

### **NoChgUpdts**

The number of updates that did not change any columns in the table.

**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.

### **CCLogReads**

The number of times the currently committed version of a row was retrieved for the table.



**StoredBytes**

This column corresponds to the “Total stored temp bytes” from the `db2pd -temptable` output.

**BytesSaved**

This column corresponds to the “Total bytes saved” value from the `db2pd -temptable` output.

**PgDictsCreated**

The total number of successfully created page-level dictionaries.

**Note** The following data is only displayed when the `-all` or `-index` option is specified with the `-tcbstats` parameter.

**TCB Index Information:****InxTspace**

The table space where the index resides.

**ObjectID**

The object identifier of the index.

**PartID** For partitioned tables, the data partition identifier. For nonpartitioned tables, N/A is displayed.

**TspaceID**

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 TspaceID.

**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. The value reported in IndexObjSize has been deprecated. If you perform a reorganization to reclaim extents, IndexObjSize output does not accurately reflect the number of pages in the index object because the value still includes the pages that were released during the reorganization. You should use instead the INDEX\_OBJECT\_P\_SIZE or INDEX\_OBJECT\_L\_SIZE columns of the ADMIN\_GET\_INDEX\_INFO table function to obtain accurate values.

**TCB Index Stats:****TableName**

The name of the table.

## db2pd - Monitor and troubleshoot DB2 database

**IID** The index identifier.

**PartID** For partitioned tables, the data partition identifier. For nonpartitioned tables, N/A is displayed.

**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.

**EmPgMkdUsd**  
The number of pseudo empty pages that have been reused.

**Scans** The number of scans against the index.

**IxOnlyScns**  
The number of index only scans (result of scan satisfied by access to index only).

**KeyUpdates**  
The number of updates to the key.

**Inc1Updats**  
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.

### -temptable parameter

In order to calculate the cumulative compression ratio across all of the temporary tables, the following formula can be used:

$$\% \text{ Compression} = \frac{\text{Total Bytes Saved}}{\text{Total Bytes Saved} + \text{Total Stored Temp Bytes}}$$

#### Note:

- The term Eligible indicates temporary tables that meet the compression criteria.
- The term Compressed indicates temporary tables that finally have sufficient data inserted to be compressed.

```
hotel26:/home/billyp> db2pd -db billdb -temptable
System Temp Table Stats:
  Number of Temp Tables   : 0
  Comp Eligible Temps    : 0
  Compressed Temps       : 0
  Total Temp Bytes       : 0
  Total Bytes Saved      : 0
  Total Compressed Rows  : 0
  Total Temp Table Rows  : 0
```

```
User Temp Table Stats:
  Number of Temp Tables   : 0
  Comp Eligible Temps    : 0
  Compressed Temps       : 0
  Total Stored Temp Bytes : 0
  Total Bytes Saved      : 0
  Total Compressed Rows  : 0
  Total Temp Table Rows  : 0
```

All of the counters can be reset to zero by using the hidden reset option.

```
hotel26:/home/billyp> db2pd -db bill -temptable reset
Resetting counters to 0.
```

See Sample output of the **db2pd -temptable** command.

### **-thresholds parameter**

For the **-thresholds** parameter, the following information is returned:

- Threshold Name: Threshold name
- Threshold ID: Threshold identifier
- Domain: Threshold domain
- Domain ID: Threshold domain identifier
- Predicate ID: Threshold predicate identifier
- Maximum Value: Threshold maximum value
- Statement Text: Statement text associated with a statement threshold
- Enforcement: Threshold enforcement scope
- Queuing: Threshold is a queuing threshold
- Queue Size: Threshold queue size setting
- Collect Flags: Setting of COLLECT ACTIVITY DATA option for threshold
- Partition Flags: Partitions where COLLECT ACTIVITY option setting applies
- Execute Flags: Threshold action setting
- Enabled: State of threshold, enabled or disabled
- Check Interval (seconds): Frequency setting for threshold
- Remap Target Serv. Subclass: Target service subclass setting for remapping threshold action
- Log Violation Evmon Record: THRESHOLD VIOLATIONS event monitor log setting

If the threshold is a queuing threshold, the queue section will also show:

- Queue information for threshold: Threshold Name
- Max Concurrency: Maximum concurrency setting
- Concurrency: Actual concurrency value
- Max Queue Size: Maximum threshold queue size setting

## db2pd - Monitor and troubleshoot DB2 database

- Agents Currently Queued: At the catalog database partition, the list of all agents waiting in the threshold queue (shown only when agents are queued)

### -transactions parameter

For the **-transactions** parameter, the following information is returned:

#### **AppHndl**

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.

**FirstLso**

First LSO of the transaction.

**LastLso**

Last LSO of the transaction.

**SpaceReserved**

The amount of log space that is reserved for the transaction.

**LogSpace**

The total log space that is required for the transaction, including the used space and the reserved space for compensation log 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.

**ClientUserID**

Client userid for the transaction, which is the same as tpmon\_client\_userid (TP Monitor Client User ID monitor element).

**ClientWrkstnName**

Client workstation name for the transaction, which is the same as tpmon\_client\_wkstn (TP Monitor Client Workstation Name monitor element).

**ClientAppName**

Client application name driving the transaction, which is the same as tpmon\_client\_app (TP Monitor Client Application monitor element).

**ClientAcctng**

Accounting string of the client driving the transaction, which is the same as tpmon\_acc\_str (TP Monitor Client Accounting String monitor element).

### -utilities parameter

For the **-utilities** parameter, the following information is returned:

**ID**

Unique identifier corresponding to the utility invocation.

**Type**

Identifies the class of the utility.

**State**

Describes the state of the utility.

**Invoker**

Describes how a utility was invoked.

**Priority**

Specifies the amount of relative importance of a throttled utility with respect to its throttled peers. A priority of 0 implies that a utility is executing unthrottled. Non-zero priorities must fall in the range of 1-100, with 100 representing the highest priority and 1 representing the lowest.

**StartTime**

Specifies the date and time when the current utility was originally invoked.

**DBName**

Identifies the database operated on by the utility.

## db2pd - Monitor and troubleshoot DB2 database

**NumPhases**

Specifies the number of phases a utility has.

**CurPhases**

Specifies the phase that is currently executing.

**Description**

A brief description of the work a utility is performing. This includes the load operation ID and the application ID.

### -wlocks parameter

For the **-wlocks** parameter, the following information is returned:

**AppHandle**

The application handle, including the node and the index.

**TranHandle**

The transaction handle that is requesting the lock.

**LockName**

The name of the lock.

**Type** The type of lock.

**Mode** The lock mode. The possible values are:

- IS
- IX
- S
- SIX
- X
- IN
- Z
- U
- NS
- NW

**Conv** The lock mode to which the lock will be converted after the lock wait ends.

**Sts** The lock status. The possible values are:

- G (granted)
- C (converting)
- W (waiting)

**CoordID**

The EDU ID of the coordinator agent for the application.

**AppName**

The name of the application.

**AuthID** The authorization identifier.

**AppID** The application ID. This value is the same as the **app\_id** monitor element data.

See Sample output of the **db2pd -wlocks** command.

### **-workactionsets parameter**

For the **-workactionsets** parameter, the following information is returned:

- Address
- Work action set ID
- Work action set name
- Associated work class set ID
- Type of object work action set is associated (database or service class)
- ID of the object (service class or database) work action set is associated with
- All the work actions within the work action set:
  - address
  - action ID
  - action type
  - reference object ID (threshold ID or service class ID or null depending on the action type)

### **-workclasssets parameter**

For the **-workclasssets** parameter, the following information is returned:

- address
- work class ID
- reference counter (number of different work action sets that reference this work class set)
- All the work classes within the work class set (shown in their evaluation order):
  - address
  - class ID
  - class name
  - attributes
  - work type
  - timeron cost from
  - timeron cost to

### **-workloads parameter**

For the **-workloads** parameter, the following information is returned, specific to the headings:

#### **Workload definitions**

- Workload ID and name
- Database access permission for workload occurrences
- Maximum degree of parallelism
- Number of concurrent workload occurrences
- Workload thresholds
- Associated service class
- Statistics collection settings
- Histogram template IDs

#### **Usage privilege holders**

- Workload ID

## db2pd - Monitor and troubleshoot DB2 database

- Type of holder
- Authorization ID

### Local partition workload statistics

- Workload ID and name
- Workload occurrence statistics
- Time since last statistics reset
- Activity statistics

See Sample output of the **db2pd -workloads** command.

## Sample output

### -addnode

The following example is a sample of the output of the **db2pd -addnode** command:

```
-----
Summary of add partition processing done for partition[50]
-----
00:Creating database partitions                : True
01:Database partitions are created            : True
08:Collecting storage information             : True
09:Storage information is collected           : True
11:FCM Send & Receive daemons are blocked    : True
12:FCM Send & Receive daemons are reactivated : True
13:db2start processing is complete           : True

Conflicting states or activities for add partition for partition[50]

-----
[14] Messages found for partition [50]
-----
[Fri Oct 24 16:16:27 2008]:Addnode agent:Got automatic storage details
[Fri Oct 24 16:16:28 2008]:Addnode agent:Skeleton database is created
[Fri Oct 24 16:16:28 2008]:Addnode agent:Scanning for db alias=[PE      ] name=[PE      ]
[Fri Oct 24 16:16:28 2008]:Addnode agent:Found db alias=[PE      ] name=[PE      ]
[Fri Oct 24 16:16:28 2008]:Addnode agent:Instance directory already exists
[Fri Oct 24 16:16:28 2008]:Addnode agent:Node directory already exists
[Fri Oct 24 16:16:28 2008]:Addnode agent:Node directory is created
[Fri Oct 24 16:16:29 2008]:Addnode agent:Getting automatic storage details
[Fri Oct 24 16:16:29 2008]:Addnode agent:Got automatic storage details
[Fri Oct 24 16:16:30 2008]:Addnode agent:Skeleton database is created
[Fri Oct 24 16:16:30 2008]:Addnode agent:Database activation is not required
[Fri Oct 24 16:16:30 2008]:Addnode agent:Database activation is complete
[Fri Oct 24 16:16:30 2008]:Addnode agent:Online mode processing is complete
[Fri Oct 24 16:16:30 2008]:db2start is complete
```

### oldviewapps

Returns information about which applications see the number of database partition servers in the instance before the add database partition server operation occurred.

The following example is a sample of the output of the **db2pd -addnode oldviewsapps** command:

```
-----
Summary of add partition processing done for partition[0]
-----

Conflicting states or activities for add partition for partition[0]
-----

Applications with old view of instance for partition [0]
-----
```



## db2pd - Monitor and troubleshoot DB2 database

```
App.Handle(00000000,00000072) view has [3] nodes, instance has [4] nodes
App.Handle(00000000,00000065) view has [3] nodes, instance has [4] nodes
App.Handle(00000000,00000071) view has [3] nodes, instance has [4] nodes
App.Handle(00000000,00000005) view has [3] nodes, instance has [4] nodes
App.Handle(00000000,00000051) view has [3] nodes, instance has [4] nodes
App.Handle(00000000,00000070) view has [3] nodes, instance has [4] nodes
App.Handle(00000000,00000069) view has [3] nodes, instance has [4] nodes
App.Handle(00000000,00000068) view has [3] nodes, instance has [4] nodes
App.Handle(00000001,00000058) view has [3] nodes, instance has [4] nodes
App.Handle(00000000,00000067) view has [3] nodes, instance has [4] nodes
App.Handle(00000000,00000073) view has [3] nodes, instance has [4] nodes
```

**detail** When used with the **db2pd** command, returns detailed information about the add database partition server operation, including the step in progress and events that are incompatible with the add database partition server operation. When used with the **oldviewapps** option, also returns information about which applications have a view of the instance that does not include recently added database partition servers.

The following example is a sample of the output of the **db2pd -addnode detail** command:

```
-----
Add partition processing with detail for partition[50]
-----
00:Creating database partitions                : True
01:Database partitions are created             : True
02:Dropping database entries                   : False
03:Dropping db entries are completed           : False
04:Activating databases explicitly             : False
05:Database explicit activation is completed   : False
06:Updating database configuration             : False
07:Database configuration is updated           : False
08:Collecting storage information              : True
09:Storage information is collected            : True
10:Add partition operation is complete         : False
11:FCM Send & Receive daemons are blocked    : True
12:FCM Send & Receive daemons are reactivated : True
13:db2start processing is complete            : True

Conflicting states or activities for add partition for partition[50]
-----
restricted                :False
db2start                   :False
db2stop                    :False
instance quiesced         :False
database quiesced         :False
quiesce instance          :False
unquiesce instance        :False
quiesce db                 :False
unquiesce db              :False
activate db                :False
deactivate db             :False
exclusive use of db       :False
create db                  :False
drop db                    :False
create tablespace         :False
alter tablespace          :False
drop tablespace           :False
add partition              :False
backup database           :False
restore database          :False
snapshot restore          :False

[14] Messages found for partition [50]
-----
```

## db2pd - Monitor and troubleshoot DB2 database

```
[Fri Oct 24 16:16:27 2008]:Addnode agent:Got automatic storage details
[Fri Oct 24 16:16:28 2008]:Addnode agent:Skeleton database is created
[Fri Oct 24 16:16:28 2008]:Addnode agent:Scanning for db alias=[PE      ] name=[PE      ]
[Fri Oct 24 16:16:28 2008]:Addnode agent:Found db alias=[PE      ] name=[PE      ]
[Fri Oct 24 16:16:28 2008]:Addnode agent:Instance directory already exists
[Fri Oct 24 16:16:28 2008]:Addnode agent:Node directory already exists
[Fri Oct 24 16:16:28 2008]:Addnode agent:Node directory is created
[Fri Oct 24 16:16:29 2008]:Addnode agent:Getting automatic storage details
[Fri Oct 24 16:16:29 2008]:Addnode agent:Got automatic storage details
[Fri Oct 24 16:16:30 2008]:Addnode agent:Skeleton database is created
[Fri Oct 24 16:16:30 2008]:Addnode agent:Database activation is not required
[Fri Oct 24 16:16:30 2008]:Addnode agent:Database activation is complete
[Fri Oct 24 16:16:30 2008]:Addnode agent:Online mode processing is complete
[Fri Oct 24 16:16:30 2008]:db2start is complete
```

Total [00] Conflicting application handles for partition [50]

-----  
Conflicting operations are shown as in the following example:

Total [01] Conflicting application handles for partition [20]

-----  
Agents for app\_handle 00000000 00000052 : Activity occurrence:[1] time(s) ActivityName:[exclusive use of db]

The following example is a sample of the output of the db2pd  
-addnode oldviewapps detail command:

-----  
Add partition processing with detail for partition[0]

```
-----  
00:Creating database partitions                : False  
01:Database partitions are created             : False  
02:Dropping database entries                  : False  
03:Dropping db entries are completed          : False  
04:Activating databases explicitly            : False  
05:Database explicit activation is completed  : False  
06:Updating database configuration            : False  
07:Database configuration is updated          : False  
08:Collecting storage information              : False  
09:Storage information is collected           : False  
10:Add partition operation is complete         : False  
11:FCM Send & Receive daemons are blocked    : False  
12:FCM Send & Receive daemons are reactivated : False  
13:db2start processing is complete            : False
```

Conflicting states or activities for add partition for partition[0]

```
-----  
restricted                :False  
db2start                   :False  
db2stop                    :False  
instance quiesced          :False  
database quiesced          :False  
quiesce instance           :False  
unquiesce instance         :False  
quiesce db                  :False  
unquiesce db                :False  
activate db                 :False  
deactivate db               :False  
exclusive use of db        :False  
create db                   :False  
drop db                     :False  
create tablespace          :False  
alter tablespace           :False  
drop tablespace            :False  
add partition               :False  
backup database            :False  
restore database           :False  
snapshot restore           :False
```

## db2pd - Monitor and troubleshoot DB2 database

```
create/alter nodegroup      :False
drop nodegroup              :False
add storage                  :False
redistribute                 :False
```

Total [00] Conflicting application handles for partition [0]

-----  
Applications with old view of instance for partition [0]  
-----

```
App.Handle(00000000,00000072) view has [3] nodes, instance has [4] nodes
[Viewnodes:0:1:2:]
App.Handle(00000000,00000065) view has [3] nodes, instance has [4] nodes
[Viewnodes:0:1:2:]
App.Handle(00000000,00000071) view has [3] nodes, instance has [4] nodes
[Viewnodes:0:1:2:]
App.Handle(00000000,00000005) view has [3] nodes, instance has [4] nodes
[Viewnodes:0:1:2:]
App.Handle(00000000,00000051) view has [3] nodes, instance has [4] nodes
[Viewnodes:0:1:2:]
App.Handle(00000000,00000070) view has [3] nodes, instance has [4] nodes
[Viewnodes:0:1:2:]
App.Handle(00000000,00000069) view has [3] nodes, instance has [4] nodes
[Viewnodes:0:1:2:]
App.Handle(00000000,00000068) view has [3] nodes, instance has [4] nodes
[Viewnodes:0:1:2:]
App.Handle(00000001,00000058) view has [3] nodes, instance has [4] nodes
[Viewnodes:0:1:2:]
App.Handle(00000000,00000067) view has [3] nodes, instance has [4] nodes
[Viewnodes:0:1:2:]
App.Handle(00000000,00000073) view has [3] nodes, instance has [4] nodes
[Viewnodes:0:1:2:]
```

### -apinfo

The following example is a sample of the output of the **db2pd -apinfo** command:

```
db2pd -apinfo 12 -db mydb10
```

```
Database Partition 0 -- Database MYDB10 -- Active -- Up 0 days 00:03:28
```

```
Application :
Address : 0x0780000000D76EE0
AppHandl [nod-index] : 12 [000-00012]
Application PID : 1384708
Application Node Name : boson
IP Address: n/a
Connection Start Time : (1195265036)Fri Nov 16 21:03:56 2007
Client User ID : venus
System Auth ID : VENUS
Coordinator EDU ID : 1801
Coordinator Partition : 0
Number of Agents : 1
Locks timeout value : 4294967294 seconds
Locks Escalation : No
Workload ID : 1
Workload Occurrence ID : 1
Trusted Context : n/a
Connection Trust Type : non trusted
Role Inherited : n/a
Application Status : Lock-wait
Application Name : db2bp
Application ID : *LOCAL.venus.071117020356
ClientUserID : n/a
ClientWrkstnName : n/a
ClientApplName : n/a
ClientAcctng : n/a
```

## db2pd - Monitor and troubleshoot DB2 database

List of active statements :

```
*UOW-ID :      8
Activity ID :   2
Package Schema : NULLID
Package Name :  SQLC2G13
Package Version :
Section Number : 201
SQL Type :     Dynamic
Isolation :    CS
Statement Type : DML, Select (blockable)
Statement :    select * from t2
```

List of inactive statements of current UOW :

```
UOW-ID :      8
Activity ID :   1
Package Schema : NULLID
Package Name :  SQLC2G13
Package Version :
Section Number : 203
SQL Type :     Dynamic
Isolation :    CS
Statement Type : DML, Insert/Update/Delete
Statement :    insert into t1 values 1
```

### -catalogcache

The following example is a sample of the SYSTABLES and TABLESPACES output of the **db2pd -catalogcache** command:

```
Catalog Cache:
Configured Size      819200
Current Size         537464
Maximum Size        4294901760
High Water Mark     589824
```

```
SYSTABLES:
Address          Schema  Name                Type TableID  TspaceID  LastRefID  CatalogCacheLoadingLock  CatalogCacheUsageLock  Sts
0x00002B3AD9CB5C40 SYSCAT  TABLESPACES       V   0         0         165         010000003A2B0000405CCBD9C3 000005000C110000405CCBD9C3 V
0x00002B3AD9E97680 DBUSER1 SALES              T  16         2         164         010000003A2B00008076E9D9C3 000005000E1B00008076E9D9C3 S
0x00002B3AD9E5F920 DBUSER1 VSTAFAC2           V   0         0         164         010000003A2B000020F9E5D9C3 00000500001D000020F9E5D9C3 V
0x00002B3AD9EDD600 DBUSER1 PRODUCT           T   4         4         164         010000003A2B000060DDBED9C3 00000500061D000060DDBED9C3 S
0x00002B3AD9E62920 SYSIBM  SYSPERIODS        T  164        0         164         010000003A2B00002029E6D9C3 00000500050800002129E6D9C3 V
0x00002B3AD9E6E1A0 DBUSER1 EMP_PHOTO          T   7         2         164         010000003A2B0000A0E1E6D9C3 00000500021B0000A0E1E6D9C3 V
0x00002B3AD9E5A500 SYSPUBLI HMON_COLLECTION   0   0         0         164         010000003A2B00000A5E5D9C3 000000000000000000000000 I
0x00002B3AD9E11C60 SYSIBM  SYSTABLES         T   5         0         164         010000003A2B0000601CE1D9C3 000005000400000061CE1D9C3 V

0x00002B3AD9E60060 DBUSER1 EMP_RESUME         T   8         2         164         010000003A2B000060D0E6D9C3 00000500031B000060D0E6D9C3 V
0x00002B3AD9CB56A0 SYSTOOLS POLICY          T   4         5         164         010000003A2B0000A056CBD9C3 000005000D1D0000A056CBD9C3 V
0x00002B3AD9E66C60 DBUSER1 EMPLOYEE         T   6         2         164         010000003A2B0000606CE6D9C3 00000500001B0000606CE6D9C3 S
0x00002B3AD9CB6E00 SYSCAT  TABLES           V   0         0         164         010000003A2B000000E6CBD9C3 000005000B11000000E6CBD9C3 V
0x00002B3AD9E642E0 DBUSER1 EMPPROJACT        T  11         2         164         010000003A2B0000E042E6D9C3 00000500071B0000E042E6D9C3 S
0x00002B3AD9DA26A0 DBUSER1 CUSTOMER          T   6         4         164         010000003A2B0000A026DAD9C3 00000500081D0000A026DAD9C3 S
0x00002B3AD9E996E0 DBUSER1 ACT                T  12         2         164         010000003A2B0000E096E9D9C3 000005000A1B0000E196E9D9C3 V
```

```
TABLESPACES:
Address          Name                TspaceID  LastRefID  CatalogCacheLoadingLock  CatalogCacheUsageLock  Sts
0x00002B3AD9BED6C0 SYSCATSPACE        0         164         110000003A2B0000C0D6BED9C3 0000210004000000C0D6BED9C3 V
0x00002B3AD9BE3080 TEMPSPACE1         1         31         110000003A2B00008030BED9C3 00002100050000008030BED9C3 V
0x00002B3AD9BF2F00 USERSPACE1         2         164         110000003A2B0000002FBFD9C3 000021000600000002FBFD9C3 V
0x00002B3AD9E62EC0 IBMDB2SAMPLEXML    4         164         110000003A2B0000C02EE6D9C3 0000210008000000C02EE6D9C3 V
0x00002B3AD9BF2E00 SYSTOOLSPACE       5         164         110000003A2B0000002EBFD9C3 000021000900000002EBFD9C3 V
```

### -cleaner

The following is sample output of the **-cleaner** option:

```
db2pd -db sample -cleaner
```

```
Database Partition 0 - Database SAMPLE - Active - Up 0 days 00:06:34 -
Date 08/09/2010 14:17:58
```

Page Cleaners:

Group ID	Clnr	Idx	State	Cycle	Task	Pgs	Gthr	Pgs	Ga'd	IO	outstnd	Max	AIO	Pgs	Thrsh	Pgs	D	Stl	Pgs	Skpdd
-1	0		Wait	0	None	0		0		0	32			0				0		
-1	1		Wait	0	None	0		0		0	32			0				0		
-1	2		Wait	0	None	0		0		0	32			0				0		
-1	3		Wait	0	None	0		0		0	32			0				0		
-1	4		Wait	0	None	0		0		0	32			0				0		
...																				

## db2pd - Monitor and troubleshoot DB2 database

```

Dirty lists for Bufferpool ID : 1
List ID # Dirty Clnr Idx Trigger Target LSN Pgs for Gap
0 4 0 None 0000000000000000 0
0 0 0 None 0000000000000000 0
1 8 1 None 0000000000000000 0
1 0 1 None 0000000000000000 0
2 2 2 None 0000000000000000 0
2 0 2 None 0000000000000000 0
3 1 3 None 0000000000000000 0
    
```

### -dirtypages

The following is sample output of the **-dirtypages** option:

```
db2pd -db sample -dirtypages
```

```

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:00:20 --
Date 08/09/2010 14:11:44
    
```

```

Bufferpool: 1
  Dirty pages %      : 34 / 1000 (3.40% dirty)
  Bufferpool minbuflsn: 00000000138800C
    
```

```

Oldest page info:
DirtyLst TspID PPNum   ObjID OPNum   Typ UFlag fixcount   wgt CPC LSN           pgLtch
n/a      0      327      15    3          4    3    0           2    0  00000000138800C 0x07000000323508E8

  Dirty pages:
DirtyLst TspID PPNum   ObjID OPNum   Typ UFlag fixcount   wgt CPC LSN           pgLtch
0        0      272      14    0          0    3    0           2    0  00000000138881C 0x070000003236C2E8 hX:0 sH:0 xW:0 rC:0
0        0      273      14    1          0    3    0           1    0  00000000138881C 0x070000003236B228 hX:0 sH:0 xW:0 rC:0
0        0      7541     18    9          1    3    0           2    0  00000000138E237 0x07000000323678A8 hX:0 sH:0 xW:0 rC:0
0        0      7540     18    8          1    3    0           2    0  00000000138E402 0x0700000032367A28 hX:0 sH:0 xW:0 rC:0
1        0      6945     15    5          1    3    0           2    0  000000001388107 0x070000003236F3E8 hX:0 sH:0 xW:0 rC:0
1        0      300      14    4          1    3    0           2    0  00000000138889D 0x070000003236B6A8 hX:0 sH:0 xW:0 rC:0
...
    
```

```

Recovery information:
  lowtranlsn      : 00000000138E486
  minbuflsn      : 00000000138800C
  nextlsn        : 00000000138E4B0
  LFH lowtranlsn : 00000000138E486
  LFH minbuflsn  : 00000000138800C
  LFH nextlsn    : 00000000138800C
  Active Log bytes in use : 25764
  Current Softmax  : 4096000
    
```

```

DirtyLst - dirty list ID in this bufferpool
TspID - tablespace ID of this page
PPNum - pool page number
ObjID - object ID
OPNum - object page number
Typ - type of the object
UFlag - internal page flag
fixcount - number of active fixes on this page (in-use count)
wgt - weight of the page
CPC - clock
LSN - page LSN
pgLtch - page latch address, hX - held X, sH - # shard holders, xW - # of
X waiters
    
```

### -edus The following example is a sample of the output of the **db2pd -edus** command:

```
Database Partition 0 -- Active -- Up 0 days 01:14:05
```

```
List of all EDUs for database partition 0
```

```

db2sysc PID: 18485
db2wdog PID: 18483
db2acd  PID: 18504
    
```

EDU ID	TID	Kernel TID	EDU Name	USR	SYS
24	47155322546496	12108	db2pfchr (TESTDB)	0.010000	0.000000
23	47155326740800	12107	db2pclnr (TESTDB)	0.000000	0.000000
22	47155330935104	12106	db2pclnr (TESTDB)	0.000000	0.000000
21	47155335129408	12105	db2pclnr (TESTDB)	0.000000	0.000000
20	47155339323712	12104	db2dlock (TESTDB)	0.000000	0.000000
19	47155343518016	12103	db2lfr (TESTDB)	0.000000	0.000000
18	47155347712320	12102	db2loggw (TESTDB)	0.000000	0.000000
17	47155351906624	12101	db2loggr (TESTDB)	0.080000	0.000000
16	47155356100928	27704	db2agent (TESTDB) (suspended)	0.930000	0.140000
15	47155360295232	18502	db2resync	0.080000	0.000000

## db2pd - Monitor and troubleshoot DB2 database

14	47155364489536	18500	db2ipccm	0.030000	0.000000
13	47155368683840	18499	db2licc	0.000000	0.000000
12	47155372878144	18498	db2thc1n	0.000000	0.000000
11	47155377072448	18497	db2alarm	0.000000	0.000000
1	47155117025600	18493	db2sysc	3.340000	0.070000

If you include an interval, such as **db2pd -edus interval=10** then an additional two columns would be added to the right side of the output after the SYS column:

```
... USR DELTA      SYS DELTA
... =====
... 0.141799      0.045431
... 0.101154      0.045117
... 0.038113      0.020154
... 0.005668      0.007978
... 0.005139      0.004392
... 0.005003      0.004105
... 0.003913      0.004100
... 0.001785      0.001282
... 0.001083      0.001550
... 0.001005      0.000433
... 0.000181      0.000098
... 0.000095      0.000091
... 0.000000      0.000000
... 0.000000      0.000000
... 0.000000      0.000000
```

### **-fmpexechistory | -fmpe**

The following example is a sample of the output of the **db2pd -fmpexechistory** command:

```
db2pd -fmpexechistory -pid 761872 -n 10
```

```
Database Partition 0 -- Active -- Up 0 days 00:00:11
```

```
FMP Process:
```

FmpPid	Bit	Flags	ActiveThrd	PooledThrd	ForcedThrd	Active
761872	64	0x00000002	2	1	1	YES

```
Active Threads:
```

```
EduPid: 123456 ThreadId: 987654
```

```
RoutineID  Timestamp
```

1	2009-05-06-17.12.30.000000
2	2009-05-06-17.12.30.005000
1	2009-05-06-17.12.30.100000

```
EduPid: 234567 ThreadId: 987000
```

```
RoutineID  Timestamp
```

1	2009-05-06-17.12.31.000000
3	2009-05-06-17.12.30.000000

```
Pooled Threads:
```

```
ThreadId: 540021
```

```
RoutineID  Timestamp
```

4	2009-05-06-17.10.30.000000
---	----------------------------

```
Forced Threads:
```

```
ThreadId: 120021
```

```
RoutineID  Timestamp
```

10	2009-05-06-15.10.30.000000
----	----------------------------

The following example is a sample of the output of the **db2pd -fmpexechistory** command with genquery option:

## db2pd - Monitor and troubleshoot DB2 database

```
db2pd -fmpExecHistory pid=761872 n=10 genquery
```

```
Database Partition 0 -- Active -- Up 0 days 00:00:11
```

```
WITH RTNHIST ( PID, TID, RTNID, RTNTIME) AS
  ( VALUES (761872, 987654, 1, TIMESTAMP('2009-07-13-16.17.10.818705')),
    (761872, 987654, 2, TIMESTAMP('2009-07-13-16.17.11.818710')),... )
SELECT R.PID, R.TID, R.RTNTIME, ROUTINESCHEMA, ROUTINEMODULENAME, ROUTINENAME, SPECIFICNAME, ROUTINEID
FROM syscat.routines, RTNHIST as R
WHERE ROUTINEID = R.RTNID
ORDER BY R.PID, R.TID, R.RTNTIME ;
```

### -logs

The following example is a sample of the output of the **db2pd -logs** command:

### -pages

The following example is a sample of the output of the **db2pd -pages** command without specifying the summary parameter:

```
venus@baryon:/home/venus =>db2pd -pages -db pdtest
```

```
Database Partition 0 -- Database PDTEST -- Active -- Up 0 days 00:01:28
```

```
Bufferpool Pages:
First Active Pool ID      1
Max Bufferpool ID         1
Max Bufferpool ID on Disk 1
Num Bufferpools           5
```

```
Pages for all bufferpools:
```

Address	BPID	TbspaceID	TbspacePgNum	ObjID	ObjPgNum	ObjClass	ObjType	Dirty	Prefetched
0x0000002AC22ABAC0	1	0	92	10	0	EMP	Data	N	N
0x0000002AC22ABB80	1	0	2503	10	11	Perm	Index	N	N
0x0000002AC22ABC40	1	0	2501	10	9	Perm	Index	Y	N
0x0000002AC22ABD00	1	0	2494	10	2	Perm	Index	N	N
0x0000002AC22ABDC0	1	0	3437	5	17	Perm	Data	N	N
0x0000002AC22ABE80	1	0	2504	10	12	Perm	Index	Y	N
0x0000002AC22ABF40	1	0	2505	10	13	Perm	Index	N	N
0x0000002AC22AC000	1	0	2506	10	14	Perm	Index	N	N
0x0000002AC22AC0C0	1	0	28	5	0	EMP	LOB	N	N
0x0000002AC22AC180	1	0	2509	10	17	Perm	Index	N	N
0x0000002AC22AC240	1	0	2495	10	3	Perm	Index	Y	N
0x0000002AC22AC300	1	0	2498	10	6	Perm	Index	Y	N
0x0000002AC22AC3C0	1	2	128	4	0	Perm	Data	Y	N
0x0000002AC22AC480	1	0	2499	10	7	Perm	Index	N	N
0x0000002AC22AC540	1	0	99	10	3	Perm	Data	Y	N
0x0000002AC22AC600	1	0	96	10	0	Perm	Data	Y	N
0x0000002AC22AC6C0	1	0	110	5	2	Perm	Index	N	N
0x0000002AC22AC780	1	0	2500	10	8	Perm	Index	N	N
0x0000002AC22AC840	1	0	2740	5	16	Perm	Index	N	N
0x0000002AC22AC900	1	0	2507	10	15	Perm	Index	Y	N

Total number of pages: 20

```
Summary info for all bufferpools:
```

BPID	TbspaceID	ObjID	Total	Dirty	Permanent	Temporary	Data	Index	LongField	XMLData	SMP	LOB	LOBA	BMP
1	0	5	4	0	3	0	1	2	0	0	0	1	0	0
1	0	10	15	7	14	0	3	12	0	0	0	0	0	0
1	2	4	1	1	1	0	1	0	0	0	0	0	0	0

Total number of pages: 20

The following example is a sample of the output of the **db2pd -pages** command specifying the **summary** parameter:

```
venus@baryon:/home/venus =>db2pd -pages summary -db pdtest
```

```
Database Partition 0 -- Database PDTEST -- Active -- Up 0 days 00:02:07
```

```
Bufferpool Pages:
First Active Pool ID      1
Max Bufferpool ID         1
Max Bufferpool ID on Disk 1
Num Bufferpools           5
```

```
Total number of pages: 20
```

## db2pd - Monitor and troubleshoot DB2 database

Summary info for all bufferpools:

BPID	TbSpaceID	ObjID	Total	Dirty	Permanent	Temporary	Data	Index	LongField	XMLData	SMP	LOB	LOBA	BMP
1	0	5	4	0	3	0	1	2	0	0	0	1	0	0
1	0	10	15	7	14	0	3	12	0	0	0	0	0	0
1	2	4	1	1	1	0	1	0	0	0	0	0	0	0

Total number of pages: 20

### -reorgs index

The following section is an example of output obtained using the **-reorgs index** parameter which reports the index reorg progress for a range-partitioned table with 2 partitions.

**Note:** The first output reports the Index Reorg Stats of the non-partitioned indexes. The following outputs report the Index Reorg Stats of the partitioned indexes on each partition; the index reorg statistics of only one partition is reported in each output.

Index Reorg Stats:

Retrieval Time: 02/08/2010 23:04:21

TbSpaceID: -6 TableID: -32768

Schema: ZORAN TableName: BIGRPT

Access: Allow none

Status: Completed

Start Time: 02/08/2010 23:03:55 End Time: 02/08/2010 23:04:04

Total Duration: 00:00:08

Prev Index Duration: -

Cur Index Start: -

Cur Index: 0 Max Index: 2 Index ID: 0

Cur Phase: 0 ( - ) Max Phase: 0

Cur Count: 0 Max Count: 0

Total Row Count: 750000

Retrieval Time: 02/08/2010 23:04:21

TbSpaceID: 2 TableID: 5

Schema: ZORAN TableName: BIGRPT

PartitionID: 0 MaxPartition: 2

Access: Allow none

Status: Completed

Start Time: 02/08/2010 23:04:04 End Time: 02/08/2010 23:04:08

Total Duration: 00:00:04

Prev Index Duration: -

Cur Index Start: -

Cur Index: 0 Max Index: 2 Index ID: 0

Cur Phase: 0 ( - ) Max Phase: 0

Cur Count: 0 Max Count: 0

Total Row Count: 375000

Retrieval Time: 02/08/2010 23:04:21

TbSpaceID: 2 TableID: 6

Schema: ZORAN TableName: BIGRPT

PartitionID: 1 MaxPartition: 2

Access: Allow none

Status: Completed

Start Time: 02/08/2010 23:04:08 End Time: 02/08/2010 23:04:12

Total Duration: 00:00:04

Prev Index Duration: -

Cur Index Start: -

Cur Index: 0 Max Index: 2 Index ID: 0

Cur Phase: 0 ( - ) Max Phase: 0

Cur Count: 0 Max Count: 0

Total Row Count: 375000

### -scansharing

The following section is an example of output using the **-scansharing** parameter. The output shows two sharing sets. The table scan set has two groups and the block index scan set has one group.



## db2pd - Monitor and troubleshoot DB2 database

Database Partition 0 -- Database TPCD -- Active -- Up 0 days 00:00:45

### Scan Sets:

TbSpaceID	TableID	ScanObject	NumGroups	Footprint	TableSize	FastScanRate	SlowScanRate
2	3	0	2	11520	22752	2486	1000

#### Group Information:

FootPrint	NumScannersInGroup
8288	3

#### Scans In Group :

AgentID	ApplID	Mode	Wrappable	Fast/Slow	Speed	ThrottleTime	Absolute Location	Remaining Pages
9768	1173	0	0	1	2486	0	32	22751
11332	1165	0	0	1	2486	0	5056	17727
15466	1155	0	0	1	2486	0	8288	14495

#### Group Information:

FootPrint	NumScannersInGroup
3232	2

#### Scans In Group :

AgentID	ApplID	Mode	Wrappable	Fast/Slow	Speed	ThrottleTime	Absolute Location	Remaining Pages
15209	1150	0	0	1	2486	0	14080	8703
12103	1148	0	0	1	2486	0	17280	5503

### Scan Sets:

TbSpaceID	TableID	ScanObject	NumGroups	Footprint	TableSize	FastScanRate	SlowScanRate
2	3	1	1	9056	22752	1000	1000

#### Group Information:

FootPrint	NumScannersInGroup
9056	3

#### Scans In Group :

AgentID	ApplID	Mode	Wrappable	Fast/Slow	Speed	ThrottleTime	Relative Location	Estimated Remaining Pages
6170	1209	0	0	1	1000	0	896	13535
13645	1215	0	0	1	1000	0	3552	10879
4371	1204	0	0	1	1000	0	9920	4511

## -serverlist

The following are samples of the serverlist output

Sample serverlist output from db2pd -serverlist -db sample

Database Member 0 -- Active -- Up 0 days 00:10:43 -- Date 10/06/2010 12:22:39

### Server List:

Time: Wed Oct 6 12:13:17  
Database Name: SAMPLE  
Count: 2

Hostname	Non-SSL Port	SSL Port	Priority
coralxib23.torolab.ibm.com	49712	0	34
coralxib24.torolab.ibm.com	49712	0	65

Sample service subclass output from db2pd -serverlist -alldbs

Database Member 0 -- Active -- Up 0 days 00:06:15 -- Date 10/06/2010 12:18:11

### Server List:

Time: Wed Oct 6 12:13:17  
Database Name: SAMPLE  
Count: 2

Hostname	Non-SSL Port	SSL Port	Priority
coralxib23.torolab.ibm.com	49712	0	34
coralxib24.torolab.ibm.com	49712	0	65

Database Member 0 -- Active -- Up 0 days 00:06:15 -- Date 10/06/2010 12:18:11

### Server List:

Time: Wed Oct 6 12:17:00  
Database Name: SAMPLE2  
Count: 2

Hostname	Non-SSL Port	SSL Port	Priority
coralxib23.torolab.ibm.com	49712	0	56
coralxib24.torolab.ibm.com	49712	0	43

## -serviceclasses

The following example is a sample of the service classes information output for one service superclass and its subclass.

## db2pd - Monitor and troubleshoot DB2 database

### Sample service superclass output:

```
Service Class Name      = SYSDEFAULTSYSTEMCLASS
Service Class ID       = 1
Service Class Type     = Service Superclass
Default Subclass ID   = 11
Effective Service Class State = Enabled
Catalog Service Class State = Enabled
Effective Agent Priority = 0
Catalog Agent Priority = Default
Effective Prefetch Priority = Medium
Catalog Prefetch Priority = Default
Effective Bufferpool Priority = Low
Catalog Bufferpool Priority = Default
Effective Outbound Correlator = None
Catalog Outbound Correlator = None
CPU Shares             = 1000
CPU Share Type         = Soft
CPU Limit              = None
Work Action Set ID     = N/A
Collect Activity Opt   = None
Collect Request Metrics = Base

Num Connections        = 5
Last Statistics Reset Time = 12/16/2008 15:27:42.000000
Num Coordinator Connections = 5
Coordinator Connections HWM = 5
```

### Associated Workload Occurrences (WLO):

AppHandl	[nod-index]	WL ID	WLO ID	UOW ID	WLO State
10	[000-00010]	0	0	1	UOWWAIT
11	[000-00011]	0	0	1	UOWWAIT
12	[000-00012]	0	0	1	UOWWAIT
13	[000-00013]	0	0	1	UOWWAIT
14	[000-00014]	0	0	1	UOWWAIT

### Sample service subclass output:

```
Service Class Name      = SYSDEFAULTSUBCLASS
Service Class ID       = 11
Service Class Type     = Service Subclass
Parent Superclass ID   = 1
Effective Service Class State = Enabled
Catalog Service Class State = Enabled
Effective Agent Priority = 0
Catalog Agent Priority = Default
Effective Prefetch Priority = Medium
Catalog Prefetch Priority = Default
Effective Bufferpool Priority = Low
Catalog Bufferpool Priority = Default
Effective Outbound Correlator = None
Catalog Outbound Correlator = None
Collect Activity Opt   = None
Collect Request Metrics = None
Collect Aggr Activity Opt = None
Collect Aggr Request Opt = None
Act Lifetime Histogram Template ID = 1
Act Queue Time Histogram Template ID = 1
Act Execute Time Histogram Template ID = 1
Act Estimated Cost Histogram Template ID = 1
Act Interarrival Time Histogram Template ID = 1
Request Execute Time Histogram Template ID = 1

Access Count           = 0
Last Stats Reset Time = 12/16/2008 15:27:42.000000
Activities HWM         = 0
Activities Completed   = 0
Activities Rejected    = 0
```

## db2pd - Monitor and troubleshoot DB2 database

Activities Aborted = 0

Associated Agents:

EDU ID	AppHandl	[nod-index]	WL ID	WLO ID	UOW ID	Activity ID
26	10	[000-00010]	0	0	0	0
29	11	[000-00011]	0	0	0	0
28	12	[000-00012]	0	0	0	0
27	13	[000-00013]	0	0	0	0
30	14	[000-00014]	0	0	0	0

Associated Non-agent threads:

PID	TID	Thread Name
6834	2948590480	db2loggr
6834	2947541904	db2loggw
6834	2946493328	db2lfr
6834	2945444752	db2dlock
6834	2944396176	db2pclnr
6834	2943347600	db2pfchr
6834	2942299024	db2pfchr
6834	2941250448	db2pfchr

### -storagegroups and -storagepaths

The following section is an example of output using the **-storagegroups** parameter or the **-storagepaths** parameter.

```
db2pd -db testdb -storagegroups
```

Storage Group Configuration:

Address	SGID	DeflIt	DataTag	Name
0x00002BA9E6CFF4C0	0	Yes	0	IBMSTOGROUP
0x00002BA9E6D0F4C0	1	No	1	SG_SSD
0x00002BA9E6D1DAE0	2	No	5	SG_IBMSAN

Storage Group Statistics:

Address	SGID	State	NumPaths	NumDropPen
0x00002BA9E6CFF4C0	0	0x00000000	1	0
0x00002BA9E6D0F4C0	1	0x00000000	2	0
0x00002BA9E6D1DAE0	2	0x00000000	2	0

Storage Group paths:

Address	SGID	PathID	PathState	PathName
0x00002BA99CD23540	0	0	InUse	/filesystem1
0x00002BA99CE13540	1	1024	InUse	/filesystem2
0x00002BA99CF03540	1	1025	InUse	/filesystem3
0x00002BA99D0F3540	2	2048	InUse	/filesystem4
0x00002BA99D1E3540	2	2049	InUse	/filesystem5

### -tablespaces

The following example is a sample of the output of the **db2pd -tablespaces** command showing information such as PathsDropped and PathID that is applicable to databases (some of the columns have been left out for readability):

Tablespace Configuration:

...

Tablespace Statistics:

Address	Id	...	State	MinRecTime	NQuiescers	PathsDropped
0x070000004108AB40	0	...	0x00000000	0	0	Yes
0x070000004108B520	1	...	0x00000000	0	0	Yes
0x0700000041078100	2	...	0x00000000	0	0	Yes

Tablespace Autoresize Statistics:

...

Tablespace Storage Statistics:

Address	Id	DataTag	Rebalance	SGID	SourceSGID
---------	----	---------	-----------	------	------------

## db2pd - Monitor and troubleshoot DB2 database

```
0x00002BA9E6CFF4C0 0 0 No 0 -
0x00002BA9E6D0F4C0 1 5 No 0 -
0x00002BA9E6D1DAE0 2 1 Yes 1 0
0x00002BA9E73696C0 3 5 No 0 -
```

### Containers:

Address	TspId	...	PathID	StripeSet	Container
0x070000004108B240	0	...	0	0	/dataPath1/inst/NODE0000/TESTDB/T0000000/C0000000.CAT
0x070000004108B398	0	...	1	0	/dataPath2/inst/NODE0000/TESTDB/T0000000/C0000001.CAT
0x070000004108BBC0	1	...	0	0	/dataPath1/inst/NODE0000/TESTDB/T0000001/C0000000.TMP
0x070000004108BD18	1	...	1	0	/dataPath2/inst/NODE0000/TESTDB/T0000001/C0000001.TMP
0x07000000410787A0	2	...	0	0	/dataPath1/inst/NODE0000/TESTDB/T0000002/C0000000.LRG
0x07000000410788F8	2	...	1	0	/dataPath2/inst/NODE0000/TESTDB/T0000002/C0000001.LRG

If the table is managed by manual storage, the SGID will output a dash (-).

A new 'Max HWM' column is added to the **db2pd -tablespaces** output to indicate the maximum HWM for a DMS table space since the instance was started. The 'HWM' column in the output is the current HWM, which for a temporary DMS table space, represents the point-in-time value of the amount of disk space used. For SMS table spaces, the HWM and Max HWM will not have any value.

After a query has been issued, in-memory information about the temporary tables used in the last transaction will be available using **db2pd**. The following example shows the new column in **bold**. The value of the Max HWM will always be equal to, or greater than, the HWM.

```
hotel26:/home/billyp> db2pd -db bill -tablespaces
```

```
Database Partition 0 -- Database BILL -- Active -- Up 0 days 00:02:15
```

### Tablespace Configuration:

Address	Id	Type	Content	PageSz	ExtentSz	Auto	Prefetch	BufID	BufIDDisk	FSC	NumCntrs	MaxStripe	LastConsecPg	Name
0x00002B9DCA582720	0	DMS	Regular	4096	4	Yes	4	1	1	Off	1	0	3	SYSCATSPACE
0x00002B9DCA583560	1	DMS	UsrTmp	4096	2	Yes	2	1	1	Off	1	0	1	DMSUSRTEMP
0x00002B9DCA5863E0	2	DMS	Large	4096	32	Yes	32	1	1	Off	1	0	31	USERSPACE1
0x00002B9DCA587220	3	DMS	SysTmp	4096	2	Yes	2	1	1	Off	1	0	1	DMSSTEMP
0x00002B9DCA58A0A0	4	DMS	Large	4096	4	Yes	4	1	1	Off	1	0	3	SYSTOOLSPACE

### Tablespace Statistics:

Address	Id	TotalPgs	UsablePgs	UsedPgs	PndFreePgs	FreePgs	HWM	<b>Max HWM</b>	State	MinRecTime	NQuiescers
0x00002B9DCA582720	0	12544	12540	12308	0	232	12308	<b>12308</b>	0x00000000	0	0
0x00002B9DCA583560	1	20000	19998	3266	0	16732	3266	<b>3266</b>	0x00000000	0	0
0x00002B9DCA5863E0	2	7168	7136	3232	0	3904	7072	<b>7072</b>	0x00000000	0	0
0x00002B9DCA587220	3	20000	19998	1700	0	18298	1700	<b>2000</b>	0x00000000	0	0
0x00002B9DCA58A0A0	4	256	252	144	0	108	144	<b>200</b>	0x00000000	0	0

### -temptable

The system monitor elements could also be used to determine the effectiveness of temporary table compression by examining the amount of buffer pool reads and writes. The following example is a sample of the output of the **db2pd -temptable** command:

```
hotel26:/home/billyp> db2pd -db billdb -temptable
```

### System Temp Table Stats:

```
Number of Temp Tables : 0
Comp Eligible Temps : 0
Compressed Temps : 0

Total Stored Temp Bytes : 0
Total Bytes Saved : 0
Total Compressed Rows : 0
Total Temp Table Rows : 0
```

### User Temp Table Stats:

```
Number of Temp Tables : 0
Comp Eligible Temps : 0
Compressed Temps : 0
```

## db2pd - Monitor and troubleshoot DB2 database

```
Total Stored Temp Bytes : 0
Total Bytes Saved       : 0
Total Compressed Rows   : 0
Total Temp Table Rows   : 0
```

The same information is stored for system temporary tables as well as user temporary tables. However, all of the counters mentioned previously are cumulative, and are updated as temporary tables are dropped. As such, these counters represent only historical information.

### -thresholds

The following example is a sample of the threshold information for a database threshold and its queue.

```
Threshold Name           = MAXDBACTIVITIES
Threshold ID             = 6
Domain                  = 10
Domain ID               = 10
Predicate ID            = 90
Maximum Value           = 2
Enforcement              = D
Queueing                = Y
Queue Size              = 0
Collect Flags           = V
Partition Flags         = C
Execute Flags           = C
Enabled                 = Y
Check Interval (seconds) = -1
Remap Target Serv. Subclass = 0
Log Violation Evmon Record = Y
```

Sample database threshold queue output:

Database Threshold Tickets:

```
Ticket information for threshold: TH1 with threshold ID 1
Activity ID UOW ID      Classification AppHandl [nod-index]
1           6           READ_DML      51      [000-00051]
```

```
Queue information for threshold: MAXDBACTIVITIES
Max Concurrency           = 2
Concurrency               = 2
Max Queue Size            = 0
```

Agents Currently Queued:

```
EDU ID      AppHandl [nod-index]      Agent Type  Activity ID  UOW ID
36          14994   [000-14994]  1           4           1
```

The following example is a sample of the threshold information for a statement threshold:

```
db2pd -thresholds -db sample db2
```

```
Database Member 0 -- Database SAMPLE -- Active -- Up 0 days 00:01:28 -- Date 04/13/2011 09:57:09
```

Statement Thresholds:

```
Threshold Name           = T_THRESHOLD_FIRING_23
Threshold ID             = 1
Domain                  = 60
Domain ID               = 1
Predicate ID            = 30
Maximum Value           = 60
Enforcement              = D
Queueing                = N
Queue Size              = 0
Collect Flags           = V
Partition Flags         = C
Execute Flags           = C
Enabled                 = Y
Check Interval (seconds) = -1
Remap Target Serv. Subclass = 0
Log Violation Evmon Record = Y
Statement Text           = CREATE TABLE T2 (X INT)
```

## db2pd - Monitor and troubleshoot DB2 database

```
$ db2 "select thresholdname, domain from sysibm.systhresholds"
```

```
THRESHOLDNAME                                     DOMAIN
-----
T_THRESHOLD_FIRING_23                             SQ
1 record(s) selected.
```

**-sort** The following example is a sample of the output of the **db2pd -sort** command:

### -wlocks

The following example is a sample of the output of the **db2pd -wlocks** command:

```
db2pd -wlocks -db mydb2
```

```
Database Partition 0 -- Database MYDB2 -- Active -- Up 0 days 00:02:17
```

```
Locks being waited on :
```

AppHandl	[nod-index]	TranHdl	Lockname	Type	Mode	Conv	Sts	CoorEDU	AppName	AuthID	AppID
13	[000-00013]	7	0002000B00000000340000452	Row	..X		G	352614	db2bp	VENUS	*LOCAL.venus.071117030309
15	[000-00015]	9	0002000B00000000340000452	Row	.NS		W	1176046	db2bp	VENUS	*LOCAL.venus.071117030358
12	[000-00012]	2	0002000B00000000340000452	Row	.NS		W	1052748	db2bp	VENUS	*LOCAL.venus.071117030231
12	[000-00012]	2	0002000400000000080001652	Row	..X		G	1052748	db2bp	VENUS	*LOCAL.venus.071117030231
14	[000-00014]	8	0002000400000000080001652	Row	.NS		W	634900	db2bp	VENUS	*LOCAL.venus.071117030340

### -workclasssets

The following example is a sample of the output for the basic work class information:

```
Work Class Sets:
```

Address	ClassSetID	ReferenceCounter
0x00002BA89DDF5AE0	1	1

```
Work Classes:
```

Address	ClassSetID	ClassID	ClassName
0x00002BA89DDF5BC0	1	1	WCDDL

```
Attributes:
```

```
Work Type: DML
```

```
Timeron Cost From: 1 Timeron Cost To: 1000
```

Address	ClassSetID	ClassID	ClassName
0x00002BA89DDF5C40	1	2	WCDDL

```
Work Type: DML
```

### -workloads

The following example is a sample of the output for the default workloads SYSDEFAULTUSERWORKLOAD and SYSDEFAULTADMWORKLOAD:

```
Database Partition 0 -- Database SB -- Active -- Up 0 days 00:00:57
```

```
Workload Definitions:
```

```
Address = 0x00002B3E772ACB40
WorkloadID = 1
WorkloadName = SYSDEFAULTUSERWORKLOAD
DBAccess = ALLOW
Maximum Degree = 4
ConcWLOThresID = 0
ConcWLOThresName = ^H
MaxConcWLOs = 9223372036854775806
WLOActsThresName = ^H
WLOActsThresID = 0
MaxWLOActs = 9223372036854775806
ServiceClassID = 13
Collect Activity Opt = None
Collect Lock Timeout = Without History
Collect Deadlock = Without History
Collect Lock Wait = None
Collect Aggr Activity Opt = None
Collect Activity Metrics = Base
```

## db2pd - Monitor and troubleshoot DB2 database

```
Collect Unit of Work Data = None
Act Lifetime Histogram Template ID = 1
Act Queue Time Histogram Template ID = 1
Act Execute Time Histogram Template ID = 1
Act Estimated Cost Histogram Template ID = 1
Act Interarrival Time Histogram Template ID = 1

Address = 0x00002B3E772ACD50
WorkloadID = 2
WorkloadName = SYSDEFAULTADMWORKLOAD
DBAccess = ALLOW
Maximum Degree = DEFAULT
ConcWLOThresID = 0
ConcWLOThresName = ^H
MaxConcWLOs = 9223372036854775806
WLOActsThresName = ^H
WLOActsThresID = 0
MaxWLOActs = 9223372036854775806
ServiceClassID = 13
Collect Activity Opt = None
Collect Lock Timeout = Without History
Collect Deadlock = Without History
Collect Lock Wait = None
Collect Aggr Activity Opt = None
Collect Activity Metrics = Base
Collect Unit of Work Data = None
Act Lifetime Histogram Template ID = 1
Act Queue Time Histogram Template ID = 1
Act Execute Time Histogram Template ID = 1
Act Estimated Cost Histogram Template ID = 1
Act Interarrival Time Histogram Template ID = 1
```

### Usage Privilege Holders:

Address	WorkloadID	Type	AuthID
0x00002B3E772BCD60	1	GROUP	PUBLIC

### Local Partition Workload Statistics:

```
Address = 0x00002B3E772DA0C0
WorkloadID = 1
WorkloadName = SYSDEFAULTUSERWORKLOAD
NumWLO = 0
LastResetTime = 10/07/2008 16:34:43.000000
WLO HWM = 0
WLOActHWM = 0
WLOCompleted = 0
ActCompleted = 0
ActAborted = 0
ActRejected = 0
```

```
Address = 0x00002B3E7730A0C0
WorkloadID = 2
WorkloadName = SYSDEFAULTADMWORKLOAD
NumWLO = 0
LastResetTime = 10/07/2008 16:34:43.000000
WLO HWM = 0
WLOActHWM = 0
WLOCompleted = 0
ActCompleted = 0
ActAborted = 0
ActRejected = 0
```

## 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 authority levels is required:

- The SYSADM authority level.
- The SYSCTRL authority level.
- The SYSMANT authority level.
- The SYSMON authority level.

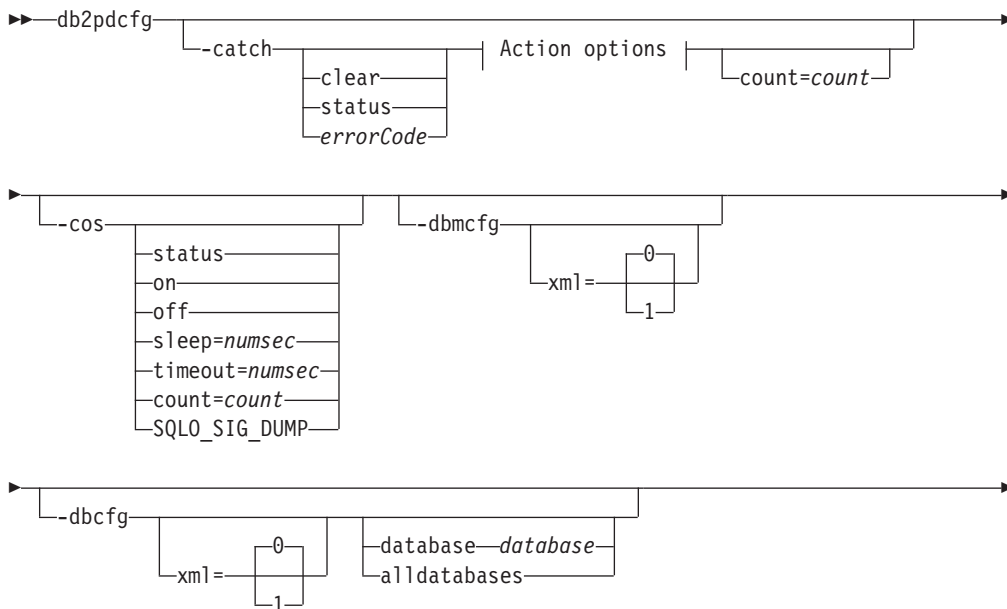
When the SYSMON authorization level is granted, the following options are not available:

- **catch**
- **cos**
- **dbcfg**
- **dbmcfg**
- **fodc**
- **trapresilience**

### Required connection

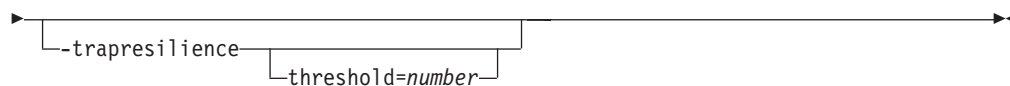
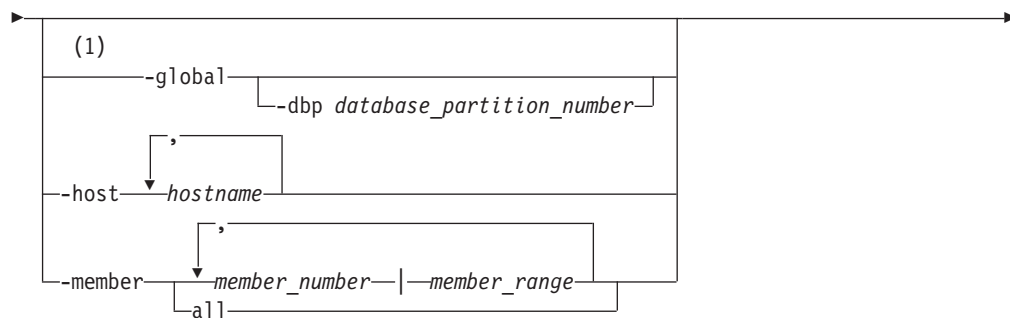
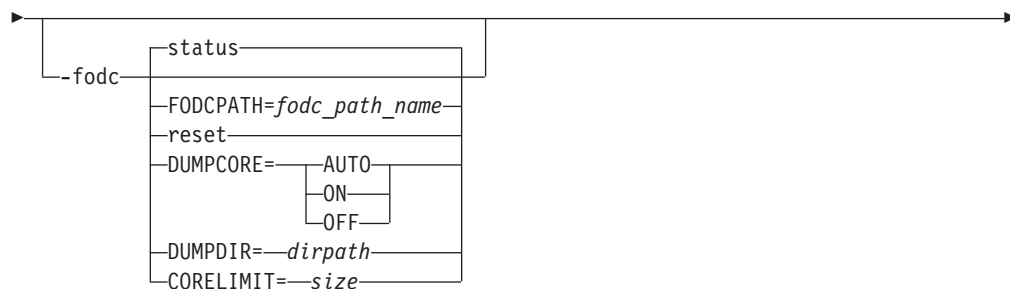
There is no minimum connection requirement. However, if a database scope option is specified, that database must be active before the command can return the requested information.

### Command syntax

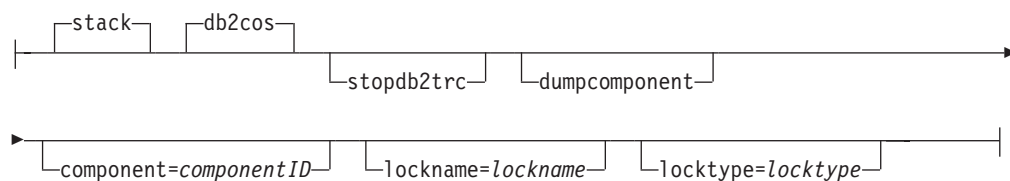




## db2pdcfg - Configure DB2 database for problem determination behavior



### Action options:



### Notes:

- 1 The **-global** parameter has been deprecated. You can use the **-member all** parameter options to obtain information globally.

### Command parameters

**-catch** Instructs the database manager to catch an error or warning.

**clear** Clear any catch flags that are set.

**status** Display any catch flags that are set.

*errorCode*

Catch specific flags that are set.

Possible *errorCode* options are:

- *sqlCode[,reasonCode]* / *sqlCode=sqlCode[,reasonCode]*
- ZRC (hex or integer)
- ZRC #define (such as SQLP\_LTIMEOUT)
- ADM (such as ADM1611)
- String (such as diagstr="Hello World")

## db2pdcfg - Configure DB2 database for problem determination behavior

- ECF (hex or integer)
- “deadlock” or “locktimeout”

Catching specific flags that you specify, such as an administration notification message (ADM1611) or any string (“Hello World”) observed in the **db2diag** log file, causes the **db2cos** script to dump out this information.

**stack** Produce stack trace in **db2diag** log file. Default.

### **db2cos**

Run the **db2cos** callout script found in the bin directory. Default.

### **stopdb2trc**

Stop **db2trc** command.

### **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).

**count=count**

The number of times the database manager executes **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.

**status** Print the status.

**off** Turn off the database manager call to **db2cos** during a database manager trap.

**on** Turn on the database manager call to **db2cos** during a database manager trap.

**sleep=numsec**

Amount of time to sleep between checking the size of the output file generated by **db2cos**. The default is 3 seconds.

**timeout=numsec**

Amount of time to wait before assuming **db2cos** script has completed. The default is 30 seconds.

**count=count**

The number of times to execute **db2cos** during a database manager trap. The default is 255.

### **SQLQ\_SIG\_DUMP**

Enable **db2cos** execution when SQLQ\_SIG\_DUMP signal is received.

### **-dbmcfgr**

Sets DBM Config Reserved Bitmap. This option is password protected which can be obtained from IBM DB2 Service.

## db2pdcfg - Configure DB2 database for problem determination behavior

**xml=0 | 1**

Values 0 (default) or 1 (instance has xml data).

**-dbcfg** Sets Database Config Reserved Bitmap. This option is password protected which can be obtained from IBM DB2 Service.

**xml=0 | 1**

Values 0 (default) or 1 (database has xml data).

**-fodc** Sets flags in the DB2 database memory sets. This influences the database system behavior during problem determination situations involving first occurrence data collection (FODC).

The supported **-fodc** options with their potential values and defaults are:

**reset** Restore all the FODC options to their defaults.

**status** Display status of all FODC options. This is a default option, that is, FODC status will be displayed when **db2pdcfg** is invoked without parameters.

**FODCPATH=fodc\_path\_name**

Specifies the full path name of an existing directory where the FODC data package is created. The path you specify must be writable by the members on the machine and by the fmp processes running on the member or partition.

**DUMPCORE=**

Enables or disables core file generation on only UNIX and Linux operating systems.

**AUTO**

Core file is generated if trap cannot be sustained and instance is shut down.

**ON** Enables core file generation and overrides **DB2RESILIENCE** registry variable setting.

**OFF** Disables core file generation.

**DUMPPDIR=dirpath**

Specifies absolute pathname of the directory where core file or shared memory dump will be created. This option may be used for other large binary dumps that have to be stored outside of the FODC package, not only core file and shared memory dumps. Default is **diagpath** directory or the default diagnostic directory if **diagpath** is not defined.

**CORELIMIT=size**

The maximum size of core files created. This value will override the current core file size limit setting. A default value of unlimited is enforced. Consideration should be given to the available file system space because core files can be quite large. The actual size is dependent on the DB2 configuration and the state of the process at the time the problem occurs.

If **CORELIMIT** is to be changed using **db2pdcfg**, it is subject to the usual UNIX access permissions and in some cases **CORELIMIT** will not be able to exceed your ulimit setting. Use **DB2FODC** registry variable to change that value on **db2start** or use large ulimit setting before starting DB2 product.

## db2pdcfg - Configure DB2 database for problem determination behavior

### **-global**

Specifies that **db2pdcfg** will also be run on remote hosts. If the **-file** parameter is specified, a single file consisting of all the individual files from remote host will be created on the computer from where you issued the **db2pdcfg** command.

**Note:** This command parameter is available in DB2 Version 9.8 Fix Pack 3 and later fix packs. This command parameter is deprecated in DB2 Version 9.7 Fix Pack 4 and later fix packs.

### **-dbp** *database partition number*

Specifies that **db2pdcfg** will be run on the remote host of the specified database partition. If no database partition is specified with the **-global** option, **db2pdcfg** will run on all remote hosts.

### **-host** *hostname*

Specifies the host or hosts on which the command is issued. The command is issued for all members that reside on the host. If this option is not specified, the command is issued on the local host. If multiple hosts are specified, all host names must be valid for the command to complete. This option cannot be specified together with the **-member** option.

### **-member** *member\_number* | *member\_range*

Specifies the member or members on which the command is issued. If this option is not specified, the command is issued on the current member. Multiple members can be specified as a comma separated list of *member\_number* (member1, member2), or using *member\_range*, where *member\_range* is a range of members (member1-member3), or using any combination of the first two methods. This option cannot be specified together with the **-host** option.

**all** Specifies that the command is issued on all members, including members on remote hosts.

### **-trapresilience**

This option displays or modifies trap resilience parameters for problem determination purposes.

The following is a sample output when this option is specified:

```
DB2 trap resilience is enabled.  
Current threshold setting : 0 (threshold disabled)  
Number of traps sustained : 0
```

### **threshold=number**

Default value: 0 (threshold disabled)

Minimum value: 0

Maximum value: 4294967295

Specifying a number after the **threshold=** option sets the upper limit of traps which will be sustained for the life of the instance. When this threshold is surpassed, the instance will be stopped regardless if the next trap could have been sustained.

The following is a sample output when this option is specified:

```
db2pdcfg -trapresilience threshold=1
```

```
DB2 trap resilience threshold is set to 1
```

## db2pdcfg - Configure DB2 database for problem determination behavior

### Examples

To display the current FODC package settings on members 11, 12, 13, and 15:

```
db2pdcfg -fodc status -member 11-13,15
```

To direct the FODC package to a directory on the local host:

```
db2pdcfg -fodc FODCPATH=/home/hotel149/user/FODC/FODClocal
```

### Usage notes

**db2pdcfg** is a method for dynamically changing (online) the FODC options.

Since **db2pdcfg** sets flags in the DB2 database memory, changes done with the **db2pdcfg** tool will be active only while the instance is up. In order to make the changes permanent, use the **DB2FODC** registry variable.

In the **-fodc** option, some of the settings are specified with the format *variable=value*. Multiple options can be specified in a single command line:

```
db2pdcfg -fodc DUMPCORE=ON -fodc CORELIMIT=8GB
```

Alternatively, several settings can be concatenated in a single command line string using spaces:

```
db2pdcfg -fodc DUMPCORE=ON CORELIMIT=8GB
```

Executing the **db2pdcfg** command, without any options, provides the following informative summary output with respect to trap resilience (highlighted in bold):

```
$ db2pdcfg
Current PD Control Block Settings:

All error catch flag settings cleared.
```

```
db2cos is enabled for engine traps.
PD Bitmap: 0x1000
Sleep Time: 3
Timeout: 300
Current Count: 0
Max Count: 255
```

```
Current bitmap value: 0x0
```

```
Instance is not in a sleep state
```

```
DB2 trap resilience is enabled.
Current threshold setting : 0 ( disabled )
Number of traps sustained : 0
```

```
Database Member 0
```

```
FODC (First Occurrence Data Capture) options:
Dump directory for large objects (DUMPDIR)= /home/hotel185/vivmak/sql1lib/db2dump/
Dump Core files (DUMPCORE)= AUTO
Current hard core file size limit = Unlimited
Current soft core file size limit = 0 Bytes
```

---

### db2perf - 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 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.

#### Examples

The following example resets performance values for all active DB2 databases:  
db2perf

The following example resets performance values for specific DB2 databases:  
db2perf dbalias1 dbalias2

The following example resets performance values for all active DB2 DCS databases:  
db2perf -d

The following example resets performance values for specific DB2 DCS databases:  
db2perf -d dbalias1 dbalias2

#### 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 **db2perf**). Invoking **db2perf** also resets the values seen by anyone remotely accessing DB2 performance information when the command is executed.

## **db2perf - Reset database performance values**

The db2ResetMonitor API allows an application to reset the values it sees locally, not globally, for particular databases.

---

## 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 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 action:

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
```



## db2perfr - Performance monitor registration tool

The **db2perfr** command is used to register an administrator user name and password with DB2 when accessing the performance monitor on Windows operating systems.

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 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 the DB2 database system, 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 the DB2 database system 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 reserved word in DB2 database products and cannot be used.

---

## db2rbind - Rebind all packages

Rebinds packages in a database.

### Authorization

DBADM

### 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** *logfile*

Specifies the (optional) path and the (mandatory) file name to be used for recording the package revalidation process.

*Example:*

```
cat <logfile>
Starting time .... Thu Jun 18 02:47:11 2009
Succeeded to rebind = 0
Failed to rebind = 0
Ending time .... Thu Jun 18 02:47:11 2009
```

**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** *userid*

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

**-p** *password*

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 objects that use the SQL path for resolution are considered during resolution on static DML statements in the package. This option is not supported by DRDA. Valid values are:

**conservative**

Only those objects in the SQL path that were defined before the last explicit bind time stamp are considered for resolving references to any objects that use the SQL path for object resolution.

## db2rbind - Rebind all packages

Conservative binding semantics are used. This is the default. This option is not supported for an inoperative package.

- any** All possible matches in the SQL path are considered for resolving references to any objects that use the SQL path for object resolution. Conservative binding semantics are not used.

### Usage notes

- This command uses the rebind API (sqlarbind) to attempt the re-validation 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.
- If you issue the **db2rbind** command and the instance is active, you will see the SQL1026N error message.

---

### 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.

The target database must be offline before running the **db2relocatedb** command to modify the control files and metadata of the target database.

The changes that the **db2relocatedb** command makes to files and control structures of a database are not logged and are therefore not recoverable. A good practice is to make a full backup after running the command against a database, especially if the database is recoverable with log files being retained.

#### 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
...
FAILARCHIVE_PATH=newDirPath
LOGARCHMETH1=newDirPath
LOGARCHMETH2=newDirPath
MIRRORLOG_PATH=newDirPath
OVERFLOWLOG_PATH=newDirPath
...
```

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.

**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**

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. You can specify this parameter to modify any storage path in all storage groups. However, you cannot specify this parameter to modify the storage paths for an individual storage group.

**Note:** This parameter is not applicable to a database created with the AUTOMATIC STORAGE NO clause. Although, you can create a database specifying the AUTOMATIC STORAGE NO clause, the AUTOMATIC STORAGE clause is deprecated and might be removed from a future release.

**FAILARCHIVE\_PATH**

Specifies a new location to archive log files if the database manager fails to archive the log files to either the primary or the secondary archive locations. You should only specify this field if the database being relocated has the **failarchpath** configuration parameter set.

**LOGARCHMETH1**

Specifies a new primary archive location. You should only specify this field if the database being relocated has the **logarchmeth1** configuration parameter set.

**LOGARCHMETH2**

Specifies a new secondary archive location. You should only specify this field if the database being relocated has the **logarchmeth2** configuration parameter set.

**MIRRORLOG\_PATH**

Specifies a new location for the mirror log path. The string must

## db2relocatedb - Relocate database

point to a path name, and it must be a fully qualified path name, not a relative path name. You should only specify this field if the database being relocated has the **mirrorlogpath** configuration parameter set.

### OVERFLOWLOG\_PATH

Specifies a new location to find log files required for a rollforward operation, to store active log files retrieved from the archive, and to find and store log files required by the db2ReadLog API. You should only specify this field if the database being relocated has the **overflowlogpath** configuration parameter set.

Blank lines or lines beginning with a comment character (#) are ignored.

## 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 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
NODENUM=10
```

On database partition server 20, the following configuration file will be used:

## db2relocatedb - Relocate database

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODENUM=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
NODENUM=0
CONT_PATH=/maininst_files/allconts/*, /MAINDB/*
```

### 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. The new instance must be at the same release level as the instance where the database currently resides.
- If the new instance has a different owner than the current instance, grant access to the new instance owner.
- 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.

When moving a database from a database path where more than one database resides, the sqlbdir directory within that database path must be copied and not moved. This directory is still needed in the old location for DB2 to locate the databases that are not moving. After copying the sqlbdir directory to the new location, a LIST DB DIRECTORY ON *newPath* command lists databases that were not moved. These references cannot be removed and new databases with those names cannot be created on this same path. However, databases can be created with those names on a different path.



The **db2relocatedb** command cannot be used to move existing user created containers for a table space that was converted to use automatic storage using the ALTER TABLESPACE MANAGED BY AUTOMATIC STORAGE statement.

If the instance is changing, the command 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.

You cannot use the **db2relocatedb** command to relocate a database that has a load in progress or is waiting for the completion of a **LOAD RESTART** or **LOAD TERMINATE** command.

**Limitation:** In a partitioned database environment, you cannot relocate an entire node if that node is one of two or more logical partitions that reside on the same device.

---

## db2rfe - Enable root features for non-root installations

Enables the supported root features, in non-root installations of DB2 database systems, according to the configuration file. The DB2 non-root instance needs to be stopped before the **db2rfe** command is executed.

### Authorization

Root user authority

### Required Connection

None

### Command syntax

```
▶▶ db2rfe -f db2rfe_config_file [ -h | -? ] ▶▶▶▶
```

### Command parameters

- f** *db2rfe\_config\_file*  
Specifies the configuration file to be used to enable root features.
- h | -?**  
Displays help information.

### Usage notes

Each root feature, in the configuration file, will be in a separate section. Each section will have a start and end mark, comments describing what the section will enable, and the command to enable the root feature. The sample configuration file *db2rfe.cfg* will be installed to the *\$DB2DIR/instance* directory.

The sample configuration file will look like the following (the non-root install owner is *db2inst3* in this example):

```
** =====  
**  
** Sample configuration file for db2rfe of IBM DB2  
** -----  
**  
** To select features and settings to configure, uncomment the corresponding  
** keywords and specify values for those keywords.  
**  
** Comments are made by placing either an asterisk (*) or a number sign (#) at  
** the start of a line  
**  
** =====  
  
INSTANCENAME=db2inst3  
** This is required keyword.  
  
** -----  
** Set hard/soft data ulimit to unlimited, and hard/soft nofile ulimit to 65536.  
**
```

## db2rfe - Enable root features for non-root installations

\*\* Note: This is for AIX only. On other platforms, refer to system documentation  
\*\* to set it manually.

\*\* -----  
\*\* Valid value is NO and YES. Change to YES if you need to set the ulimit.

SET\_ULIMIT=NO

\*\* -----  
\*\* Enable DB2 High Availability (HA) feature

\*\* -----  
\*\* Valid value is NO and YES. Change to YES if you need to enable this feature.

ENABLE\_HA=NO

\*\* -----  
\*\* ENABLE DB2 Authentication on the server using local operating system security.

\*\* -----  
\*\* Valid value is NO and YES. Change to YES if you need to enable this feature.

ENABLE\_OS\_AUTHENTICATION=NO

\*\* -----  
\*\* Reserve DB2 remote connection service entry

\*\* -----  
\*\* Valid value is NO and YES. Change to YES if you need to enable this feature.

RESERVE\_REMOTE\_CONNECTION=NO

\*SVCENAME=db2c\_db2inst3

\*\* char(14)

\*SVCEPORT=48000

\*\* Valid value: 1024 - 65535

\*\* -----  
\*\* Reserve DB2 text search service entry

\*\* -----  
\*\* Valid value is NO and YES. Change to YES if you need to enable this feature.

RESERVE\_TEXT\_SEARCH\_CONNECTION=NO

\*SVCENAME\_TEXT\_SEARCH=db2j\_db2inst3

\*\* char(14)

\*SVCEPORT\_TEXT\_SEARCH=55000

\*\* Valid value: 1024 - 65535

### 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 -path global_log_control_files_dir ▶▶
```

#### 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.

**-path** *global\_log\_control\_files\_dir*

Specifies the full path to the directory where the global log control files, SQLLOGCTL.GLFH.1 and its mirror copy SQLLOGCTL.GLFH.2, reside.

---

## db2rmicons - Remove DB2 tools from main menu

Removes main menu entries for DB2 tools.

On Linux operating systems, the **db2rmicons** command removes main menu entries for DB2 tools for the current user. The main menu entries for DB2 tools are removed by manually running the **db2rmicons** command, or are removed automatically when specific DB2 commands are run (for example, **db2\_deinstall** or **db2idrop**.) For non-root installations, the **db2\_deinstall** command removes the entries for the DB2 instance related to the non-root installation.

### Authorization

None

### Command syntax

```
▶▶—db2rmicons [ -h ]
```

### Command parameters

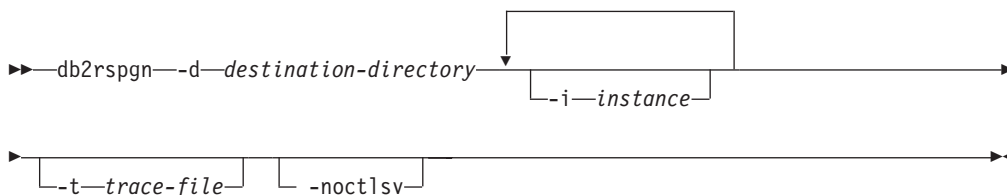
**-h**      Displays usage information.

---

## db2rspgn - Response file generator

Generates response files and instance configuration profiles for the current copy. These generated files are used to re-create an exact setup on other machines.

### Command syntax



### Command parameters

#### `-d destination-directory`

Specifies the full path to the output directory for generated files. If the output directory specified is an existing directory, the directory must be empty and writable. If the output directory specified does not exist, the new directory is created if the location is writable. This parameter is mandatory.

#### `-i instance`

Generates the specified instance configuration and saves this information in the generated response file and instance configuration profile. This parameter is optional. By default, all instances are selected. To specify multiple instances, specify this parameter multiple times. For example, `-i db2inst1 -i db2inst3`.

#### `-t trace-file`

Linux and UNIX operating systems only. Turns on the debug mode. The debug information is written to the file name specified as `trace-file`.

#### `-noctlsv`

Windows operating systems only. Indicates that an instance configuration profile file will not be generated for the Control Server instance. This parameter is optional.

### Usage Notes

The `db2rspgn` command is not supported in a DB2 pureScale environment.

## db2sampl - Create sample database

Creates a sample database named SAMPLE.

**Note:** On DB2 Workgroup Server Edition and DB2 Express Edition, the SAMPLE database includes materialized query tables (MQT), and multidimensional cluster tables (MDC) that causes a license violation. This violation can only be removed by upgrading to DB2 Enterprise Server Edition.

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

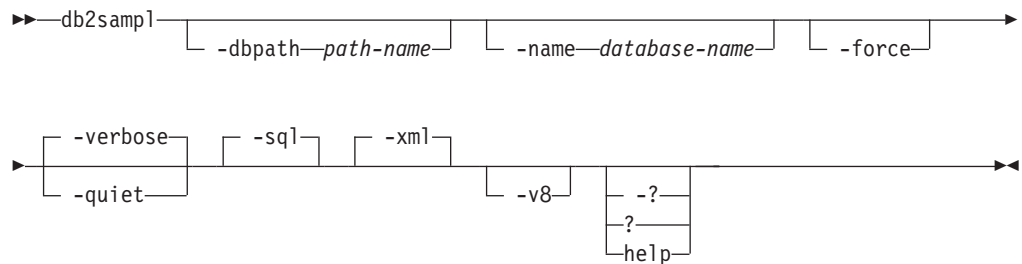
One of the following authorities:

- 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.

**-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.

## db2sampl - Create sample database

- 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.  
  
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).
- ? | ? | help** Returns the **db2sampl** command syntax help.

### Default behavior 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) code set 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 a Unicode (UTF-8) code set 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 tables with data.
- All database object names are qualified with the value of the CURRENT\_SCHEMA special register.

### Usage notes

- The **db2sampl** command can only be issued on a computer where a DB2 database server is installed. It cannot be issued from a remote IBM Data Server Client.
- 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 default schema and to view status messages, issue:  
`db2sampl -dbpath E -name mysample -sql -force -verbose`
- To create the DB2 Version 8 sample database, issue:  
`db2sampl -v8`

---

### db2schex - Active Directory schema extension

Extends the Microsoft Active Directory schema to include the DB2 object classes and attribute definitions that you require to use the Lightweight Directory Access Protocol (LDAP) directory server feature with Windows Server 2003 and later.

You should run this command before installing DB2 products and creating databases otherwise you have to manually register the node and catalog the databases. For more information, see the "Extending the Active Directory Schema for LDAP directory services (Windows)" topic.

The **db2schex** command is included on the product DVD. The location of this command on the DVD is in the path `x:\db2\windows\utilities`, where `x:` specifies the DVD drive.

#### Authorization

To update the Active Directory schema, you must be a member of the Schema Administrators group or have been delegated the rights to update the schema.

#### Required connection

Access to a Windows Domain Controller server in the target domain.

#### Command syntax

```
db2schex [-b bindDN] [-w password] [-k] [-u] [-x filename]
```

#### Command parameters

- b** *bindDN*  
Specifies the user Distinguished Name.
- w** *password*  
Specifies the bind password.
- k**  
Forces the uninstall to continue, ignoring errors.
- u**  
Uninstall the schema.
- x** *filename*  
Specify this parameter to write the changes to the Active Directory schema, performed by the utility, to a file.

#### Examples

To install the DB2 schema, execute the following command:

```
db2schex
```

To install the DB2 schema and specify a bind DN and password, execute the following command:

```
db2schex -b "cn=A_Name,dc=toronto1,dc=ibm,dc=com" -w password
```

or,

```
db2schex -b Administrator -w password
```

## db2schex - Active Directory schema extension

To uninstall the DB2 schema, execute the following command:

```
db2schex -u
```

To uninstall the DB2 schema and ignore errors, execute the following command:

```
db2schex -u -k
```

### Usage notes

If *bindDN* and *password* are not specified, **db2schex** binds as the currently logged in user.

The *bindDN* parameter can be specified as a Windows username.

The DB2 schema extension command carries out the following tasks:

- Detects which server is the Schema Master
- Binds to the Domain Controller that is the Schema Master
- Ensures that the user has sufficient rights to add classes and attributes to the schema
- Ensures that the Schema Master is writable (that is, the safety interlock in the registry is removed)
- Creates all the new attributes
- Creates all the new object classes
- Detects errors and, if they occur, the program will roll back any changes to the schema.

## db2set - DB2 profile registry

Displays, sets, or deletes the values of DB2profile variables. The **db2set** command is an external environment registry command that supports local and remote administration through the DB2 administration server (DAS).

You can also use the ENV\_GET\_REG\_VARIABLES table function to retrieve the values of the registry variables that the instance is using and the values that are stored in the registry.

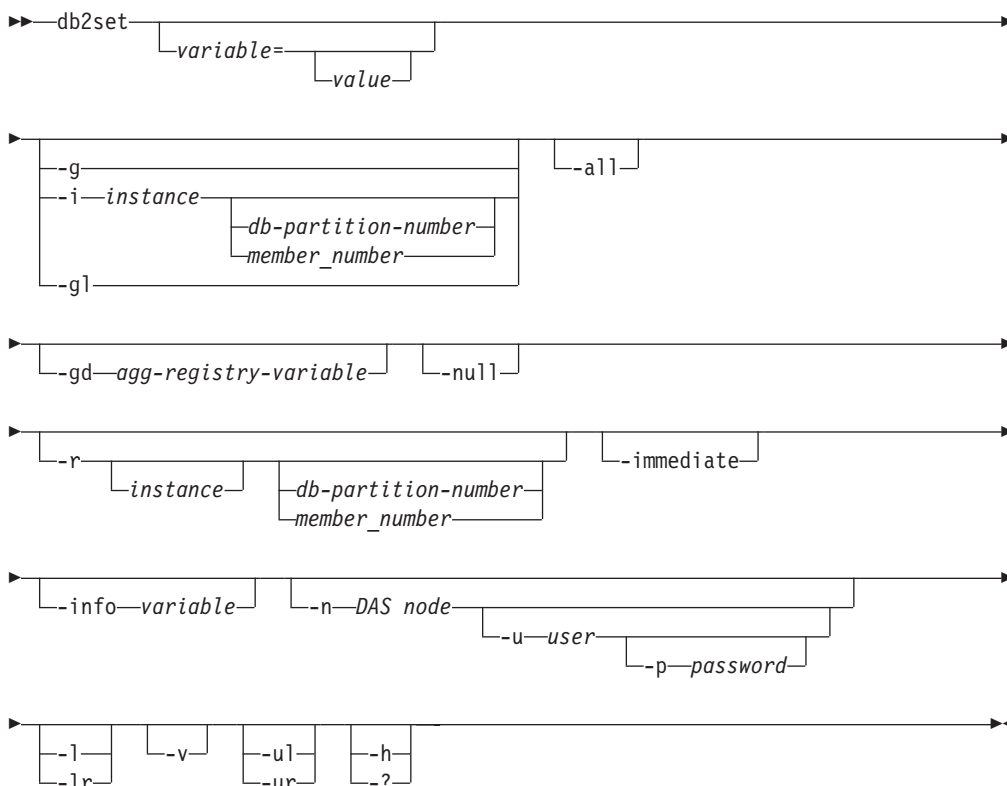
### Authorization

SYSADM and, for the **-g** command parameter, root access on Linux and UNIX operating systems or Local Administrator authority on Windows operating systems.

### Required connection

A local instance attachment is required when updating variables immediately, if there is no instance attachment, a local instance attachment is created. Otherwise, no connection is required. A remote attachment is not supported.

### Command syntax



### Command parameters

*variable=*

Displays the value of the specified variable.

*value* Sets the specified variable to the specified value. If you do not

specify a value after the equal sign (=), the current value of the variable is deleted. There are a set of registry variables which are immediate by default and another set of registry variables that are immediate if you specify the **-immediate** parameter. Both of these sets of registry variables take effect the next time the SQL statement is compiled. Otherwise, changes take effect after you restart the instance.

**-g** Accesses the global profile registry variables for all instances pertaining to a particular DB2 copy. This allows you to view or change the values of these variables at this level.

**-i** *instance*

Specifies the instance profile to use. If you do not specify an instance profile, the current, or default one is used.

*db-partition-number* **or** *member-number*

Specifies a number in the `db2nodes.cfg` file.

If you use the **-immediate** parameter with the *member-number* option, only the specific member sees the update the next time the SQL statement is compiled.

**-gl** Accesses the global profile variables stored in LDAP. This parameter is effective only if you set the **DB2\_ENABLE\_LDAP** registry variable to YES.

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

- The operating system 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]

**-gd** *agg-registry-variable*

Displays the group definition of an aggregate registry variable. For additional information, see “Aggregate registry variables” in the *Data Servers, Databases, and Database Objects Guide*.

**-null** Sets the value of the variable to NULL at the registry level you specified or at the default level, which is the instance level, if you did not specify a registry level. This means that DB2 considers the variable as being not set and will not search for a value for the variable in the next available registry levels, as defined by the order of precedence of the registry levels.

**-r** Resets the profile registry for a particular instance.

*instance*

Specifies the instance for which you want to reset the profile. If you do not specify an instance and an instance attachment exists, this option resets the profile for the current instance. If you do not specify an instance and no attachment exists, this option resets the profile for the instance that is specified by the **DB2INSTANCE** environment variable.

*db-partition-number* **or** *member-number*

Specifies a number in the `db2nodes.cfg` file.

If you use the **-immediate** parameter with the *member-number* option, the value is reset only for the specified member.

## db2set - DB2 profile registry

### **-immediate** | **-im**

Specifies that the update takes effect the next time an SQL statement is compiled for registry variables that support this feature.

Immediate changes to registry variables that affect the SQL compiler will take effect the next time that you compile an SQL statement. There are two types of SQL statements:

#### **Dynamic SQL statements**

If a dynamic SQL statement is already present in the package cache, the statement will not be invalidated and therefore the statement will not be recompiled with new settings. In order for the **-immediate** parameter to take effect, you must issue the **FLUSH PACKAGE CACHE** statement in order to remove previous statements from the package cache so that your SQL statement can be recompiled without having to restart the instance.

#### **Static SQL statements**

If a static SQL statement is already present in a package, the statement will not be invalidated and therefore the statement will not be recompiled with new settings. In order for the **-immediate** parameter to take effect, you must issue the **BIND** command or the **REBIND** command in order for the package to be recompiled without having to restart the instance

You cannot combine this parameter with either the **-g**, **-gl**, or **-n** parameters.

### **-info** *variable*

Returns the properties of the specified variable. The properties state whether an immediate change is supported by the variable and whether the change is immediate by default.

### **-n** *DAS node*

Specifies the remote DAS 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 DB2 product installation.

### **-lr**

Lists all supported registry variables.

### **-v**

Specifies that verbose output is to be used while the command is running.

### **-ul**

Accesses the user profile variables. This parameter is supported on Windows operating systems only.

### **-ur**

Refreshes the user profile variables. This allows multiple users to have different variable settings under the same instance or under the same environment settings. This parameter is supported on Windows operating systems only.

### **-h** | **-?**

Displays help information. If you specify this parameter, all other parameters are ignored.

## **Examples**

The following examples show how to issue the various parameters with the **db2set** command:

- Display all defined instant profiles pertaining to a particular installation:

- ```
db2set -l
```
- Display all supported registry variables:
 

```
db2set -lr
```
- Display all defined global variables that are visible to 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 the **DB2COMM** registry variable for the current instance:
 

```
db2set -all DB2COMM
```
- Reset all defined variables for the instance INST on member 3:
 

```
db2set -r -i INST 3
```
- Delete the value of the **DB2CHKPTR** registry variable 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 **DB2COMM** registry variable to TCPIP for all instances pertaining to a particular installation:
 

```
db2set -g DB2COMM=TCPIP
```
- Set the **DB2COMM** registry variable to TCPIP only for instance MYINST:
 

```
db2set -i MYINST DB2COMM=TCPIP
```
- Set the **DB2COMM** registry variable to null at the default level. The default level is the instance level:
 

```
db2set -null DB2COMM
```
- Delete the current value of the registry variable **DB2\_ANTIJOIN** so that it takes effect the next time the SQL statement is compiled:
 

```
db2set DB2_ANTIJOIN= -immediate
```

## Usage notes

You can set a variable at one or more of four levels: operating-system environment, node instance, instance, and global. The DB2 database system uses this order of precedence to access and resolve variable settings. If you already set the value of a variable at the operating-system-environment level, an update to the value of the variable does not take effect immediately, even if you use the **-immediate** parameter, because the operating-system-environment level takes precedence over the registries.

If you do not specify a variable name, the values of all defined variables are displayed. If you specify a variable name, the value of only that variable is displayed. To display all the defined values of a variable, specify the *variable* and **-all** parameters. To display the values of all the defined variables in all registries, specify the **-all** parameter.

To make registry variable changes for remote registries on Windows operating systems, issue the **db2\_all** or **rah** command with the **db2set** command.

## **db2set - DB2 profile registry**

Although the command behaves the same way for root and for non-root installations of the DB2 product, not all parameters are available, such as the **-n** parameter, which specifies the DAS node name.



## db2setup - Install DB2 database products

Installs DB2 database products. This command is only available on Linux and UNIX operating systems. The command for Windows operating systems is **setup**.

This utility is located on the DB2 database installation media. It launches the DB2 Setup wizard to define the installation and install DB2 database products. If invoked with the **-r** option, it performs an installation without further input, taking installation configuration information from a response file.

### Authorization

Root authority. On Linux and UNIX operating systems, root installations require root user authority. For non-root installations, you must log on with the user ID that owns the non-root installation.

### Command syntax

```

▶▶ db2setup [-i language] [-l log_file] [-t trace_file]
           [-r response_file] [-c] [-f nobackup] [--?]
           [-h]
  
```

### Command parameters

#### **-i** *language*

Two-letter language code of the language in which to perform the installation.

#### **-l** *log\_file*

Writes the log to the file name specified. For root installations, the default log file is `/tmp/db2setup.log`. For non-root installations, the default log file is `/tmp/db2setup_userID.log`, where *userID* represents the user ID that owns the non-root installation. If the IBM Tivoli System Automation for Multiplatforms (SA MP) is being installed with **db2setup**, the install log file for SA MP will be located in the same directory as the DB2 database log files.

#### **-t** *trace\_file*

Generates a file with install trace information.

#### **-r** *response\_file*

Full path and file name of the response file to use. This parameter is mandatory when the **-c** parameter is specified.

#### **-c**

Validates the contents of the response file without performing the installation. The validation results are printed to the log file. The location of the log file is printed in the resulting message. When this parameter is specified, you must also specify the **-r** parameter.

#### **-f** *nobackup*

This applies to the non-root upgrade only. Force **db2setup** to not backup installation files when the components are updated. If you choose not to backup the files, the space requirement of the installation directory is reduced. However, choosing not to backup the files also means that if any

## db2setup - Install DB2 database products

errors occur, the DB2 installer will not be able to perform a rollback operation. In this case, you will need to manually clean up the files and reinstall the product.

**-? | -h**

Generates usage information.

### Usage notes

You must log on with the ID that has proper authority or use **su** with the "-" flag (**su -**) to set the process environment as if you had logged in with the ID that has proper authority. If the process environment is not set with the ID that has proper authority, the installation process finishes without errors but you will encounter errors when you run the DB2 copy.

## db2snapcore - DB2 snapcore for Linux and Solaris

On Linux and Solaris operating systems, this command mimics the functionality provided by the snapcore command on AIX operating systems. The command extracts the shared objects list section from the EDU trap file, combines the section together with a core file and compresses the output into an archive which can be sent to IBM Support for analysis.

### Authorization

One of the following authorities:

- Instance owners
- DBADM authority

### Command syntax

```

▶▶ db2snapcore -c corefile -t trapfile -e executable_path
▶
┌-o output_directory┐
└-p┐
  ┌-pax┐
  └-tar┘
└--v┘
▶▶

```

### Command parameters

- c corefile** | **-corefile corefile**  
Specifies a relative or absolute path to the core file.
- t trapfile** | **-trapfile trapfile**  
Specifies a relative or absolute path to the trap file.
- e executable\_path** | **-executable executable\_path**  
Specifies a relative or absolute path to the executable file. This file is either provided by IBM Support or obtained from the trap file under the guidance of IBM Support.
- o output\_directory** | **-outdir output\_directory**  
Specifies an output directory for the compressed output file. The specified output directory must exist.
- p** | **-packCommand**  
Compresses all files into an archive. On Linux operating systems, files are compressed into a pax archive by default. On the Solaris operating system, the **pax** option is not available and files are compressed into a tar archive by default.
  - pax**  
Specifies that the files are compressed using the **pax** (portable archive exchange) command. On the Solaris operating system, the **pax** option is not available.
  - tar**  
Specifies that the files are compressed using the **tar** (tape archive) command. Requires the **tar** command to be installed on the system.
- v** | **-verbose**  
Enables verbose mode.

## db2snapcore - DB2 snapcore for Linux and Solaris

### Example

To archive the extracted shared objects list section from an EDU trap file and a core file, issue a command like the following, replacing core and trap file names as required:

```
db2snapcore -corefile db2sysc.6270.98042.core -trapfile 28930.16.000.trap.txt -e ~/sqllib/adm/db2sysc
```

Depending on which archiving method is used (**pax** command or **tar** command), the file name for the resulting archive is either `db2snapcore.pax.gz` or `db2snapcore.tar.gz`.

---

## db2start - Start DB2

Starts the database manager on the target member or all members. In a DB2 pureScale environment, it may also be used to start the cluster caching facility (CF).

**db2start** can be executed as a system command or a CLP command.

Start DB2 at the server before connecting to a database, precompiling an application, or binding a package to a database.

The **db2start** command launches the DB2 database product installation as a Windows service. The DB2 database product installation on Windows can still be run as a process by specifying the **/D** switch when invoking **db2start**. The DB2 database 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\_time** database manager configuration parameter, the database partitions that have timed out will not have the database manager instance background processes started (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\_time** database manager configuration parameter.

For root-install DB2 copies on Linux and UNIX operating systems, the **db2start** command sets the ulimit value required by the database manager without changing the permanent setting of ulimit for the instance owner ID.

For non-root install, you should set the ulimit for 'data' to 'unlimited' and 'nofiles' to 'unlimited' or the maximum value allowed on the system.

**Related information:**

### db2stat - DB2 process status for Windows

On Windows systems, all of the DB2 processes running under an instance can be displayed using the **db2stat** command.

#### Authorization

None

#### Required connection

None

#### Command syntax

▶—db2stat—▶

#### Command parameters

##### **db2stat**

Outputs all of the DB2 processes running under an instance.

#### Examples

```
C:\Program Files\IBM\SQLLIB\BIN>db2stat
```

##### Environment Strings

```
--> DB2CLP=DB20FADE  
--> DB2INSTANCE=DB2  
--> DB2PATH=C:\Program Files\IBM\SQLLIB
```

##### DB2 Processes

|            |      |      |
|------------|------|------|
| DB2DASRRM  | 1960 | x7A8 |
| DB2MGMTSVC | 2012 | x7DC |
| DB2RCMD    | 1212 | x4BC |
| DB2DASSTM  | 2044 | x7FC |
| DB2SYSTRAY | 724  | x2D4 |
| DB2        | 3100 | xC1C |
| DB2BP      | 3180 | xC6C |
| DB2SYSCS   | 1592 | x638 |
| DB2FMP     | 3468 | xD8C |
| DB2STAT    | 1748 | x6D4 |

#### Usage notes

One thing to note in the Windows case is that because DB2 is thread-based, not process-based, you will only see one process (DB2SYSCS) for all of an instance's EDUs. It is obvious that the same degree of information is not returned in Windows as is returned in Linux/UNIX systems, but it is still useful, at times, to know the process ID for a running instance. For example, you can use the Windows Task Manager utility to determine the CPU and memory usage for a given process ID.

---

## db2stop - Stop DB2

Stops the database manager on the target member or all members. In a DB2 pureScale environment, it may also be used to stop the cluster caching facility (CF).

**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\_time** 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\_time** might experience this behavior. To resolve this behavior, increase the value of **start\_stop\_time**.

**Related information:**

## db2support - Problem analysis and environment collection tool

Collects environment data about either a client or server machine and places the files that contain system data into a compressed file archive.

The **db2support** command that is shipped with DB2 installation images supports only a subset of the command parameters that are available after you install the DB2 product. Until you install the DB2 product, the only **db2support** command parameters that you can use are the **-install** and **-host** parameters.

This tool can also collect basic data about the nature of a problem through an interactive question and answer process with the user.

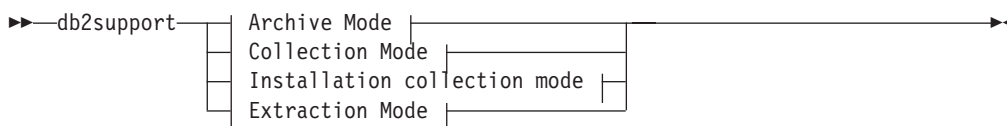
### Authorization

For the most complete output, run this utility with SYSADM authority. Users with more limited privileges on the system can run this tool, however some of the data collection actions results in reduced reporting and reduced output.

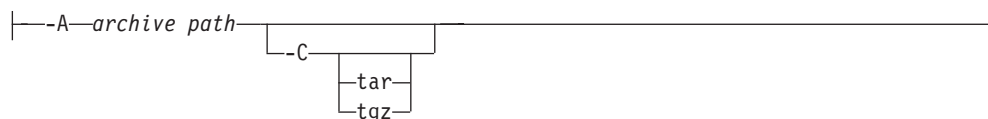
### Required connection

None

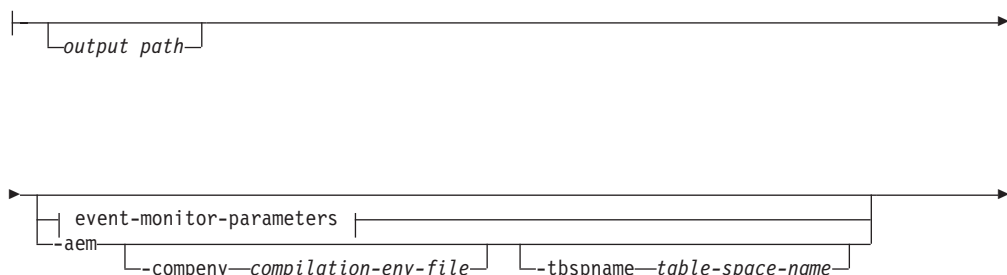
### Command syntax



#### Archive Mode:



#### Collection Mode:





## db2support - Problem analysis and environment collection tool

`--B` `--cd`—*current degree* `--cfg` `--cl`—*collect level*

`--cm` `--cfs` `--udapl` `--co` `--cs`—*current schema*  
`--pc`  
`--ps`  
`--purecluster`  
`--purescale`

`--d`—*database name* `--nco` `--C` `--u`—*userid* `--p`—*password*

`--extenddb2batch` `--fodc`—*list of FODC directories*

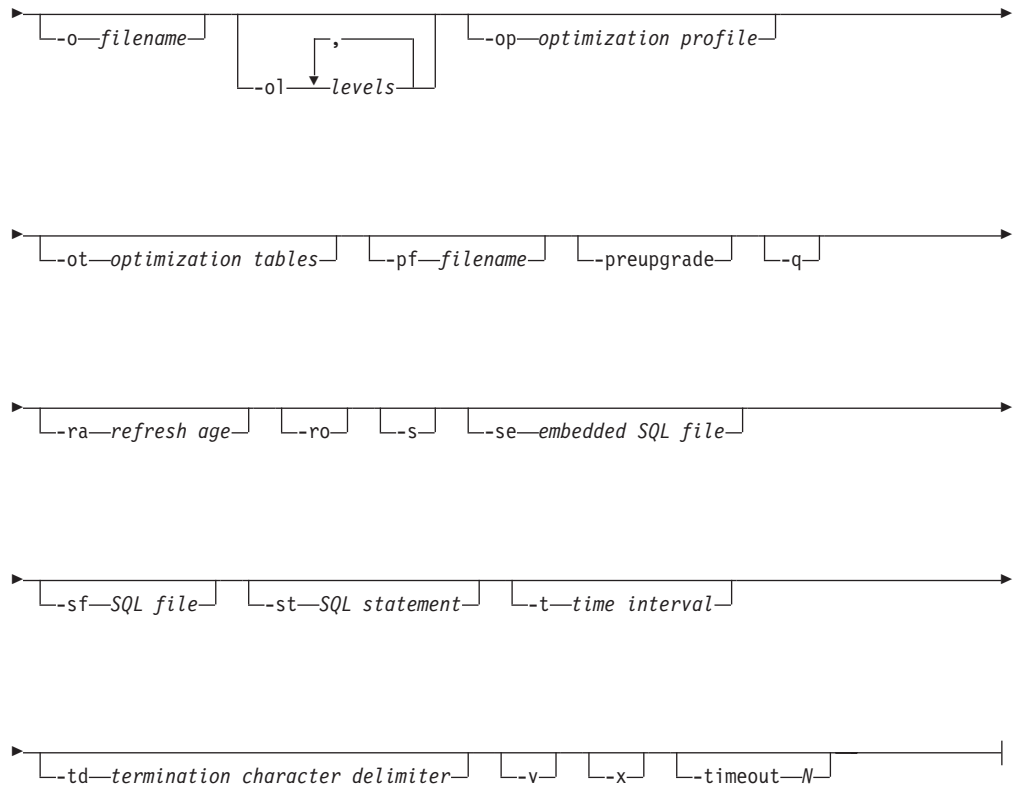
`--fodcpath`—*fodc\_path\_name* `--F` `--fp`—*function path*

`--host` `--hostname` `--localhost` `--member` `--member_number` `--member_range` `--all`

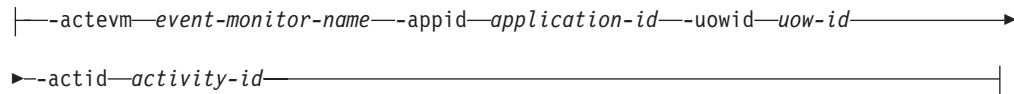
`--sdir`—*shared directory path* `--h` `--H`—*history period*

`--il`—*isolation level* `--l` `--m` `--n` `--nc` `--nl`

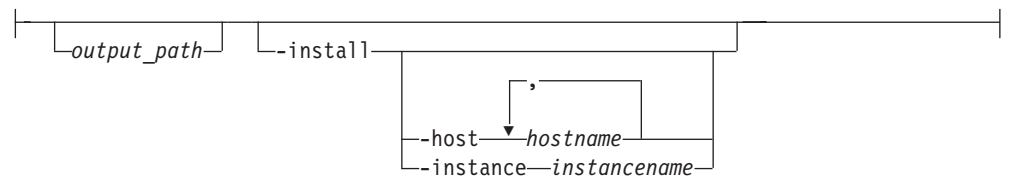
## db2support - Problem analysis and environment collection tool



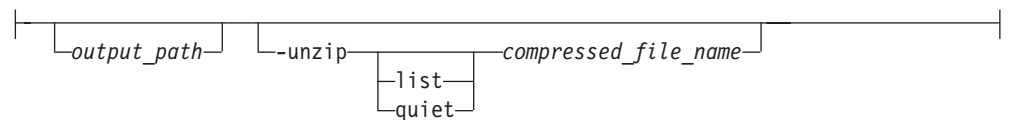
### event-monitor-parameters:



### Installation collection mode:



### Extraction Mode:



### Note:

## db2support - Problem analysis and environment collection tool

1. The **db2support** tool collects bad query-related information only if **-st**, **-sf**, or **-se** parameters are specified. If there is an error or trap during optimization, use the **-cl 0** (collect level zero) option to collect all catalog tables and **db2look** table definitions without trying to explain a bad query. The **-aem** (with **-st** or **-sf**) or event monitor options could be specified if activity event monitor related information needs to be collected as the part of optimizer collection. One of these options must be specified to work with optimizer problems.

**Note:** Every option for the optimizer collection needs to be issued with the **-d <dbname>** option.

2. If special registers are set to values other than the default during the statement execution, pass these values to the **db2support** command to ensure correct problem analysis.

**Note:** The special register options are ignored by **db2caem** collection.

3. The **-global** parameter is discontinued. The **-host all** option is the default behavior of the **db2support** command and thus information from all hosts is collected by default.
4. The **db2support** tool takes a long time to run since it collects most diagnostic data specific to DB2 pureScale components by default. If you specify the **-purecluster**, **-purescale**, **-cm**, **-cfs**, or **-udapl** parameter, the **db2support** tool collects additional diagnostic data that is space intensive or takes a longer time to get collected, but can help speed up the problem determination process in DB2 pureScale environments.

### Command parameters

#### *output path*

Specifies the path where the compressed archive file is to be created or extracted. This path is the directory where user-created files must be placed for inclusion in the archive, or the directory where files are extracted to when the **-unzip** parameter is specified. The current directory is used when this parameter is not specified.

#### **-A** *archive\_path* | **-archive** *archive\_path*

This parameter archives all the data from the directory specified in the **diagpath** configuration parameter into the specified archive path. A new directory will be created in the specified archive path with the name DB2DUMP with the system host name and timestamp appended, for example, **DB2DUMP\_systemhostname\_2009-01-12-12.01.01**.

This parameter also archives all the data from the directory specified in the **alt\_diagpath** configuration parameter into the specified archive path. The name of this directory is **ALT\_DB2DUMP**. Also, files from the **events/** subdirectory are archived into the **ALT\_EVENTS** directory, and files from the **stmmlog/** subdirectory are archived into the **ALT\_STMM** directory.

This parameter is not available on Windows operating systems.

#### **-aem**

Specifies that the **db2caem** information for the SQL statement specified through **-st** or **-sf** is to be collected. **-aem** does not support **-se**. The default behavior for **db2support** is to not collect **db2caem** information. With this parameter, **db2caem** creates an activity event monitor in order to collect the requested information. The activity event monitor and other created tables are cleaned up from the system.

## db2support - Problem analysis and environment collection tool

**Note:** The **db2support** special register options are ignored by the **db2caem** tool; **-compenv** should be used to specify the compilation environment for **db2caem** to use. The specified SQL statement is executed by the **db2caem** tool.

### **-B** | **-basic**

Restricts the collection to only optimizer information. No other information is collected except information for the db2supp\_opt.zip file. The **-basic** parameter must be used with the **-st**, **-sf**, or **-se** parameters or a syntax error is returned.

### **-c** | **-connect**

Specifies to connect to the specified database.

### **-cd** | **-curdegree**

Specifies the value of the current degree special register to use. The default is the value of the **dft\_degree** database configuration parameter.

### **-cfg**

Collect configuration information and exclude all other support-related data. This parameter can be combined with the only following parameters: **-c**, **-connect**, **-d**, **-database**, **-m**, **-html**, **-n**, **-number**, **-o**, **-output**, **-p**, **-password**, **-u**, **-user**, **-v**, **-verbose**.

**Note:** This command parameter is available in DB2 Version 9.7 Fix Pack 2 and later fix packs.

### **-cfs**

Specifies that additional diagnostic data for cluster file system is packaged into the generated .zip file. This parameter collects only additional cluster file system data that is space intensive or takes a long time to get collected.

### **-c1** | **-collect**

Specifies 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**

**Note:** If you specify an event monitor parameter (**-actevm**, **-appid**, **-uowid**, **-actid**) without **-st**, **-sf**, or **-se**, the effective collection level is 1 with only **db2caem** information collected (no **db2exfmt** collection).

### **-cm**

Specifies that additional diagnostic data for cluster manager is packaged into the generated .zip file. This parameter collects only additional cluster manager data that is space intensive or takes a long time to get collected.

### **-co**

Collect catalogs for all tables in the database. The default is to collect catalog information only for the tables used in a query that has a problem.

### **-compenv** *compilation-environment-file*

Specifies the name of the file containing the name of the compilation environment that is used when the **db2caem** command is executed. The compilation environment (**comp\_env\_desc**) is in BLOB data type and is specified through a file as an optional input. If the parameter is not provided, the default compilation environment is used when executing **db2caem**.

## db2support - Problem analysis and environment collection tool

### **-cs | -curschema**

Specifies 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.

### **-C | -compress**

Enables archive compression. By default, the archive data is compressed into a single file. Archive compression is available only in archive mode, so you must also specify the **-A** parameter; otherwise, a syntax error is returned.

#### **tar**

Specifies that the files are archived using the **tar** (tape archive) command. The **tar** parameter is supported on UNIX and Linux operating systems.

#### **tgz**

Specifies that files are archived using the **tar** command and compressed using the **gzip** command. The **tgz** parameter is supported on UNIX and Linux operating systems.

### **-d database\_name | -database database\_name**

Specifies the name of the database for which data is being collected.

**Note:** By default an attempt is made to connect to the specified database. To override this behavior, specify the **-noconnect** or **-nco** parameter.

### **-nco | -noconnect**

Specifies that no attempt to connect to the specified database is to be made.

### **-c | -connect**

Specifies that an attempt to connect to the specified database is to be made.

**Note:** This command parameter is included by default when a database is specified.

### **event-monitor-parameters**

The following parameters uniquely identify the SQL statement for which the activity event monitor data is collected. They must be specified together.

#### **-actevm activity-event-monitor-name**

Specifies the name of the existing activities event monitor whose activitystmt logical grouping contains the data to be collected.

#### **-appid application-id**

Specifies the application identifier (**appl\_id** monitor element) uniquely identifying the application that issued the activities to be collected.

#### **-uowid uow-id**

Specifies the unit of work ID (**uow\_id** monitor element) whose data is to be collected. The unit of work ID is unique only within a given application.

#### **-actid activity-id**

Specifies the activity ID (**activity\_id** monitor element) whose data is to be collected. The activity ID is unique only within a given unit of work.

### **-extenddb2batch**

Specifies that **db2batch** information for all the optimization levels specified with the **-o1** or **-optlevel** parameter are to be captured. At least one value for the **-o1** parameter and a **-c1** parameter value of 3 must be specified with the **-extenddb2batch** parameter or the **db2support** command returns a syntax error.

## db2support - Problem analysis and environment collection tool

### **-fodc**

Specifies that only the FODC directories and the **db2diag** log files are collected. If no directories are specified, the **db2support** command shows a list of all the FODC directories for you to choose from. The directories are listed in ascending chronological order, based on usage timestamps, making the most recently used directories most visible.

The **db2support** command can only collect the FODC directories on the physical database host from where the command was run. As such, the **-global** parameter does not work with **-fodc**.

You can specify the time interval (**-t** or **-time**) or history (**-H** or **-history**) parameters, but if a specified FODC directory is outside the specified timeframe, **db2support** returns an error.

You also cannot specify the archive (**-A** or **-archive**) or basic (**-B** or **-basic**) parameters when using **-fodc**.

*Trap | Panic | BadPage | Hang | IndexError | Perf | DBMarkedBad*  
Specifies the category of FODC directories to collect.

*list of FODC directories*

A comma-separated list of existing FODC directories.

### **-fodcpath** *fodc\_path\_name*

Specifies the full path name to an existing directory where the **db2support** command can search for FODC packages. The **db2support** command searches the following paths for collecting FODC packages:

- The diagnostic data directory as specified by the **diagpath** and **alt\_diagpath** database manager configuration parameters
- The instance level FODCPATH settings in the **DB2FODC** registry variable
- The FODCPATH settings for each member in that machine
- The **db2pdcfg** setting in memory
- The *fodc\_path\_name* specified using the **db2support -fodcpath** command

### **-F | -full**

Specifies that all **db2support** information and optimizer-specific information are to be captured with nothing excluded.

### **-fp | -funcpath**

Specifies the value of the function path special register to use 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.

### **-h | -help**

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

### **-H** *history\_period* | **-history** *history\_period*

Limits the data that is collected to a particular interval of time. The *history\_period* variable must be specified. The *history\_period* variable can be specified with a number and time type with an optional beginning time value separated by a colon. The available types are:

d = days  
h = hours  
m = minutes  
s = seconds

## db2support - Problem analysis and environment collection tool

The beginning of time value is specified in time stamp format. Time stamp format is *YYYY-MM-DD-hh.mm.ss.nnnnnn*, where *YYYY* specifies a year, *MM* for a month of a year (01 through 12), *DD* for a day of a month (01 through 31), *hh* for an hour of a day (00 through 23), *mm* for minute of an hour (00 through 59), *ss* for the seconds of a minute (00 through 59), *nnnnnn* for microseconds on UNIX operating systems or milliseconds on Windows operating systems. You can omit some or all of the fields that follow the year field. If fields are omitted, the default values are used. The default values are 1 for the month and day, and 0 for all other fields.

The number and time type can be positive or negative, which you specify with + or - signs. If only a number and time type are specified, the default is negative. If a number and time type are specified and a beginning time value is specified, the default is positive. For example, `-history 6d` collects data for the past six days. `-history 6d:2009` collects data for the first six days of 2009.

This parameter cannot be used with the `-time` or `-t` parameter.

### **-host**

Specifies the host or hosts on which the command is issued. If this parameter is not specified, the command is issued on all hosts by default.

### **all**

Specifies that the command is issued on all hosts. This setting is the default behavior of the **db2support** command.

### *hostname*

Specifies the host or hosts on which the command is issued. If this option is not specified, the command is issued on all hosts. If multiple hosts are specified, all host names must be valid for the command to complete.

If you specify the `-host` option in an environment that is not a DB2 pureScale environment or a partitioned database environment, **db2support** returns an error.

### **-il | -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.

### **-install**

Collects the diagnostic data that is required to troubleshoot a problem with the DB2 installation process or with the creation of an instance. The diagnostic data is stored in the `db2support.zip` file. Copy the `db2support.exe` file to your local system; this ensures that when you issue the **db2support** command the `db2support.zip` file is placed into whichever directory that you copied `db2support.exe` to. For the most complete collection of diagnostic data, issue the command with root authority. Also, another recommendation is to indicate an output path for the `db2support.zip` file by specifying the *output\_path* variable with the `-install` parameter.

### **-host hostname | -host hostname\_list**

Specifies the host or hosts where diagnostic data is collected. For data collection on remote hosts, an SSH connection is required. If you do not specify a host name, diagnostic data is collected on the local host.

To collect diagnostic data on multiple hosts specify the `-host` parameter followed by *hostname\_list*, where *hostname\_list* is a comma-separated list of hosts for which you want to collect diagnostic data.

## db2support - Problem analysis and environment collection tool

### **-instance** *instancename*

Specifies the instance name for which the diagnostic data is being collected. If you do not specify this parameter, diagnostic data by default is collected on the instance defined in the **DB2INSTANCE** environment variable. To collect diagnostic data on a particular instance, specify the **-instance** parameter followed by *instancename*, where *instancename* is the name of the instance for which you want to collect diagnostic data.

### **-l** | **-logs**

Specifies that active logs are to be captured.

### **-localhost**

Specifies that the command is issued on the local host. If this option is not specified, the command is issued on all hosts.

### **-m** | **-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.

### **-member** *member\_number* | *member\_range*

Specifies the member or members on which the command is issued. If this parameter is not specified, the command is issued on the current member. Multiple members can be specified as a comma-separated list of *member\_number* (member1, member2), or using *member\_range*, where *member\_range* is a range of members (member1-member3), or using any combination of the first two methods.

### **all**

Specifies that the command is issued on all members defined in `db2nodes.cfg`.

The **db2support** tool runs per host. If multiple members reside on one host, **db2support** runs only once on the host. If you specify the members on which the command is issued, the member numbers you provide are used only to determine the hosts on which **db2support** runs.

If you specify the **-member** option in an environment that is not a DB2 pureScale environment or a partitioned database environment, **db2support** returns an error.

### **-n** | **-number**

Specifies the problem management report (PMR) number or identifier for the current problem.

### **-nc** | **-nocatalog**

Specifies that catalog information is not to be collected. By default, catalog information is collected.

### **-nl** | **-nodb2look**

Specifies that **db2look** command information is not to be collected. By default, **db2support** command information is collected.

### **-o** *filename*

Specifies a new name for the compressed file that is generated after issuing the **db2support** command. You can specify a new file name or you can state an absolute or relative path. However, the path must exist and be accessible before specifying this parameter or else an error occurs.

If this parameter is not used, the default compressed file that is generated is `db2support.zip`.



## db2support - Problem analysis and environment collection tool

If this parameter is used with the *output path* parameter, then the path specified by *output path* is ignored and the path specified by the **-o** parameter is used.

### **-ol** *levels* | **-optlevel** *levels*

Specifies the value of the optimization level special register to use. The default is the value of the **dft\_queryopt** database configuration parameter. The optimization level value can be specified as a single value or multiple values separated by a comma.

If multiple values are specified, all optimization information is collected for the first value. For each additional optimization level value specified, the explain plans are collected and stored in a separate file along with the initial and end times of the collection for each level.

### **-op** | **-optprofile**

Specifies the value of the optimization profile special register to use. This value is needed only if there was an optimization profile in effect when the statement was bound. The default is "" (an empty string).

### **-ot** | **-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* | **-password** *password*

Specifies the password for the user ID.

### **-pc** | **-ps** | **-purecluster** | **-purescale**

Specifies that additional DB2 diagnostic data and additional diagnostic data for cluster file system, cluster manager, and uDAPL is to be collected. This parameter collects only additional diagnostic data that is space intensive or takes a longer time to get collected. Specifying this parameter is equivalent to specifying the **-cm**, **-cfs**, and **-udapl** parameters.

### **-pf** *filename* | **-profile** *filename*

Specifies an alternative profile file. An absolute path must be specified when using this parameter to get the exact location of this new profile file.

The default profile is the `db2support_collection.profile` file and is stored in the `sqllib/adm` folder. This profile file is used to collect additional information that is not included in the standard **db2support** command execution. The following are the possible templates for the profile file which can be composed of any combination and number of template 1 and template 2:

#### Template 1

```
<COLLECTION>  
<NAME>...</NAME>  
<CMD>...</CMD>  
<OUTFILE>...</OUTFILE>  
<TIMEOUT>...</TIMEOUT>  
</COLLECTION>
```

#### Template 2

```
<COLLECTION>  
<NAME>...</NAME>  
<FILE>...</FILE>  
<OUTFILE>...</OUTFILE>  
<TIMEOUT>...</TIMEOUT>  
</COLLECTION>
```

## db2support - Problem analysis and environment collection tool

Each collection item is described by a name, command or file, output file, and a timeout value. This information is used to collect the additional information. Each collection item must have either a CMD or a FILE value but never both. All blank lines are skipped when this file is being parsed. The following describes the values that are encased in each collection item:

### NAME

The name of the data that is being collected.

### CMD

The command used at the command line to gather the additional information. This is a mandatory value.

### FILE

The name of the file to collect. This is a mandatory value.

### OUTFILE

The name of the output file where the collected information is stored. This is a mandatory value.

### TIMEOUT

The amount of time in seconds that the command-line execution must not exceed. The default value is 180 seconds.

If the OUTFILE and CMD or FILE values are missing, a parse error occurs when **db2support** parses the profile file. If this occurs, **db2support** skips this collection. After the profile file is parsed, the collected information is stored in USERCOLLECTION/OUTFILE in the db2supp\_system.zip file.

### -preupgrade

Collects environment and configuration data before a critical upgrade or update such as upgrading an instance or updating to the next fix pack. This parameter helps with troubleshooting any problems that might occur after the upgrade or update. This parameter can be used with only the **-d**, **-o**, **-fodcpath**, **-nl**, **-member**, and **-host** parameters.

After the collection of data is completed, the results are compressed into a file named db2support\_preupgrade.zip.

### -q | -question\_response

Specifies that interactive problem analysis mode is to be used.

### -ra | -refreshage

Specifies the value of the refresh age special register. This value applies only if there are materialized query tables (MQTs) that reference tables in the statement. The default value of the CURRENT REFRESH AGE is zero.

### -ro | -reopt

Specifies to use EXPLAIN with the REOPT ONCE option when explaining the query. The default is to ignore the REOPT ONCE option.

### -s | -system\_detail

Specifies that detailed hardware and operating system information is to be gathered.

### -se *embedded SQL file* | -sqlembed *embedded SQL file*

Specifies the path of the embedded SQL file containing the SQL statement for which data is being collected.

### -sdir *shared directory path* | -S *shared directory path*

Specifies the shared directory that is used for temporary storage during **db2support** collection. If you do not specify this option, the default shared directory is used to store data temporarily. The default shared directory is *<db2*

## db2support - Problem analysis and environment collection tool

*instance shared directory*>/sqllib\_shared in DB2 pureScale environments and *path/sqllib* in partitioned database environments. The data that is temporarily stored is deleted when the execution of the **db2support** command is complete.

**Note:** Since *sqllib\_shared* is used as the default shared directory in DB2 pureScale environments, ensure that there is enough space in the *sqllib\_shared* directory to store data during the **db2support** collection. You can use the following formula to calculate the minimum disk space required for the *sqllib\_shared* directory:

$10\text{GB} + \text{number of member/CF host} * 2\text{GB}$

**-sf** *SQL file* | **-sqlfile** *SQL file*

Specifies the file path containing the SQL statement for which data is being collected.

**-st** *SQL statement* | **-sqlstmt** *SQL statement*

Specifies the SQL statement for which data is being collected.

**-t** *time\_interval* | **-time** *time\_interval*

Limits the data that is collected by **db2support** to a particular time interval specified by the *time\_interval* variable. The time interval can be specified as a start time, end time, or both in time stamp format separated by a colon. Time stamp format is *YYYY-MM-DD-hh.mm.ss.nnnnnn*, where *YYYY* specifies a year, *MM* for a month of a year (01 through 12), *DD* for a day of a month (01 through 31), *hh* for an hour of a day (00 through 23), *mm* for minute of an hour (00 through 59), *ss* for the seconds of a minute (00 through 59), *nnnnnn* for 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 fields are omitted, the default values are used. The default values are 1 for the month and day, and 0 for all other fields.

If only a start time is specified (**-t** 2009), the **db2support** command collects files that were modified after the start time. If only an end time is specified (**-t** :2009), the **db2support** command collects files that were modified before the end time. If both of inputs are specified (**-t** 2008:2009), the **db2support** command collects files that are modified within the interval of the start time and end time. There is no default value for this parameter. At least one of time stamps must be specified.

This parameter cannot be used with the **-history** or **-H** parameter.

**-td** | **-delimiter**

Specifies the statement termination character. This command parameter works in the same way as the **-td** parameter of the **db2** command. The default is a semicolon.

**-timeout** *N*

Specifies the timeout period in seconds after which the **db2support** tool stops its execution. The *N* variable must be specified in seconds. **timeout** specifies the total run time since the start of the execution and not the timeout for each specific collection. If a timeout occurs, the *db2support.zip* file is created and the error messages are written to the screen and to the *db2support.log* file.

This parameter can be used with all other parameters.

**-u** *userid* | **-user** *userid*

Specifies the user ID to use to connect to the database.

## db2support - Problem analysis and environment collection tool

### **-udapl**

Specifies that diagnostic data for uDAPL is packaged into the generated .zip file. This parameter collects only additional uDAPL data that is space intensive or takes a long time to get collected.

### **-unzip** *compressed\_file\_name*

Extracts the contents from the compressed file specified. An absolute or relative path must be specified for *compressed\_file\_name*. This parameter can extract the db2support.zip file on the system file where extraction utilities are not available. The **-unzip** parameter cannot be combined with parameters from other **db2support** modes. Also, the **db2support -unzip** command recognizes files specified in the *compressed\_file\_name* variable with only a .ZIP or .zip file extension.

If you specify the *output\_path* variable with the **-unzip** parameter, the extracted files are placed in the *output\_path* directory. If you do not specify the *output\_path* variable with the **-unzip** parameter, by default a new directory named *compressed\_file\_name* is created under the current directory and the extracted files are placed inside the *compressed\_file\_name* directory.

### **list**

Specifies that the contents of the compressed file are listed on standard output but not extracted. The file name, size, and date are shown. This parameter can be useful when db2support.zip is large and there is little space available on the system.

### **quiet**

Avoids the **db2support** command from prompting you for input regarding extracted files already existing in the *output\_path* or the current directory if *output\_path* was not issued. If the **quiet** parameter is not issued, you are prompted with a message asking you if you want the file specified to be overwritten or not. If this parameter is specified it overwrites all the existing files without prompts.

### **-v** | **-verbose**

Specifies that verbose output is to be used while this tool is running.

### **-x** | **-xml\_generate**

Specifies that an XML document containing the entire decision tree logic that is used during the interactive problem analysis mode (**-q** mode) is to be generated.

## Examples

### Example 1

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 - Problem analysis and environment collection tool

```
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 must be readable by an invoking user ID.

- 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.

### Example 2

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, optimizer information, and diagnostic files are collected as well.

### Example 3

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 MYPROFSCHEMA.MYPROFILE -ot ALL -il CS
```

### Example 4

To collect the same information shown in the previous example but with multiple user-specified values for the optimizer-related special registers and collect **db2batch** information for each optimizer special register value, use:

```
db2support . -d sample -st "SELECT * FROM EMPLOYEE" -cs db2usr -cd 3  
-ol 3,5,7 -cl 3 -extenddb2batch -ra ANY -fp MYSCHEMA -op MYPROFSCHEMA.MYPROFILE -ot ALL -il CS
```

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 important when correcting diagnostics.

### Example 5

To limit the data collection to files modified in the last three days before the current time, use:

```
db2support -H 3d
```

### Example 6

To limit the data collection to files modified in the first three days of 2009 (time period 2009-01-01-00.00.00.000000 through 2009-01-04-00.00.00.000000), use:

```
db2support -H 3d:2009
```

### Example 7

## db2support - Problem analysis and environment collection tool

To limit the data collection to files modified in time period 2008-01-01-00.00.00.000000 through the current time.

```
db2support -t 2008
```

### Example 8

To limit the data collection to files modified in the time period of 2009-01-01-00.00.00.000000 through 2009-03-01-00.00.00.000000, use:

```
db2support -t 2009-01:2009-03
```

### Example 9

The following section is an example profile file:

```
<COLLECTION>
<NAME>List</NAME>
<CMD>ls -la $HOME</CMD>
<OUTFILE>list.out</OUTFILE>
</COLLECTION>
```

With this profile file, **db2support** collects the information from the `ls -la $HOME` command and the results are stored in `USERCOLLECTION/list.out` in the `db2supp_system.zip` file. The timeout value is not specified because it is not mandatory. In this case, the default timeout value of 180 seconds is used.

### Example 10

To extract the contents from the `db2support_hostname1.zip` file:

```
db2support -unzip db2support_hostname1.zip
```

This command creates a directory named `db2support_hostname1` under the current directory, and the extracted files from the `db2support_hostname1.zip` file is placed in the `db2support_hostname1` directory.

To extract `db2support.zip` from the current directory and place it in the temp directory:

```
db2support temp -unzip db2support.zip
```

If some or all of the files that are being extracted exist in the destination directory, you are prompted to choose if you want to overwrite a particular file. If you want to avoid the prompts, issue the **quiet** parameter along with the **-unzip** parameter:

```
db2support temp -unzip quiet db2support.zip
```

### Example 11

The following are examples of the **-install** and **-host** parameters:

To create the `db2support.zip` file in the current directory:

```
db2support -install
```

To specify an output path `temp` for the `db2support.zip` file:

```
db2support temp -install
```

To specify a single host on which the diagnostic data is collected:

```
db2support -install -host myhost1
```

To specify multiple hosts on which the diagnostic data is collected:

```
db2support -install -host myhost1,myhost2
```

### Example 12

## db2support - Problem analysis and environment collection tool

To specify timeout for the total **db2support** collection.

```
db2support -d sample -timeout 3
```

### Example 13

To specify timeout for collection of hardware and operating system information.

```
db2support -d sample -c -s -timeout 15
```

### Example 14

To specify timeout for the optimizer **db2support** collection.

```
db2support -d sample -c -timeout 7 -st "select * from staff"
```

## Using db2support to collect activity event monitor data for the SQL statement:

For example:

- `db2support -d sample -st "select * from staff" -aem`

In addition to current collection at **-c1 2**, this command starts the **db2caem** command, which creates the activity event monitor and capture information of details, section, values, and actuals for the SQL statement "select \* from staff". The **db2support** command collects all the **db2caem** output.

- `db2support -d sample -sf badquery.sql -aem`

In addition to current collection at **-c1 2**, this command invokes **db2caem**, which creates the activity event monitor and capture information of details, section, values, and actuals for the SQL statement specified in the file `badquery.sql`. The **db2support** command collects all the **db2caem** output.

- `db2support -d sample -actevm mymon -appid *LOCAL.amytang.100203234904 -uowid 44 -actid 1`

In addition to current collection at **-c1 0**, this command starts the **db2caem** command, which captures the activity event monitor information of details, section, values, and actuals for the SQL statement identified by the event monitor options from the existing activity event monitor. The **db2caem** command does not create an activity event monitor in this case. The **db2support** command collects all the **db2caem** output.

## db2support examples for collections specific to DB2 pureScale environments

### Example 1

On the host run:

```
host:~$ db2support
```

- This command creates a `db2support.zip` file in the current folder. This file contains the DB2 diagnostic data and additional diagnostic data specific to DB2 pureScale components, such as cluster manager, cluster file system and uDAPL, that is collected from all hosts.
- The `db2support.zip` file contains a folder called `PURESCALE`. The `PURESCALE` folder has three additional subfolders called `CFS`, `CM`, and `UDAPL`, with the corresponding information files inside.
- You can specify additional parameters to this command to collect more diagnostic data.

### Example 2: Running db2support with -cm option

On the host run:

## db2support - Problem analysis and environment collection tool

```
host:~$ db2support -cm
```

- This command creates a `db2support.zip` file by default and also collects additional cluster manager data that is space intensive or takes a long time to get collected. The additional cluster manager data that is collected by the `-cm` option is stored in the `CM` folder that is located inside the `PURESCALE` folder of the `db2support.zip` file.

### Example 3: Running db2support with -purecluster or -purescale option

On the host run:

```
host:~$ db2support -purescale
```

- This command creates a `db2support.zip` file by default. This command also collects additional diagnostic data that is specific to DB2 pureScale components, such as cluster manager, cluster file system and uDAPL, and takes a long time to get collected or is space intensive. The additional diagnostic data is stored in the corresponding `PURESCALE`, `CFS`, `CM`, and `UDAPL` folders located in the `db2support.zip` file.
- This command collects diagnostic data that is similar to what is collected by the following command but this command also collects additional diagnostic data that is specific to DB2 pureScale environments:

```
host:~$ db2support -cm -cfs -udapl
```

## 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 parameters 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 by the **db2support** tool is from the machine where the tool runs. In a client-server environment, database-related information is 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 directories (`diagpath` and `alt_diagpath`) is from the local machine where the **db2support** tool is running. Data such as buffer pool information, database configuration, and table space information is 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 situation is also true for files that contain 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 to evaluate a particular bad or slow query.
- Stored procedures are not supported.

The **db2support** command does not collect explain data for dynamic SQL.

If an FODC package is stored in a directory path that is different from the default diagnostic path or not in a path specified by an `FODCPATH` setting, you need to indicate the FODC path to the **db2support** command with the `-fodcpath` parameter,



## **db2support - Problem analysis and environment collection tool**

so that the FODC package can be included in the `db2support.zip` file.

# db2swtch - Switch default DB2 copy and database client interface copy

Switches both the default DB2 copy and the default database client interface 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 Default DB2 and IBM Database Client Interface Selection wizard which you can follow to set a new default DB2 Copy and set the default database client interface copy. This command is only available on Windows operating systems.

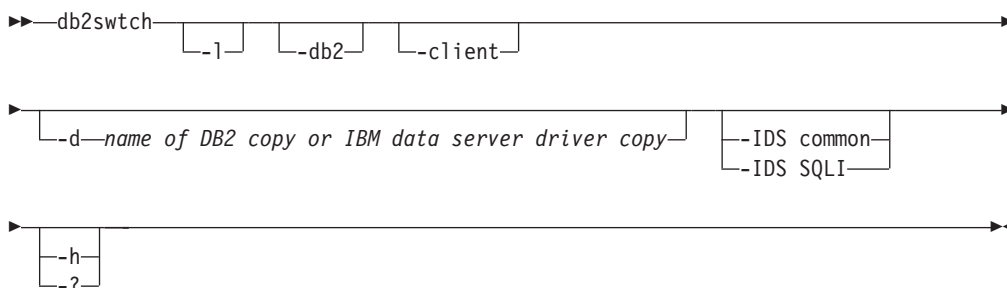
## Authorization

Local Administrator authority.

## Required connection

None

## Command syntax



## Command parameters

### no arguments

Launches the utility in graphical mode.

**-l** Displays a list of DB2 copies and IBM data server driver copies on the system.

**-db2 -d** *DB2\_copy\_name*  
Switch the default DB2 copy to the name specified.

`db2swtch -db2 -d DB2_copy_name`

**-client -d** *name of DB2 copy or IBM data server driver copy*  
Switch the default client interface copy to the name specified.

`db2swtch -client -d name of DB2 copy or IBM data server driver copy`

**-d** *DB2\_copy\_name*  
Switches both default DB2 copy and client interface copy to the name specified.

`db2swtch -d DB2_copy_name`

### -IDS

#### common

Redirects the IDS .NET data provider reference in `machine.config` to common IDS .NET data provider.

## **db2swtch - Switch default DB2 copy and database client interface copy**

**SQLI** Redirects the IDS .NET data provider reference in machine.config to SQLI IDS .NET data provider.

**-h | -?** Displays help information.

### 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.

---

## 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.

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 want 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 operating systems.

### Authorization

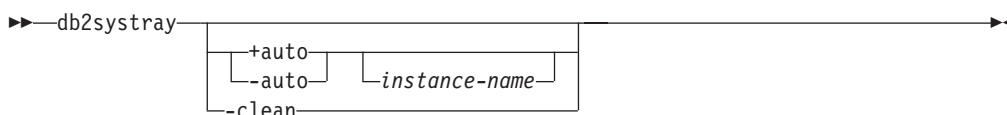
No special authority is required for starting **db2sysray**. Appropriate authority is required for taking actions.

### Required connection

None

## db2sysray - Start DB2 system tray

### 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 want to clean up.

## db2tapemgr - Manage log files on tape

Allows the storage and retrieval of DB2 log files to and from tape. The **db2tapemgr** command reads the **logarchmeth1** value and copies these log files from disk to the specified tape device, then it updates the recovery history file with the new location of the copied log files.

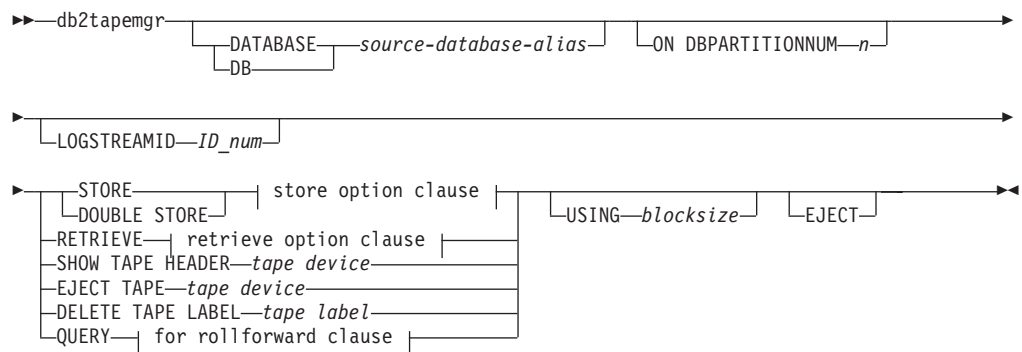
**Note:** This command is not supported in DB2 pureScale environments.

### Authorization

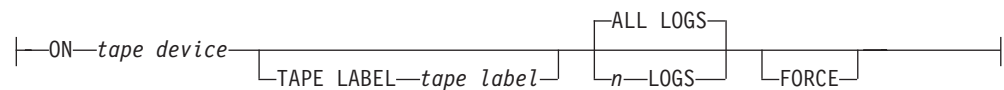
One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMANT

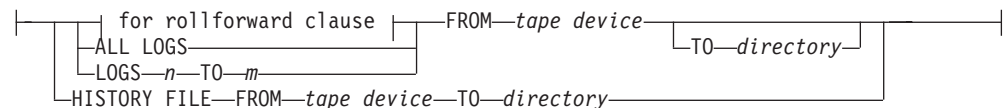
### Command syntax



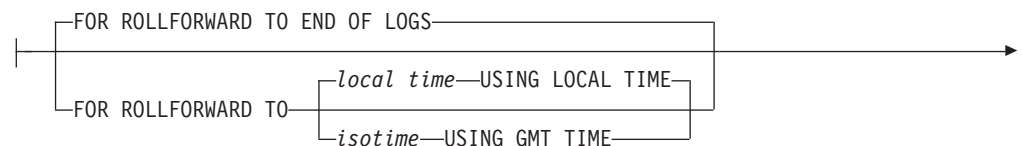
#### store option clause:



#### retrieve option clause:



#### for rollforward clause:



▶ `USING HISTORY FILE—history file`

### 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** *n*

Specifies the database partition number to work on. If no value is specified, **DB2NODE** is used.

#### **LOGSTREAMID** *ID\_num*

Specifies the log stream ID to work on. If no value is specified, all log streams will be 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.

**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.

## 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** or the **MON\_GET\_TABLESPACE** table function.

### 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
```

## db2tdbmgr - Migrate tools catalog database

The **db2tdbmgr** command migrates specific tools catalog database objects after running the **UPGRADE DATABASE** command on the tools catalog database.

### Authorization

*sysadm*

### Required connection

This command establishes a database connection.

### Command syntax

```

▶▶ db2tdbmgr -d db_name -s schema_name -u user_name -p password

```

### Command parameters

- d *db\_name*  
Tools catalog database name.
- s *schema\_name*  
Tools catalog schema name.
- u *user\_name*  
User name used to connect the tools catalog database.
- p *password*  
Password used to connect the tools catalog database.

### Examples

The following example migrates the tools catalog tables, under the database alias toolsdb and schema systools:

```
db2tdbmgr -d toolsdb -s systools -u db2inst1 -p *****
```

### Usage notes

This command will only migrate tools catalog tables to a newer version, and cannot be used to convert migrated tools catalog tables to its previous version.

The database must be cataloged before migration. Most of the time, a migration error message is self explanatory, clearly stating the location of the error. If the error message complains about any of the objects such as tables or column names, then the reported objects might be corrupted or missing under the database name submitted for migration.

## db2top - DB2 monitoring tool

The **db2top** command provides a unified, single-system view of a multi-partition database or single-partition database on the AIX, Linux, HP-UX, and Solaris operating systems. Quickly identifies global problems, or specific database partition problems in the system. By combining snapshot information from each database partition, a dynamic realtime view of a running DB2 system is provided.

### Scope

**db2top** can be run in interactive mode or in batch mode.

### Authorization

One of the following authorities:

- sysadm
- sysctrl
- sysmaint
- sysmon

### Required connection

Instance. A connection is not required in replay or client mode.

### Command syntax

```

▶▶ db2top -A
          -a
          -B
          -b suboption
          -C suboption
          -D delimiter
          -d database name
          -f file </HH:MM:SS><+offset>
          -h
          -i interval in seconds
          -k
          -L
          -m duration in minutes
          -n node name
          -o outfile
          -P member_number
          -p password
          -R
          -s number
          -u username
          -V schema
          -x
    
```

### Command parameters

**-A** Enables automatic performance analysis. The **db2top** command produces a top five performance report. Use this option in replay and background mode (**-b** parameter). Typical usage includes:

Running **db2top** in collection mode for a long period (for example, 4 hours):

```
db2top -f collect.file -C -m 240
```

Running **db2top** in replay mode with automatic performance analysis. Automatic performance analysis is available for any functions supported in background mode and is based on the default sort criteria specified in the .db2toprc configuration file for the selected suboption (automatic performance analysis is also available in background mode, even when not replaying). As an example, to analyze the most active sessions, issue the following command:

```
db2top -f collect.file -b l -A
```

Running **db2top** in replay mode, jumping to the required point in time to analyze further:

```
db2top -f collect.file /HH:MM:SS
```

**-a** Specifies that only active objects are displayed.

**-B** Displays active objects in bold (reverse) color. This is helpful when the screen does not support colors and colors are set to off in the .db2toprc configuration file.

**-b** *suboption*

Runs **db2top** in background mode. When using the **-b** parameter, the **db2top** command displays information in CSV format. **db2top** can be run in background mode in combination with reading snapshot data from a collection file using the **-f file** parameter. Issuing multiple sub-options for background mode (-b) is not supported. The **-b** parameter takes one of the following sub-option values:

- d** database
- l** sessions
- t** tablespaces
- b** bufferpools
- T** tables
- D** Dynamic SQL
- s** Statements
- U** Locks
- u** Utilities
- F** Federation
- m** Memory pools

The following parameters can only be used with the **-b** parameter: **-X**, **-L**, **-A**, **-s**, **-D**, **-o**.

**-C** Runs **db2top** in snapshot collector mode. Raw snapshot data is saved in <db2snap-<dbname>-<Machine><bits><.bin> by default (unless **-f** is specified). Pipe can also be specified for output instead of file. Specifying multiple sub-options for collector mode (-C) are supported. To include locks information in the collection file, use **-x** along with **-C**. The **-C** parameter takes one of the following sub-option values:

- **b** : buffer pools
- **D** : Dynamic SQL

## db2top - DB2 monitoring tool

- d : database
- F : Federation
- l : sessions
- s : Statements
- T : Tables
- t : tablespaces
- U : Locks

The **db2top** command runs in replay mode if the **-C** command parameter is not specified.

### **-D** *delimiter*

Specifies the field delimiter (single character). Use in background mode (**-b** parameter).

### **-d** *database name*

Specifies the database to monitor.

### **-f** *file* </HH:MM:SS><+offset>

- If the **-f** parameter is not specified, **db2top** is run in replay mode. If the **-f** parameter is specified, it indicates output filename. The database name specified with the **-d** parameter does not need to exist. It is only referenced if you want to issue explains or Dump DB Struct.

In replay mode (**-C** parameter is not used), if snapshot data has been previously collected in *file*, offset jumps to a certain point in time in the file. It can be expressed in seconds (+10s), minutes (+10m) or hours (+10h). /HH:MM:SS skips entries until the specified point in time.

- In the collector mode (**-C** parameter is used), you can specify the output filename. </HH:MM:SS><+offset> will be ignored if you are in collector mode

### **-h** Displays usage information for the **db2top** command.

### **-i** *interval in seconds*

Specifies the delay between screen updates. The delay time cannot be less than one second.

### **-k** Displays actual or delta values. For all functions in delta mode, delta values are computed and displayed on a per second basis regardless of the specified refresh interval.

### **-L**

Specifies that statements that are currently executing or most recent statements from current sessions are captured at each reporting interval specified by the **-i** parameter. If an SQL statement starts and finishes between an interval, it will not be included.

In the background mode, use **-b 1 -L** parameters to catch SQL statements executed in a session and output them to the ALL.sql file in the current working directory.

Currently executing or the most recent SQL statement for a connected session are captured at each interval (as specified by **-i** parameter). If a SQL statement starts and finishes between an interval, it will not be caught.

### **-m** *duration in minutes*

Limits the duration of **db2top** in minutes for **-b** and **-C** parameters.

- n *node name*  
Specifies the node to attach to.
- o Specifies the output file name. Used in background mode (-b parameter).
- P *dbpartition number*  
Specifies the database partition number on which the snapshot will be issued. If the -P parameter is specified and the *dbpartition number* is not specified, **db2top** attaches to the default database partition.
- p Specifies the password used to access the database. If omitted and the -u parameter has been specified, the user is prompted for the password
- R Resets snapshot at startup.
- s *number*  
Specifies how many samples will be displayed. Only supported in background mode (-b parameter).
- u Specifies the username used to access the database.
- V *schema*  
Specifies the default schema used in explains.
- x Specifies whether to display additional counters on session and application screens (might run slower on session).
- X Specifies format of output is XML. Use in background mode in conjunction with the -b parameter.

## Snapshot data collector

The **db2top** monitoring utility can be run in replay mode, which means it can run against a saved copy of the raw binary snapshot data. To run in replay mode, **db2top** must be first run in data collector mode, either in batch mode by running **db2top** from the command line with the -C parameter, or by activating or deactivating data collection from an interactive session by pressing C. This will create a file <db2snap-*hostname*.bin> in the current directory. Then, the **db2top** utility can be run against <db2snap-<dbname>-<Machine><bits>.bin> using the -f arguments. The **db2top** monitoring utility does not need to attach to the DB2 instance in replay mode, which is convenient for remote monitoring. It is possible to limit the content and size of the stream file by specifying any of the sub-options available to the -C parameter.

## Examples (batch mode)

The command parameters are as below:

```
-b l          --> Run in background mode, while catching the "sessions" related
              info in db2top
-C           --> run db2top in snapshot collector mode
-d CUST      --> database
-i 3         --> 3 second interval
-m 60        --> limit duration of db2top to 60 minutes
-n node      --> node
-o db2top.xml --> output file db2top.xml
-p password  --> password
-s 10000     --> collect 10000 samples
-u userid    --> userid
-V sv9       --> Schema
-x           --> display additional counters on session
```

## db2top - DB2 monitoring tool

The following example commands monitor a database called "CUST" on a node called "node" with a schema called "sv9". The user ID for the database is "userid", and the password is "password":

- The following example command, monitors the database in background mode in three second intervals for sixty minutes and outputs session information in XML format into a file called db2top.xml:

```
db2top -d CUST -n node -u userid -p password -V sv9 -i 3 -b 1 -X -o db2top.xml  
-m 60
```

- The following example command, monitors the database in background and snapshot collection mode, collecting 10000 samples of session information in XML format. The output binary file is db2snap-CUST-AIX64.bin:

```
db2top -d CUST -n node -u userid -p password -V sv9 -b 1 -x -s 10000 -X  
-f db2snap-CUST-AIX64.bin
```

- The following example command replays the output capture in the previous example command:

```
db2top -d CUST -f db2snap-CUST-AIX64.bin
```

For more information, refer to the DB2 problem determination using db2top utility at <http://www.ibm.com/developerworks/data/library/techarticle/dm-0812wang/>



## db2trc - Trace

**db2trc** controls the trace facility of a DB2 instance or the DB2 Administration Server (DAS). The trace facility records information about operations and formats this information into readable form.

Enabling the trace facility (OFF by default) might impact your system's performance. As a result, only use the trace facility when directed by a DB2 technical support representative; otherwise, turn off the trace once enough information has been recorded.

DB2 traces can be especially useful when analyzing recurring and reproducible problems, which greatly facilitates the support representative's job of problem determination.

When using DB2 database systems, you might on occasion encounter an error message that directs you to "get a trace and call IBM Software Support", "turn on trace and examine the trace record", or to "contact your technical support representative with the following information: problem description, SQLCODE, SQLCA contents (if possible), and trace file (if possible)". Or, when you report a problem to IBM Software Support, you might be asked to perform a trace to capture detailed information about your environment.

### Authorization

To trace a DB2 instance on a UNIX operating system, you must possess one of the following authorizations:

- SYSADM
- SYSCTRL
- SYSMANT

To trace the DB2 Administration Server on a UNIX operating system:

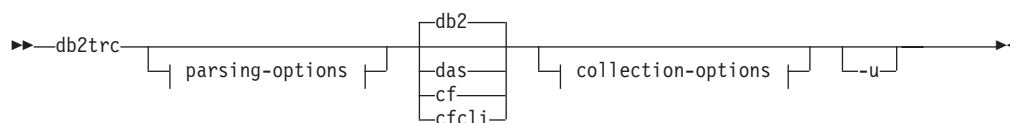
- DASADM

On a Windows operating system, no authorization is required.

### Required connection

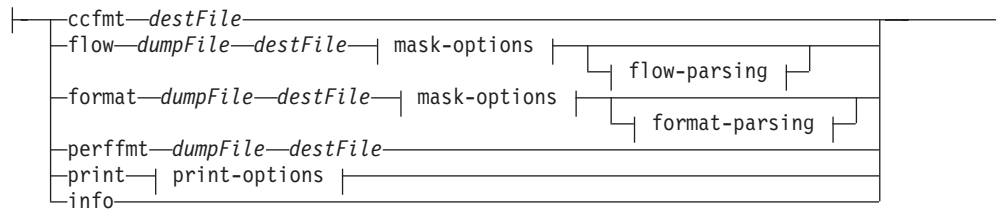
None

### Command syntax

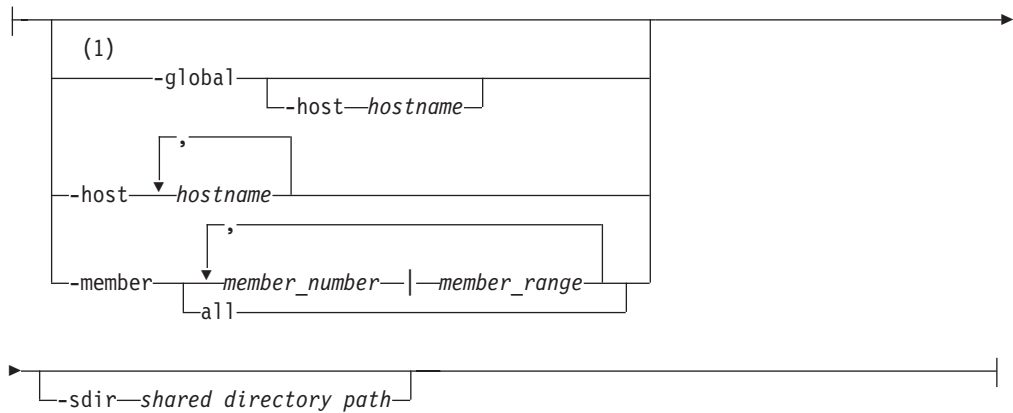


#### parsing-options:

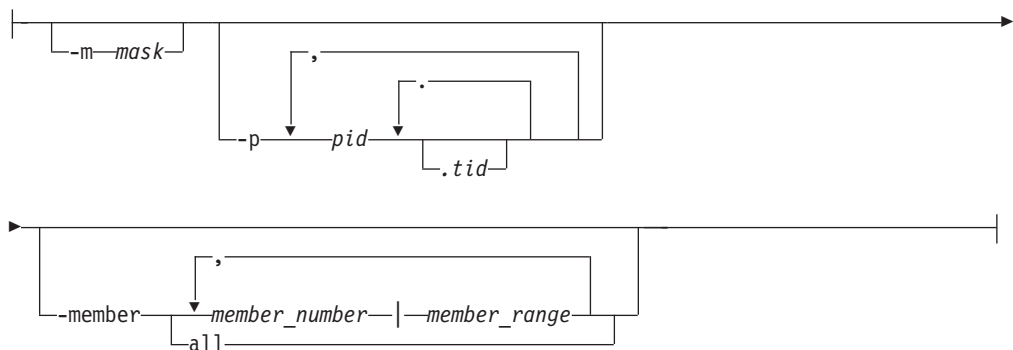
## db2trc - Trace



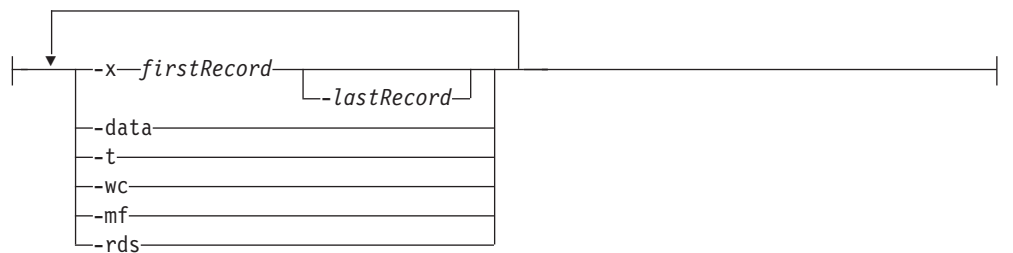
### location-options:



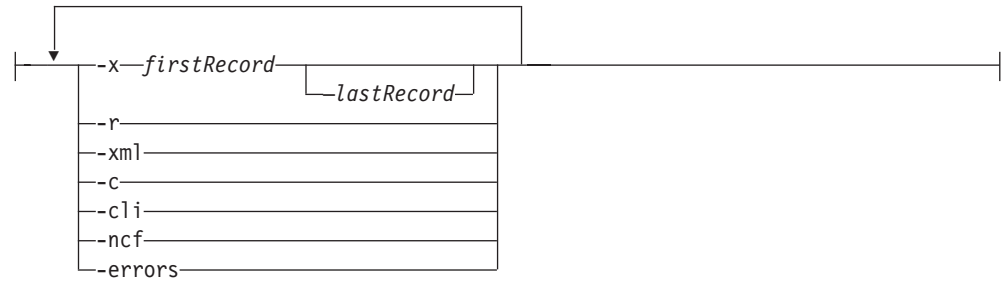
### mask-options:



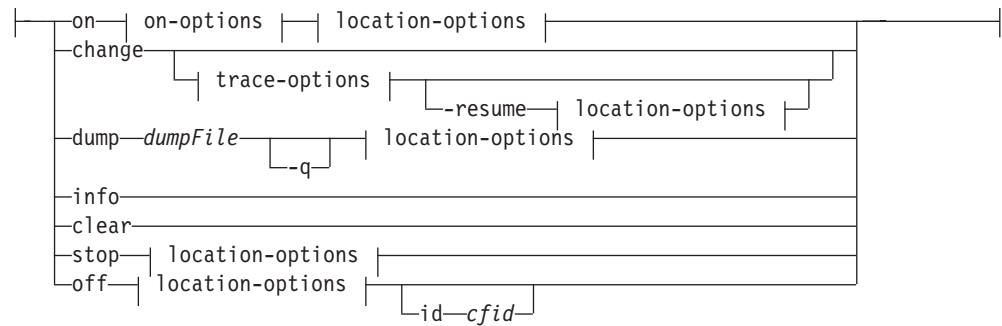
### flow-parsing:



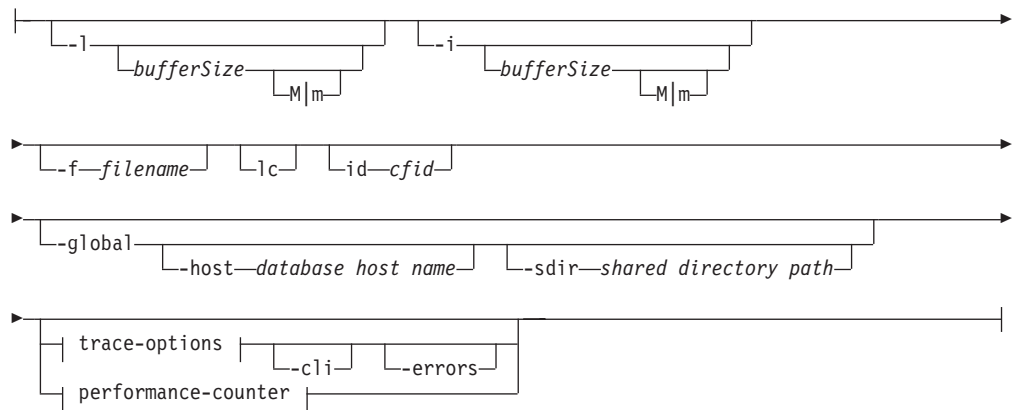
### format-parsing:



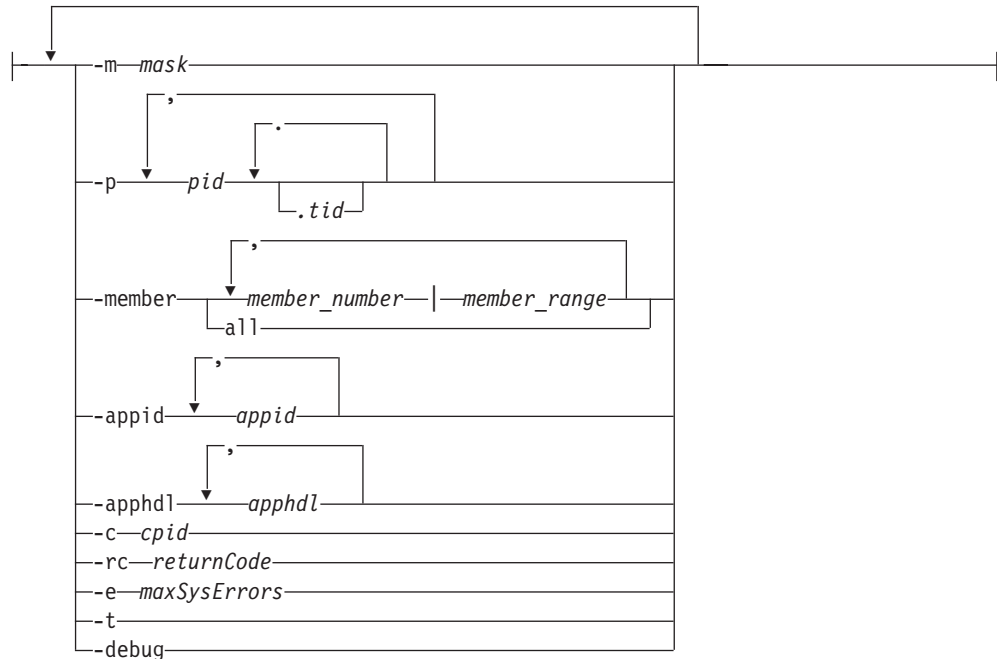
**collection-options:**



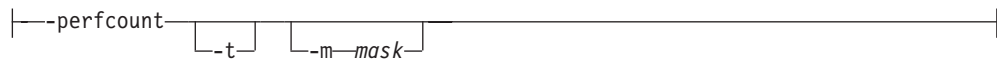
**on-options:**



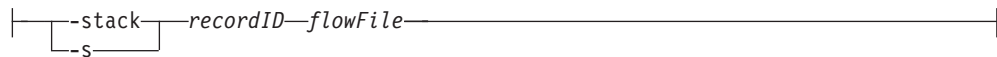
**trace-options:**



**performance-counter:**



**print-options:**



**Notes:**

- 1 The **-global** parameter has been deprecated. You can use the **-member all** parameter options to obtain information globally.

**Command parameters**

**-appid** Specifies which application IDs to trace. The **-appid** option works with the **on** and **change** options. **-appid** will not work with the **-perfcount** option.

**-apphdl**

Specifies which application handles to trace. The **-apphdl** option works with the **on** and **change** options. **-apphdl** will not work with the **-perfcount** option.

**-member member\_number | member\_range**

Specifies which database members (or partitions) to trace. If this option is not specified, the command is executed for all logical members on the host where the **db2trc** command is issued. Multiple members can be specified as a comma separated list of *member\_number* (member1, member2), or using *member\_range*, where *member\_range* is a range of members (member1-member3), or using any combination of the first two methods.

**all** Specifies that the command is issued on all members defined in `db2nodes.cfg`.

The **-member** option works with the **on**, **change**, **format** (both **flow** and **format** options), **stop**, and **off** options. See the following examples 2 and 3 for further details. The **-member** option does not work with the **-perfcount** option.

**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 instance.

**cf** Specifies that all trace operations will be performed on the CF server. The **-change**, **-clear** and **-stop** options does not work with this option.

**cfcli** Specifies that all trace operations will be performed on the members.

**Note:** To enable CF client trace, the environment variable **CA\_TRACE\_KEY\_FILE** must be set to an existing file. This file is used by the CF client library to create a SHARED MEMORY set. This set is used collect the trace data. Additionally the permission and owner information for this file will be used to set the permission and ownership of the created SHARED MEMORY. Any process that starts with this environment variable set to the same value will use the same SHARED MEMORY segment. The default name used by DB2 is `${INSTHOME}/sql1lib/tmp/.CF_DB2TRACE.key`. This environment variable is set automatically in the `db2profile` and `db2cshrc` files. However, if these are not being sourced then you will need to manually set this value.

**Note:** The CF client trace buffer cannot exceed 10 MB.

**on** Use this parameter to start the trace facility. See the following *Shared trace-options* section for a list of parameters.

**-l** [*bufferSize*]

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). The buffer size can be specified in either bytes or megabytes. To specify the buffer size in megabytes, add the character **M** | **m** to the buffer size. For example, to start **db2trc** with a 4 MB 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.

**-i** [*bufferSize*]

This option specifies the size and behavior of the trace buffer. **-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** | **m** to the buffer size.

**-f** *filename*

When tracing to a file, a fully-qualified file name must be specified, and if **-l** or **-i** is used with **-f** option, their *bufferSize* values will limit the size of the file on disk. **-l** will preserve the last trace

records and will be allowed to wrap within the file. **-i** will preserve the initial trace records and will stop tracing when the file size limit is reached. To specify the file size in megabytes, add the character M | m, and for gigabytes, add the character G | g after the value specified for **-i** and/or **-l** *buffersize*.

**-cli** Trace the CLI/ODBC driver and CLI applications.

### **-errors**

Trace only errors and nonzero return codes on function exit. This option cannot be specified at the same time with the **-debug** or **-perfcount** options.

**-lc** List all of the valid CF trace components and trace levels for either the CF server or CF client.

### **-id** [*cf-id*]

The CF server id as specified in the `db2nodes.cfg` file. By default the operation will be performed against all CF servers. However, if this option has been specified then only the specified CF server will be affected.

**change** This collection option lets you change the trace options that are in effect. See the following *Shared trace-options* section for a list of parameters.

### **-resume**

This option lets you resume execution of a suspended process. You cannot resume if **-suspend** was not enabled.

## Shared trace-options

Common trace options shared between **on** and **change**.

### **-m** *mask*

Reduces the amount of data collected or formatted. The trace mask has the following format:

*types.products.components.functions.categories*

Values for the mask would be provided by IBM Support.

The mask consists of five parts (trace record types, products, components, functions, and function categories). Each part can consist of comma separated lists, hyphen separated ranges, or single entries. An asterisk (\*) can be used to match anything. Field values may be specified by their names or corresponding numbers. Short form of mask specifying names of either *products*, *components* or *functions* parts of the full format may be used. Setting the mask to `"*.*.*.*.*"` is equivalent to not specifying a mask.

Example: `-m "entry,exit.*.SQL0,SQLLE.*.*"`

For DB2 pureScale environments, the mask has been overloaded so that tracing of the CF server and/or CF client can be enabled. The mask is used somewhat differently when dealing with the CF. The values for the mask can be found by using the **-lc** option. For the CF server and CF client the trace mask pieces will have the following meaning:

- *types* will always be \*
- *products* will be either CF or CFCLI (case does not matter)
- *components* will be any combination of one or more CF server or CF client components as can be listed using the **-lc** flag

- *functions* will always be \* because the CF does not support this feature
- *categories* will be any combination of one or more CF server or CF client trace level as can be listed using the **-lc** flag

Example: `-m "*.CF.svr_list.*.CF_TRACE_ERROR,CF_TRACE_ALL"`

#### **-p** *pid* [*.tid*]

Enables the trace facility only for the specified process IDs (*pid*) and thread IDs (*tid*). The period (.) must be included if a *tid* is specified. You can specify multiple thread ID's for a process ID. A period separates multiple *tid*'s for a *pid*. A comma separates each *pid tid* pair. A maximum number 64 *tid*'s can be paired with a *pid*.

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
```

To enable tracing for process 77 with threads 1, 2, 3, and 4, and for process 88 with threads 5, 6, 7, and 8 the syntax is:

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

**-c** *cpid* Trace or format only this companion process.

#### **-rc** *returnCode*

Treat *returnCode* as a system error. *returnCode* must be specified as a signed integer.

#### **-e** *maxSysErrors*

Stop trace after *maxSysErrors* system errors occurred.

**-t** Include timestamps.

#### **-debug**

This is an internal option used for debugging purposes by IBM Support. Usage is not recommended.

**info** The following is an example of environment information listed with this parameter:

```
D:\Program Files\IBM\SQLLIB\BIN>db2trc info
Marker                : @TRACE@
Trace version         : 7.0
Platform              : NT
Build level           : s060629
maxBufferSize         : 2097152 bytes (2 MB)
auxBufferSize         : 6291456 bytes (6 MB)
allocationCount       : 1
DB2TRCD pid           : 2384
DB2TRCD64 pid         : 0
Trace destination     : <shared memory buffer>
debug                 : disabled
debug runtime passno  : 0
numSuspended          : 0
Trace starting time   : 2011-03-25-15.03.58.909713-240

Buffer size           : 2097152 bytes (2 MB)
Allow buffer to wrap  : yes
Mask                  : *.*.*.*.*
Timestamps            : enabled
```

## db2trc - Trace

```
PID.TID mask      : all
Fixed data mask #1 : all
Fixed data mask #2 : all
Max system errors : infinite
Treat this rc as sys err: none
Member mask      : none
Application ID mask : none
Application Handle mask : none
```

### **dump** *dumpFile*

Dumps the binary format trace information, stored in the buffer, to a file. The following command will put the information in the current directory in a file called db2trc.dmp:

```
db2trc dump db2trc.dmp
```

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

**-q** Quiet mode.

### **ccfmt** *destFile*

Dump and format a code coverage trace. Specify a destination file name for the dump.

### **flow** *dumpFile destFile*

After the trace is dumped to a binary file, format it into a readable text file. Use the **flow** option to format records sorted by process or thread. Specify the name of the dump file and the name of the destination file that will be generated. For example:

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

**-x** *firstRecord* [-*lastRecord*]

Only show record numbers *firstRecord* to *lastRecord*.

**-data** Include any trace record data in the flow.

**-t** Include timestamps (in sec:nsec format), if available.

**-wc** Include wall-clock timestamps, if available. To make use of this option, you must turn trace ON by also specifying the **-t** option which includes the capture of timestamps. For a usage example, see Example 1.

**-mf** Generate a separate destination file for each distinct flow.

**-rds** Include RDS operators information, if available.

### **format** *dumpFile destFile*

After the trace is dumped to a binary file, format it into a readable text file. Use the **format** option to format records chronologically.

**-x** *firstRecord* [-*lastRecord*]

Only show record numbers *firstRecord* to *lastRecord*.

**-r** Output in reverse order.

**-xml** Output data in xml parsable format.

**-c** Format communications buffers.

**-cli** Format the CLI/ODBC driver and CLI application buffers.

**-ncf** Do not use component custom formatting.

**-errors**

Trace only errors and nonzero return codes on function exit.



**-global**

Specifies that **db2trc** will also be run on remote hosts. This option is deprecated in DB2 Version 9.7 Fix Pack 4 and later fix packs.

**-host** *hostname*

Specifies that **db2trc** will be run only on the specified host or hosts. If this parameter is not specified, the command is issued on the local host. If multiple hosts are specified, all host names must be valid for the command to complete.

**-sdir** *shared directory path*

Specifies the shared directory **db2trc** will use to save files applicable to the **-f** and **dump** parameters.

**perffmt** *dumpFile destFile*

The performance trace formatter is a parsing option that formats a dump file containing performance counter data into readable text

**print -stack** *recordID flowFile*

Prints a backtrace of all stack frames for the specified record ID from the specified flow file. See Example 5 for further details.

**clear** Clears the contents of the trace buffer, particularly just before connecting to a specific database. This option can be used to reduce the amount of collected information by clearing the buffers of accumulated useless information before a connection to the wanted database is established.

**stop** This collection option stops tracing on demand; all processes suspend tracing, but the contents of the trace buffer are preserved so that they can be dumped later. This action is in contrast to the **off** option, which disables the trace facility altogether.

**off** Disables the trace facility. After the trace is dumped to a file, disable the trace facility by typing:  
db2trc off

**-id** [*cf-id*]

The CF server id as specified in the `db2nodes.cfg` file. By default the operation will be performed against all CF servers. However, if this option has been specified then only the specified CF server will be affected.

**-u** Provides additional information about most of the command line options. General form of the command line entry is shown in the following *Usage notes* section. Here is an example to obtain more information about the dump command for the DAS instance:  
db2trc das dump -u

## Examples

### Example 1

To capture a trace with wall-clock timestamps included, you must specify the **-t** option when you turn trace ON and you must specify **-t -wc** options with the **flow** option to output the wall-clock timestamps into the readable text file. The following is an example of the steps you can perform:

1. Turn trace ON, capture timestamp information, and specify the dump file name by executing the following command:  
db2trc on -t -f db2trc.dmp
2. After the trace period has ended, turn trace OFF by executing the following command:

db2trc off

- To format the binary dump (.dmp) file into a readable text file (for example, db2trc.flw), execute the following command:

```
db2trc flow -t -wc db2trc.dmp db2trc.flw
```

The following is an example of the output displayed when the formatting has been completed:

```
Total number of trace records      : 3349
Trace truncated                    : NO
Trace wrapped                      : NO
Number of trace records formatted  : 43 (pid: 5414 tid 182967198368 node: 0)
Number of trace records formatted  : 2690 (pid: 29615 tid 182960067008 node: 0)
Number of trace records formatted  : 118 (pid: 5394 tid 183102335328 node: 0)
Number of trace records formatted  : 498 (pid: 29616 tid 182965078816 node: -1)
```

- To view the contents of the readable text file db2trc.flw, execute the following command:

```
more db2trc.flw
```

The following is an example of the output displayed when reading the contents of the readable text file containing wall-clock timestamp information:

```
pid = 5414 tid = 182967198368 node = 0

1      0.000000000  clp_bp_con data [probe 21] 2009-06-16-11.02.32.38407400
2      0.000038000  | sqlossig entry   2009-06-16-11.02.32.38411200
3      0.000050000  | sqlossig exit   2009-06-16-11.02.32.38412400
4      0.000057000  | sqlorque2 entry 2009-06-16-11.02.32.38413100
5      0.000062000  | sqlogmbkEx entry 2009-06-16-11.02.32.38413600
6      0.000070000  | | sqloGetPrivatePoolHandle entry 2009-06-16-11.02.32.38414400
7      0.000077000  | | sqloGetPrivatePoolHandle exit 2009-06-16-11.02.32.38415100
8      0.000088000  | sqlogmbkEx mbt [Marker:PD_OSS_ALLOCATED_MEMORY ] 2009-06-16-11.02.32.38416200
9      0.000092000  | sqlogmbkEx exit 2009-06-16-11.02.32.38416600
10     0.000094000  | sqlorqueInternal entry 2009-06-16-11.02.32.38416800
11     0.000096000  | | sqloSetAlarmApp entry 2009-06-16-11.02.32.38417000
12     0.000099000  | | | sqloSigMask entry 2009-06-16-11.02.32.38417300
13     0.000101000  | | | sqloSigMask exit 2009-06-16-11.02.32.38417500
14     0.000103000  | | | sqloSigMask entry 2009-06-16-11.02.32.38417700
15     0.000105000  | | | sqloSigMask exit 2009-06-16-11.02.32.38417900
16     0.000108000  | | | sqloSetAlarmApp exit 2009-06-16-11.02.32.38418200
2825   5.000561000  | | | sqloClearAlarmApp entry 2009-06-16-11.02.37.38463500
2826   5.000576000  | | | | sqloSigMask entry 2009-06-16-11.02.37.38465000
2827   5.000579000  | | | | sqloSigMask exit 2009-06-16-11.02.37.38465300
2828   5.000582000  | | | | sqloSigMask entry 2009-06-16-11.02.37.38465600
2829   5.000585000  | | | | sqloSigMask exit 2009-06-16-11.02.37.38465900
2830   5.000587000  | | | | sqloClearAlarmApp exit 2009-06-16-11.02.37.38466100
2831   5.000589000  | | | sqlorqueInternal exit [rc = 0x870F00B9 = -2029059911 = SQLQ_SEM_TIMEOUT] 2009-06-16-11.02.37.38466300
2832   5.000592000  | | | sqlofmbkEx entry 2009-06-16-11.02.37.38466600
2833   5.000597000  | | | sqlofmbkEx mbt [Marker:PD_OSS_FREED_MEMORY ] 2009-06-16-11.02.37.38467100
2834   5.000599000  | | | sqlofmbkEx exit 2009-06-16-11.02.37.38467300
2835   5.000601000  | | | sqlorque2 exit [rc = 0x870F00B9 = -2029059911 = SQLQ_SEM_TIMEOUT] 2009-06-16-11.02.37.38467500
2836   5.000614000  | | | clp_bp_con data [probe 21] 2009-06-16-11.02.37.38468800
2837   5.000617000  | | | sqlossig entry 2009-06-16-11.02.37.38469100
2838   5.000620000  | | | sqlossig exit 2009-06-16-11.02.37.38469400
2839   5.000623000  | | | sqlorque2 entry 2009-06-16-11.02.37.38469700
2840   5.000626000  | | | | sqlogmbkEx entry 2009-06-16-11.02.37.38470000
2841   5.000628000  | | | | | sqloGetPrivatePoolHandle entry 2009-06-16-11.02.37.38470200
2842   5.000631000  | | | | | sqloGetPrivatePoolHandle exit 2009-06-16-11.02.37.38470500
2843   5.000636000  | | | | | sqlogmbkEx mbt [Marker:PD_OSS_ALLOCATED_MEMORY ] 2009-06-16-11.02.37.38471000
2844   5.000638000  | | | | | sqlogmbkEx exit 2009-06-16-11.02.37.38471200
2845   5.000640000  | | | | | sqlorqueInternal entry 2009-06-16-11.02.37.38471400
2846   5.000643000  | | | | | | sqloSetAlarmApp entry 2009-06-16-11.02.37.38471700
2847   5.000646000  | | | | | | | sqloSigMask entry 2009-06-16-11.02.37.38472000
2848   5.000647000  | | | | | | | sqloSigMask exit 2009-06-16-11.02.37.38472100
2849   5.000649000  | | | | | | | sqloSigMask entry 2009-06-16-11.02.37.38472300
2850   5.000651000  | | | | | | | sqloSigMask exit 2009-06-16-11.02.37.38472500
2851   5.000654000  | | | | | | | sqloSetAlarmApp exit 2009-06-16-11.02.37.38472800
```

## Example 2

The following are examples for the use of the -member trace mask

- db2trc on -member n1[,n2,n3,n64]

This command allows specification of which database members or partitions to trace.

**Note:** Must be an integer number. If multiple members are specified, they must be separated by a comma. You can specify up to 64 members.

- `db2trc chg -member n1[,n2,n3,n64]`  
This command allows you to change the member mask to the specified database members or partitions.
- `db2trc stop -member n1[,n2,n3,n64]`  
This command allows you to remove members from the member mask. When the last member is removed from the member mask, trace is fully stopped having the same effect as running **db2trc stop**
  - The member number issued by this command must exist in the current member mask. Running **db2trc info** will display the current member mask ).
  - If you run the **db2trc on** command without the **-member** option, there will be no members in the member mask. This means that all members are traced
  - When all members (defined by `db2nodes.cfg` file ) in the current host are specified in this command, it will have the same effect as running **db2trc stop** , in this case when running **db2trc stop** without the **-member** option, trace is fully stopped for all members on that host.
- `db2trc off -member n1[,n2,n3,n64]`  
This command allows removal of members from the member mask. When the last member is removed from the member mask, trace is turned off, this has the same effect as running **db2trc off**.
  - The member number issued by this command must exist in the current member mask. Run **db2trc info** to display the current member mask.
  - If you run the **db2trc on** command without the **-member** option, there will be no members in the member mask. This means that all members are traced
  - When all the members (defined by `db2nodes.cfg` file) in current host are specified in this command, it will have same effect as **db2trc off**. When running **db2trc off** without the **-member** option then trace is turned off on that host
- `db2trc flw -member n1[,n2,n3,n64]` and `db2trc fmt -member n1[,n2,n3,n64]`  
Running this command will allows you to specify which members to include in the formatted trace

### Example 3

The following are examples for the use of the **-member** trace mask with the `db2nodes.cfg` defined as the following data:

```
0 host1 0
1 host1 1
2 host1 2
3 host2 0
4 host3 0
```

- `db2trc on[ chg] -member 1,2,3`  
when **-member** is run the member number will be mapped to the related host name, specified by **-member** and then run on the related host by means of the **rah** (or the **db2\_all**) command . For this example **db2trc -member 1,2** will run on host1 and **db2trc -member 3** will run on host2.
- `db2trc on -host host1,host2`  
**db2trc on** will run on both host1 and host2
- `db2trc on -member all`

**db2trc on** will run on host1, host2, and host3

#### Example 4

The following are examples for the use of the **-appid** and **-apphdl**

- **db2trc on -appid appid1,appid2**

This command will turn on trace for specific application IDs. This supports up to 12 application IDs.

- **db2trc chg -appid appid1,appid2**

This command will change (or reset) the specific application ID. To reset, run command **db2trc chg -appid none**. It will remove all application IDs in the mask.

- **db2trc on -apphdl apphdl1,apphdl2,apphdl3**

This command will turn on trace for specific application handle. This supports up to 16 application handles.

- **db2trc chg -apphdl apphdl1,apphdl2,apphdl3**

This command will change (or reset) the specific application handle. To reset, run command **db2trc chg -apphdl none**. It will remove all application handles in the mask

#### Example 5

The following example is a sample flow file `test.flw`:

```
pid = 1608 tid = 47604608002368 node = 0
1      sqlossig entry [eduid 1 eduname db2sysc]
2      sqlossig exit [rc = 0x840F0001 = -2079391743 = SQL0_ACCD]
3      sqkfFastCommManager::ResourceSelfTuning entry [eduid 1 eduname db2sysc]
4      | sqlogmt entry [eduid 1 eduname db2sysc]
5      | sqlogmt exit
6      sqkfFastCommManager::ResourceSelfTuning exit
7      sqkfFastCommManager::CollectResourceUsageStats entry [eduid 1 eduname db2sysc]
8      | sqlogmt entry [eduid 1 eduname db2sysc]
9      | sqlogmt exit
10     sqkfFastCommMandager::CollectResourceUsageStats exit
11     sqkfFastCommManager::UpdateMemoryConsumptionStats entry [eduid 1 eduname db2sysc]
12     sqkfFastCommManager::UpdateMemoryConsumptionStats exit
13     sqleSyscUpdateDynamicVars entry [eduid 1 eduname db2sysc]
14     | sqloGetSysMonSetting entry [eduid 1 eduname db2sysc]
15     | | sqloGetEnvUnCached entry [eduid 1 eduname db2sysc]
16     | | | EnvPrf0pen entry [eduid 1 eduname db2sysc]
17     | | | | EnvKeyName entry [eduid 1 eduname db2sysc]

19     | | | | | sqlox1tc_app entry [eduid 1 eduname db2sysc]
20     | | | | | sqlox1tc_app exit
21     | | | | | sqloGetUserAttribByName data [probe 770]
22     | | | | | sqloGetUserAttribByName data [probe 820]
23     | | | | | sqloxult_app entry [eduid 1 eduname db2sysc]
24     | | | | | sqloxult_app exit
25     | | | | | sqloGetUserAttribByName exit
26     | | | | | EnvKeyName exit
27     | | | | | EnvPrf0pen exit
28     | | | | | sqloGetEnvUnCached exit
29     | | | | | sqloGetSysMonSetting exit
30     | | | | | sqleSyscUpdateDynamicVars exit
```

To print a backtrace of all stack frames for line 5 from the `test.flw` file, execute the following command:

```
db2trc print -stack 5 test.flw
```

or

```
db2trc print -s 5 test.flw
```

The following is the output displayed:

```
pid = 1608 tid = 47604608002368 node = 0
3      sqkfFastCommManager::ResourceSelfTuning entry [eduid 1 eduname db2sysc]
4      | sqlogmt entry [eduid 1 eduname db2sysc]
5      | sqlogmt exit
```

Note that line 18 does not exist in the test.flw file. If the record ID specified does not exist, you will get an error message:

```
$ db2trc print -stack 18 test.flw
ERROR: Unable to find the Record ID 18 . Exiting.
```

See the command syntax diagram for global option command syntax information. See Example 4 in the following CF and CFCLI usage examples section for further details.

## CF and CFCLI usage examples

### Example 6

To turn trace ON for the xport\_common (CF\_TRACE\_WARNING), srv\_init (CF\_TRACE\_PATH|CF\_TRACE\_ERROR), and srv\_common (CF\_TRACE\_ALL) components, run the following command:

```
db2trc cf on -m "*.CF.xport_common,srv_init,srv_common.*.CF_TRACE_WARNING,0x09,CF_TRACE_ALL"
```

After running the command, trace will be OFF for all other components.

### Example 7

To turn trace ON for all CF client components at the CF\_TRACE\_ALL level, run the following command:

```
db2trc cfcli on
```

### Example 8

To turn trace ON for the srv\_list (CF\_TRACE\_ERROR) component and all the other components at the CF\_TRACE\_ALL level, run the following command:

```
db2trc cf on -m "*.CF.srv_list.*.CF_TRACE_ERROR,CF_TRACE_ALL"
```

### Example 9

To turn trace ON for all CF components at the CF\_TRACE\_WARNING or CF\_TRACE\_DEBUG level, run the following command:

```
db2trc cf on -m "*.CF.*.*.CF_TRACE_WARNING|CF_TRACE_DEBUG"
```

### Example 10

To turn trace ON for all CF servers listed in the db2nodes.cfg file and all the CF components at the CF\_TRACE\_ALL level, run the following command:

```
db2trc cf on
```

### Example 11

To turn trace ON for only CF server 129 listed in the db2nodes.cfg file and all the CF components at the CF\_TRACE\_ALL level, run the following command:

```
db2trc cf on -id 129
```

## Usage notes

The **db2trc** command must be issued several times in the course of conducting a trace. With the DB2 instance stopped, the general sequence would be to first turn tracing **on**, which immediately begins the collection of the specified data and storage of it in the buffer after the DB2 instance is started, then to clear the buffer before connecting to the database, followed by dumping the binary format data into a dump file, then to turn tracing **off**, and, finally, to format the dump file into

## db2trc - Trace

an easily readable text destination file. Here's an example of the commands executed to conduct a trace of the SAMPLE database, with the contents of the trace buffer written to file dmp:

```
db2trc on -i 8m -m "*.*.2.*.*" -t
db2start
db2trc clear
db2 connect to sample
db2trc dump dmp
db2trc off
```

The general syntax of the **db2trc** command is shown in the following example. The command options can be grouped into two broad stages: collection and parsing.

- *Collection* options include turning a trace on or off; specifying the trace buffer size; specifying or changing trace options; dumping a trace; and clearing the trace buffer.
- *Parsing* options include sorting the formatted trace records chronologically, or by process, or by thread.

### STAGE #1 - COLLECTION

Usage: db2trc [facility] <command> [-u]

[facility]  
db2 - DB2 instance (default)  
das - DB2 Administration Server instance

<command>  
change - Change trace options  
clear - Clear the trace buffer  
dump - Generate trace dump file  
info - Information  
off - Disable the trace facility  
on - Enable the trace facility  
stop - Stop tracing

### STAGE #2 - PARSING

Usage: db2trc <command> [-u]

<command>  
ccfmt - Dump and format a code coverage trace  
flow - Generate control flow diagram  
format - Format  
info - Information  
perffmt - Format a performance trace

For more information add the "-u" option to any of the above commands

In Stage #2 - Parsing the preceding section, the command **ccfmt** dumps and formats a "code coverage trace". The code coverage trace is an extension of **db2trc** that keeps a count of function entries, exits, probe points, and codepaths. It can be used to gather statistics on what functions are being heavily used, or which functions are not being touched during tests.

When tracing the database server, it is recommended that the trace facility be turned on before starting the database manager. This is the most reliable method for the database manager, running on any UNIX and Linux platform, to be immediately aware of trace changes.

To turn tracing ON and receive information specific to DB2 Text Search, a mask with component code for cie (155) can be used:

```
db2trc on -m "*.*.155.*.*"
```

When the specific database partitions involved in a problem are known, only trace that database partition. Use the option, **db2trc** on **-member NN**, to specify which database partitions to trace.

When the problem is related to a specific application ID, you are able to limit trace only to that specific application ID by using the db2trc option **db2trc** on **-appid <appID>**.





## db2trcon - Trace ON options for db2trc

The **db2trcon** command issues the **db2trc** command with the **on** option. This command has several options which helps with using the **db2trc** command. The **db2trcon** command can issue the **db2trc** command for a period of time and it can also issue the **db2trc** command to be turned on for the top CPU consuming EDUs.

The **db2trcon** command supports all the parameters of the **db2trc** command.

### Authorization

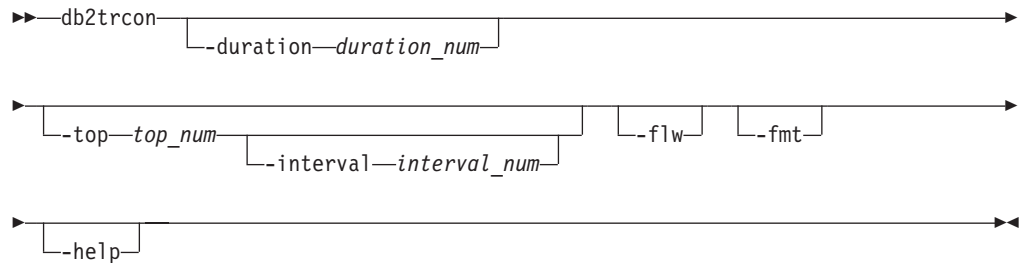
To trace a DB2 instance on a UNIX operating system, you must possess one of the following authorizations:

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

To trace the DB2 Administration Server on a UNIX operating system:

- *dasadm*

### Command syntax



### Command parameters

#### **-duration** *duration\_num*

Specifies that trace is turned on for the specified time in seconds.

#### **-top** *top\_num*

Specifies that trace is turned on for the top CPU consuming EDUs. The maximum number of EDUs for which the trace can be enabled for is 64.

#### **-interval** *-interval\_num*

Specifies the time period in seconds for collecting the top CPU consuming EDUs. If this option is not specified the default time period is 10 seconds.

#### **-flw**

Generates a trace flow report. The name of the flow files are db2trcon.<time-stamp>.flw. The value of <time-stamp> has the following format, hours\_minutes\_seconds.

#### **-fmt**

Generates a trace format report. The name of the format files are db2trcon.<time-stamp>.fmt. The value of <time-stamp> has the following format, hours\_minutes\_seconds.

#### **-help**

Displays help information.

## db2trcon - Trace ON options for db2trc

### Examples

The following example turns the **db2trc** command on for 1 minute:

```
db2trcon -duration 60
```

To turn the **db2trc** command on for the top 10 CPU consuming EDUs for the default time period of 10 seconds and to place the trace in tmpDump.dmp, issue:

```
db2trcon -top 10 -f tmpDump.dmp
```

The following example turns the **db2trc** command on for 45 seconds for the top 5 processor time consuming EDUs, collected in 15 second intervals. When the **db2trc** command is turned off, **db2trcon** generates the dump, flow and format files.

```
db2trcon -duration 45 -top 5 -interval 15 -fodc -ff
```

### Usage notes

The **db2trcon** command always generates a dump file if issued with the **-duration** parameter. The **db2trcon** command also always generates a trace dump file if it is issued without the **-f** parameter from the **db2trc** command. The name of the dump file is db2trcon.<time-stamp>.dmp. The value of <time-stamp> has the following format, hours\_minutes\_seconds.

## db2unins - Uninstall DB2 database products, features, or languages

Uninstalls one or more DB2 database products, features, or languages.

**db2unins** can be found both in the installation media and in a DB2 install copy on the system. If run from the installation media, only the **-f**, **-l**, **-t** and **-?** parameters can be used. If run from a DB2 install copy, all the options except **-f** can be used.

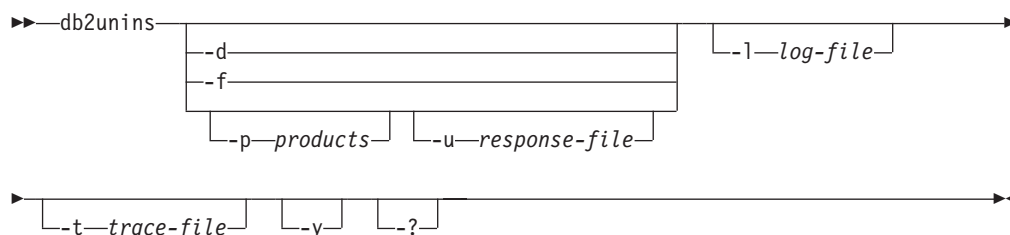
### Authorization

SYSADM

### Required connection

None

### Command syntax



### Command parameters

#### Note:

On Windows operating systems, the **db2unins** command can use the / or - switch symbols interchangeably.

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 only be issued from the installation media. Your system will reboot when you successfully issue **db2unins -f**. It can only be issued if there are no other DB2 products before version 9 installed on the system.
- p products** Specifies the products to be uninstalled. This parameter is only available when run from an installed DB2 copy. To uninstall multiple products, separate a list of products by a semicolon and enclosed in double quotation marks. When both parameters **-p** and **-u** are specified, the products specified in **-p** override the products specified in the response file. For example, `db2unins -p "ESE;QP" -u db2un.rsp` uninstalls both DB2 ESE and QP regardless of the **REMOVE\_PROD** keyword value in `db2un.rsp`.

## db2unins - Uninstall DB2 database products, features, or languages

### **-u** *response-file*

Performs an uninstallation of products, features, or languages based on what is specified in the response file. For example, `db2unins -u db2un.rsp`. This parameter is only available when run from an installed DB2 copy. If both parameters **-p** and **-u** are specified, the DB2 products specified in the **-p** parameter override the **REMOVE\_PROD** keyword in the response file.

If you have a clustered environment, before uninstalling your DB2 product using a response file, you must first run the **db2mscs** command, with the **-u** option, from the same server that originally ran the **db2mscs** command to create the failover infrastructure. For details, see the **db2mscs** command.

### **-l** *log-file*

Specifies the location of the log file. The default log file location is `My Documents\DB2LOG\db2un_<timestamp>.log`.

### **-t** *trace-file*

Turns on the trace functionality. The trace file will be used for debugging problems with the **db2unins** command.

**-y** Ensures that no confirmation is done during the uninstallation process.

**-?** Displays help for the **db2unins** command.

## Usage notes

- On Windows operating systems, a file named `db2unins` can be found in the root directory of the installation image, as well as, installed under the installation path `installPath\BIN`. However, although they have the same filename, these files are not the same and behave differently. Do not perform a copy and paste operation of the file from the installation image to the installation path directory. If you do, problems will occur.
- On Windows operating systems, when specifying the full path, the path name must be contained within double quotation marks.
- If you want to use **db2unins -f** to manually remove all the DB2 database products on the system, you should use the utility from the version which is equal to the highest DB2 product version on the system. For example, if you have two copies installed, `DB2COPY1` which is DB2 V9.1 and `DB2COPY2` which is DB2 V9.5, run **db2unins -f** to remove both DB2 versions from the DB2 V9.5 product image. If you run **db2unins -f** from the DB2 V9.1 product image, it will not clean the machine completely.
- If there are instances that are clustered with Microsoft Cluster Service (MSCS), you can uncluster the instance by issuing the **db2mscs** or **db2iclus** command before uninstallation.

## 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, perform one of the following actions:

- 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.

### db2updserv - Show product updates

Shows the product updates and enhancements available for DB2 database products. On Windows, the output from this command goes to a Web page, on UNIX, it goes to a Java application .

#### Authorization

None

#### Required Connection

Internet connection required.

For UNIX and Linux systems, if you connect to the internet through an HTTP proxy server, you must specify your HTTP proxy server host name and port in the *DB2DIR*</varname>/java/jdk64/jre/lib/net.properties file before you run the **db2updserv** command. For more information, see the Usage Notes section.

#### Command Syntax

▶—db2updserv—◀

#### Command parameters

None

#### Usage notes

- For UNIX and Linux systems only:

When using an HTTP proxy server, you must set your HTTP proxy host name and port in a properties file before you can use the **db2updserv** command. To set up your HTTP proxy server host name and port, edit the *DB2DIR*/java/jdk64/jre/lib/net.properties file, where *DB2DIR* is the location where the current version of the DB2 database product is installed. Edit the file to include the following parameters:

```
http.proxyHost=host name
http.proxyPort=port number
```

Where *host name* is the host name of your HTTP proxy server and *port number* is the port number for your HTTP proxy.

For example:

```
http.proxyHost=proxy.mycompany.com
http.proxyPort=80
```

If other Java applications need to connect to other servers directly and not through an HTTP proxy server, than specify the server host names in the nonProxyHosts parameter.

## db2val - DB2 copy validation tool

Verifies the basic functions of a DB2 copy by checking the state of installation files, instance setup, and local database connections.

### Authorization

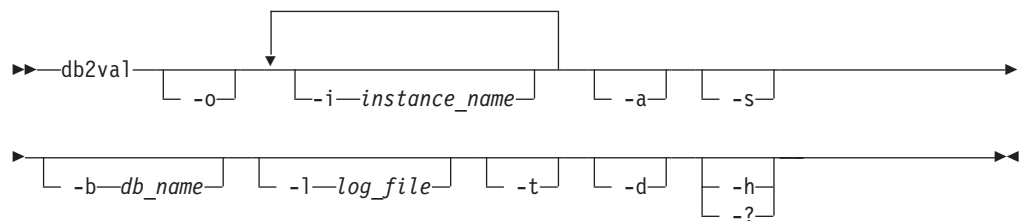
Instance validation requires one of the following authorities:

- On root copies, root authority is required on Linux and UNIX operating systems.
- SYSADM plus one of the following authorities:
  - Instance owner
  - Root access on Linux and UNIX operating systems, or Local Administrator authority on Windows operating systems

### Required Connection

None.

### Command syntax



### Command parameters

- o** Specifies that only the installation files will be validated; validation of the instance, database, and extended security will not be performed. If this parameter is specified, the `-i`, `-a`, `-b`, and `-s` parameters are ignored.
- i *instance\_name*** Specifies that the name of the instance to validate. To specify multiple instances are to be validated, specify this parameter multiple times. For example, `-i inst1 -i inst2`. On Windows operating systems, if this parameter is not specified, the current instance will be used as the default value. On Linux and UNIX operating systems, this parameter can only be used by root users in a root installation of a DB2 copy.
- a** Validates all instances in the DB2 copy. On Linux and UNIX operating systems, this parameter can only be used by root users in a root installation of a DB2 copy. This parameter overrides parameter `-i`.
- b <db\_name>** Validates database creation and connections to the database specified. Only active DB2 instances will be validated and this parameter will be ignored for DB2 client and DB2 pureScale instances.
- t <trace\_file>** This parameter applies only to Linux and UNIX operating systems. Specifies the full path and name of trace file specified by `trace_file`.

## db2val - DB2 copy validation tool

- d This parameter is deprecated and might be removed in a future release. Use the **-t** parameter instead. Valid only on Linux and UNIX operating systems. Use this parameter only when instructed by DB2 Support. Turns the debug mode on.
- s Starts the DB2 database manager for the specified instance that is part of a partitioned database environment.
- l <log\_file>  
Writes the log to the file name specified. Unless the -l parameter is specified, the default log path on Linux and UNIX operating systems is /tmp/db2valxx.log and on Windows operating systems is My Documents\DB2LOG\db2valxx.log, where xx is a generated value.
- ? | -h Displays usage information for the **db2val** command.

### Examples

To validate the instance TEST1 and the database DATA1, run the following command:

```
db2val -i TEST1 -b DATA1
```

To validate all the instances for the DB2 copy, run the following command:

```
db2val -a
```

To validate only the DB2 installation files, run the following command:

```
db2val -o
```

### Usage Notes

You can use the db2val command only to validate installation files in a DB2 pureScale environment.



---

## db2xdbmig - Migrate XSR objects

Migrates all XML schema repository (XSR) objects that are enabled for decomposition to the current version and service level of the DB2 copy where you are running the command.

This command is located in the *DB2DIR/bin* directory, where *DB2DIR* represents the installation location where the current version of the DB2 database system is installed.

### Authorization

CREATE, ALTER and DROP privileges on all of the XSR objects in the database.

### Command syntax

▶—db2xdbmig—*database-alias*—▶

### Command parameters

*database-alias*

Specifies the alias of the database that contains the XSR objects.

### Usage notes

- The **db2xdbmig** command affects only decomposition-enabled XML schemas.
- When migrating to DB2 Version 9.7 from a DB2 Version 9.1 GA or Fix Pack 1 copy, the **UPGRADE DATABASE** command implicitly runs the **db2xdbmig** command. You do not need to run this command in DB2 Version 9.7.

---

### db2xpvt - Format trap file

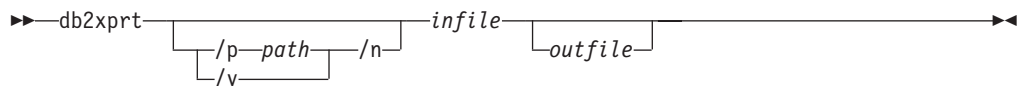
Formats the DB2 database binary trap files into a human readable ASCII file.

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 **db2xpvt** command uses DB2 symbol files (.PDB) in order to format the trap files.

#### Authorization

You must have access to the **diagpath** directory.

#### Command syntax



#### 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:

```
db2xpvt DB30882416.TRP DB30882416.FMT
```



### 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 Linux operating systems.

The `doce_deinstall` command is located at `DB2DIR/install`, where `DB2DIR` is the location where the current version of the *DB2 Information Center* is installed. The `doce_deinstall` command is also available from the DOCE DVD. If run from the DVD, the `doce_deinstall` command requires a `-b` option.

#### Authorization

Root user authority

#### Required Connection

None

#### Command syntax

```
doce_deinstall -a [-b installpath] [-l log-file]
                 [-t trace-file] [-r response_file] [-h | -?]
```

#### Command parameters

- a** Removes the Information Center from its current location.
- b** This option is valid if the command is run from the DB2 database product media. It specifies the absolute path where the DB2 database product was installed and will be uninstalled. The command will prompt for the path if the option is not specified.
- 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`.
- r response\_file**  
Removes the Information Center using a response file. For example, `doce_deinstall -r db2un.rsp`. Cannot be combined with the **-a** parameter.
- h | -?**  
Displays usage information.

#### Example

To uninstall a DB2 Information Center that is installed in `/opt/ibm/db2/doce`, issue:

```
cd /opt/ibm/db2/doce
doce_deinstall -a
```

## 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/V10.1`. This command applies only to the Linux operating systems.

### Authorization

Root user authority

### Required Connection

None

### Command syntax

```

▶▶ doce_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 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/V10.1`. 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.

#### **-n**

Specifies noninteractive 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`.

#### **-l** *log-file*

Specifies the log file. The default log file is `/tmp/doce_install.log$$`, where `$$` is the process ID.

## doce\_install - Install DB2 Information Center

**-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.

### Example

- 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/V10.1`, from an image in `/mnt/cdrom`, non-interactively in English, issue:

```
cd /mnt/cdrom
./doce_install -p doce -b /db2/V10.1 -n
```

## enable\_MQFunctions - Enable WebSphere MQ functions

Enables DB2WebSphere MQ functions for the specified database and validates that the DB2WebSphere 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 authorities:

- CREATE\_EXTERNAL\_ROUTINE authority on the database and at least one of the following authorities:
  - If the schema name of the function does not refer to an existing schema, IMPLICIT\_SCHEMA authority on the database
  - If the schema name DB2MQ or DB2MQ1C exists, CREATEIN privilege on the schema
- DBADM

### Command syntax

```

▶▶ enable_MQFunctions --n database --u userid --p password
  ┌─q queuemanager┐ ┌─force┐ ┌─xml┐ ┌─xmlSize┐ ┌─c clobSize┐
  └────────────────┘ └──────┘ └───┘ └────────┘ └──────────┘
▶ ┌─novalidate┐ ┌─v┐ ┌─all┐
  └──────────┘ └──┘ └──┘
                    └─0pc┘
                    └─1pc┘
  
```

### Command parameters

**-n** *database*

Specifies the name of the database that you want to enable.

**-u** *userid*

Specifies the user ID to connect to the database.

**-p** *password*

Specifies the password for the user ID.

**-q** *queuemanager*

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.

**-echo** Optional. Prints the detailed SQL used to create the UDFs or diagnostic information.

**-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.

## enable\_MQFunctions - Enable WebSphere MQ functions

### -xml *xmlSize*

Optional. This is for defining the XML versions of the 0pc functions. This option has no effect if the -v 1pc option is specified.

The *xmlSize* specifies the length of XML data. The minimum length is 1 byte. The maximum length is 100M. The default is 1M. You can specify the length as *n* (number of bytes), *nK* (length in kilobytes), or *nM* (length in megabytes).

### -c *clobSize*

Optional. Specifies the length of CLOB data. The minimum length is 1 byte; this is the default. The maximum length is 100M. You can specify the length as *n* (number of bytes), *nK* (length in kilobytes), or *nM* (length in megabytes).

### -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

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.



---

## installDSDriver - Extract Data Server Driver components

Installs Data Server Driver (dsdriver) components.

On Linux and UNIX operating systems, installs all the components of Data Server Driver (dsdriver) in the current directory and removes the .tar files that have been extracted.

### Authorization

None

### Command syntax

▶▶—installDSDriver -h▶▶

### Command parameters

**-h**      Displays usage information.

### Usage notes

- For 64-bit dsdriver, the script will install 64-bit drivers and remove 64-bit tar files, but will leave 32-bit tar files and directories that are also present in the package.
- For 32-bit dsdriver, if you need 32-bit drivers, you must manually extract the 32-bit files.

### installFixPack - Update installed DB2 database products

Update the installed DB2 database products in a given location, on all UNIX and Linux platforms, to the same level as the image. If there are multi-copy DB2 database 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.

Fix pack installation will proceed when the database manager (DBM) of every instance (and in a partitioned database environment, every database partition) related to the installation path is stopped, and all DB2 libraries are unloaded. If all the preconditions are satisfied, **installFixPack** will update those instances and DAS related to the installation path. An additional manual update is not required. For all UNIX and Linux operating systems, the **djxlink** bind command will be launched automatically when the database is reconnected or when applications are restarted.

In some cases, you may specify different force options to continue the fix pack installation, for example, when not all DBMs are stopped, or DB2 libraries remain loaded. **installFixPack** will continue, but you may need to manually update the instances and DAS, as well as restart the applications.

For a partitioned database environment instance, install the fix pack on all the database partitions, while the instance update is only needed on the instance owning database partition. To keep the instance fully functional after the update, it is recommended to install all the products and features on all the database partitions, at least on the instance owning database partition.

If you are applying the fix pack to a dsf instance type (that is, a DB2 pureScale instance associated with the DB2 copy in a DB2 pureScale environment), you must install the fix pack to a new path. You cannot apply the fix pack to the original installation path. The path where the fix pack is installed must be different than the existing installation. In addition, this new path must be the same on all hosts. When installing a fix pack to a new location, the DB2 pureScale instance does not need to be stopped before installing the fix pack.

If you are applying the fix pack to an ese instance type, you can either apply the fix pack on top of the original installation path or you can install the fix pack to a new path. If you install the fix pack on top of the original installation path, the instance must be stopped before applying the fix pack. For details, see: "Installing a fix pack to update a Version 9.8 ese or dsf instance type" in *Installing DB2 Servers*

#### Authorization

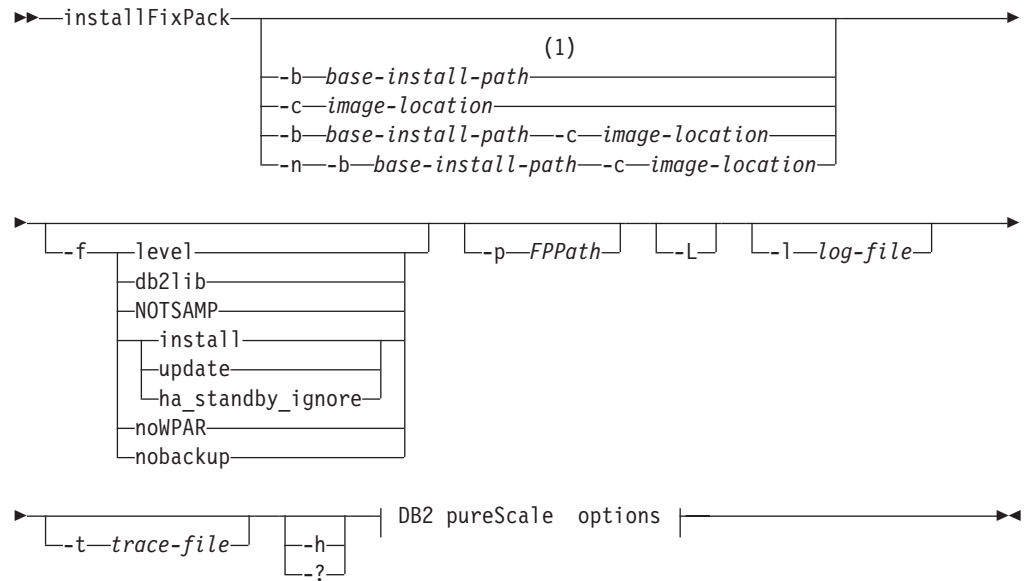
Root installations require root user authority. For non-root installations, you must log on with the user ID that owns the non-root installation.

#### Required Connection

None

#### Command syntax

## installFixPack - Update installed DB2 database products



### DB2 pureScale options:



### Notes:

- 1 If you omit this option, you will be prompted for the required information without an error message halt.

### Command parameters

**-n** Specifies non-interactive mode. When specified, you must also specify **-b**, **-p**, and **-c**. This mode can be used for applications to provide all the required information at the command line in an unattended mode.

#### **-b** *base-install-path*

Specifies the path where the DB2 database product will be installed. Mandatory when **-n** is specified. The length of the path is limited to 128 characters and is a full path name.

The **-b** option is not required for a non-root installation of DB2, but it is still mandatory for a root installation. If **-b** is used in a non-root install, the value of *base-install-path* must be the user's `HOME/sql1ib` directory, or else the path is considered invalid. If **-b** is not provided in a non-root install, the DB2 installer will use the user's `HOME/sql1ib` as the install path and continue. But, if **-b** is used and the provided install path has a DB2 copy installed by a root user, the path is considered invalid since the DB2 copy can only be updated by the user who installed it.

#### **-c** *NLPACK\_location*

Specifies the location of the related DB2 National Language Pack (NLPACK). This parameter is mandatory when **-n** is specified. The DB2 NLPACK location needs to be provided explicitly if all of the following conditions are met:

- The **-n** option is specified.
- The installation requires National Language (non-English) support.

## installFixPack - Update installed DB2 database products

- The DB2 NLPACK is neither on the DB2 DVD nor in the same subdirectory as the DB2 database product being installed.
- f** This parameter is ignored in a DB2 pureScale environment. Force option. **-f** with no argument is not supported. The following force arguments can be combined. For example, **-f level -f db2lib**.
- f level**  
Force a down level or same level fix pack install. If the fix pack image is at a higher level than the installed DB2 database product, this option is ignored.
- f db2lib**  
This parameter is ignored in a DB2 pureScale environment. Force **installFixPack** to bypass the checking on DB2 library loading. To ensure that the instances or applications work properly after the installation, the DBM must be stopped for all the related instances (including all database partitions for the related partitioned database environment instances), and all DB2 libraries related to the installation path must be unloaded.
- f NOTSAMP**  
This parameter is ignored in a DB2 pureScale environment. Specifies that the SA MP should not be updated (applicable only to root installation).
- f install**  
Force the **installFixPack** command to bypass all checking on DB2 library loading and checking that the instance and DAS are stopped. To ensure that the instances or applications work properly after the installation, the DBM must be stopped for all the related instances (including all database partitions for the related partitioned database environment instances), and all DB2 libraries related to the installation path must be unloaded. If this option is specified, neither the instance nor DAS are updated. After the installation, you need to manually update the instance and DAS. Also, note that the options update, install, and ha\_standby\_ignore are mutually exclusive and cannot be specified in the same installation.
- f update**  
Force the **installFixPack** command to bypass all the checking on DB2 library loading, and checking that the instance and DAS are stopped. To ensure that the instances or applications work properly after the installation, the DBM must be stopped for all the related instances (including all database partitions for the related partitioned database environment instances), and all DB2 libraries related to the installation path must be unloaded. If this option is specified, both the instance and DAS are updated. Also, note that the options update, install, and ha\_standby\_ignore are mutually exclusive and cannot be specified in the same installation.
- f ha\_standby\_ignore**  
This parameter ignores the check for the **sqllib** directory by forcing the **installFixPack** command to bypass this check. For example, the check for **sqllib** directory is ignored in a clustered environment where the standby node does not have the **sqllib** library mounted. To ensure that the instances or applications work properly after the installation, the DBM must be stopped for all the related instances (including all database partitions for the related partitioned database environment instances), and all DB2 libraries related to the installation path must be unloaded. If this option is specified, neither the instance nor DAS are updated. After the installation, you need to manually update the instance and DAS. Also, note

## installFixPack - Update installed DB2 database products

that the options update, install, and ha\_standby\_ignore are mutually exclusive and cannot be specified in the same installation.

### **-f noWPAR**

This parameter is ignored in a DB2 pureScale environment. Applicable to AIX 6.1 or later in a global environment. Force **installFixPack** not to do any checking or any actions on the AIX system workload partitions (WPARs) which share the DB2 copy being updated on the global environment. If **-f noWPAR** is specified, you must manually update the instances and DAS on each system WPAR that shares this DB2 copy.

### **-f nobackup**

This parameter is ignored in a DB2 pureScale environment. Force **installFixPack** to not backup installation files when the components are updated. If you choose not to backup the files, the space requirement of the installation directory is reduced. However, choosing not to backup the files also means that if any errors occur, the DB2 installer will not be able to perform a rollback operation. In this case, you will need to manually clean up the files and reinstall the fix pack.

### **-p FPPath**

Specify where the fix pack is to be installed. In a DB2 pureScale environment, this parameter is mandatory. Consider the following when using this parameter:

- When this parameter is specified, instances are not updated automatically. You must manually update the instances using the **db2iupdt** command. In addition, in a DB2 pureScale environment, if the fix pack is applied on multiple hosts, you must run the **db2iupdt** command on all hosts.
- If you are installing a fix pack to update a DB2 pureScale instance (dsf type instance), the path where the fix pack is installed must be different than the existing installation, and, this path must be the same on all hosts. For fix pack installations outside of a DB2 pureScale environment, the path can be different than the existing installation.

**-L** Indicates the fix pack is to be applied locally to the current host only. When this parameter is specified, you cannot specify the **-H** parameter.

### **-H hostsListFile**

This parameter can only be specified in a DB2 pureScale environment. This parameter enables a fix pack update across multiple hosts. All hosts specified in the host list file must have the same service level. When this parameter is specified, you cannot specify the **-L** parameter.

**Note:** After running the **installFixPack** command with this parameter, always check the log file to ensure that there are no installation errors on any of the remote hosts.

### **-l log-file**

Specifies the log file. For root installations, the default log file is `/tmp/installFixPack.log$$`, where `$$` represents the process ID. For non-root installations, the default log file is `/tmp/installFixPack_userID.log`, where `userID` represents the user ID that owns the non-root installation. If the IBM Tivoli System Automation for Multiplatforms (SA MP) is being installed or updated with the **installFixPack** command, the corresponding log file will be located in the same directory as DB2 log files.

## installFixPack - Update installed DB2 database products

**-t** *trace-file*

Turns on the debug mode. The debug information is written to the file name specified.

**-h | -?**

Displays help information.

### Usage notes

- If you have DB2 Text Search installed and Text Search is running on any instances related to the DB2 copy, installation attempts to stop the Text Search service but continues with the installation.
- Format of the host list file can be shown in the following section:

```
HOST=host1
host1.HOSTNAME=<hostname1>
HOST=host2
host2.HOSTNAME=<hostname2>
...
```

### Examples

- To perform an interactive update from GA to Fix Pack 1 when DB2 Enterprise Server Edition German is installed on /opt/ibm/db2/COPY1, from the Fix Pack 1 image, issue:  

```
./installFixPack -b /opt/ibm/db2/COPY1
```
- To perform a silent update from GA to Fix Pack 1 when DB2 Enterprise Server Edition German is installed on /opt/ibm/db2/COPY1, from the Fix Pack 1 image, issue:  

```
./installFixPack -b /opt/ibm/db2/COPY1 -c full_path_to_NLPACK_image -n
```
- If for any reason the installed DB2 database product files get corrupted, instead of uninstalling and installing again to refresh the installation, issue:  

```
./installFixPack -f level -b full_path_where_DB2_product_installed
```
- To reduce the space requirement of the installation directory, issue:  

```
./installFixPack -f nobackup -b full_path_where_DB2_product_installed
```
- To apply a fix pack to update a DB2 pureScale instance (dsf type instance), the path where the fix pack is installed must be different than the existing installation, and, this path must be the same on all hosts:  

```
./installFixPack -b full_path_where_DB2_product_installed -p path_where_DB2_fixpack_installed -L
```
- To apply a fix pack to update an ese instance type, the path can be the same as the existing installation, or the path can be different:  

```
./installFixPack -b full_path_where_DB2_product_installed -p full_path_where_DB2_product_installed -L
```
- In a DB2 pureScale environment, to apply a fix pack update across multiple hosts:  

```
./installFixPack -b full_path_where_DB2_product_installed -p full_path_where_DB2_product_installed -H hostsListFile
```

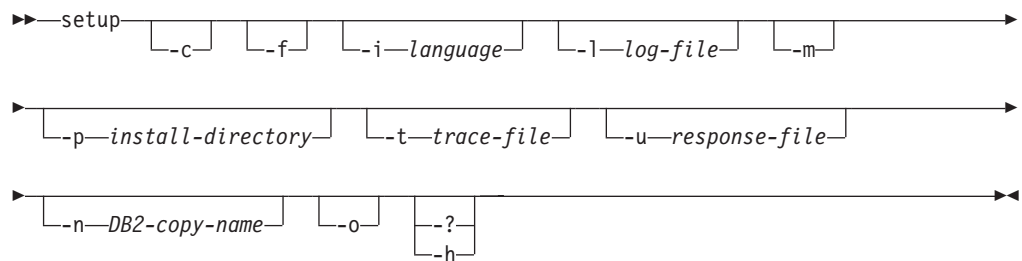
## setup - Install DB2 database products

Installs DB2 database 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 database products. If invoked with the **-u** option, it performs an installation without further input, taking installation configuration information from a response file.

When installing the IBM Data Server Runtime Client on Windows, the setup options are different from DB2 product installation. Refer to “IBM Data Server Runtime Client installation command line options (Windows)” for the appropriate options.

### Command syntax



### Command parameters

#### Note:

The install DB2 **setup** command can use the / or - switch symbols interchangeably.

- 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.

## setup - Install DB2 database products

- n *DB2-copy-name*  
Specifies the DB2 copy name that you want the install to use. Specifying this option overrides the copy name that is specified in the response file.
- o  
Always perform a new copy installation with a generated default copy name. This option is only available for installing the IBM Data Server Driver Package on Windows.
- ? | -h  
Generates usage information.

### Usage notes

- When specifying the full path, the path name must be contained within double quotation marks.



---

## Chapter 8. DB2 Text Search commands

## db2ts ALTER INDEX

This command changes the update characteristics of an index.

For execution, the command needs to be prefixed with **db2ts** at the command line.

### Authorization

The privileges held by the authorization ID of the statement must include the SYSTS\_MGR role and at least one of the following authorities:

- DBADM authority
- ALTERIN privilege on the base schema
- CONTROL or ALTER privilege on the base table on which the text search index is defined

To change an existing schedule, the authorization ID must be the same as the index creator or must have DBADM authority.

### Required connection

Database

### Command syntax

```

▶▶ ALTER INDEX index-name FOR TEXT | update characteristics | options |
▶ | connection options |
    
```

#### update characteristics:

```

| UPDATE FREQUENCY | NONE |
|                   | update frequency |
▶ | incremental update characteristics |
    
```

#### update frequency:

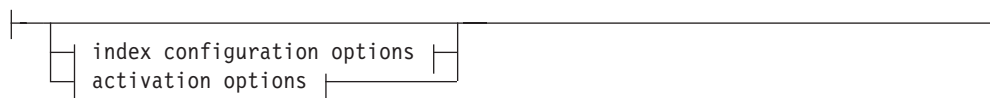
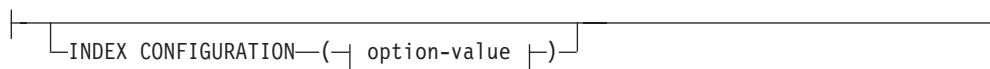
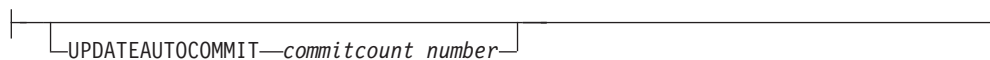
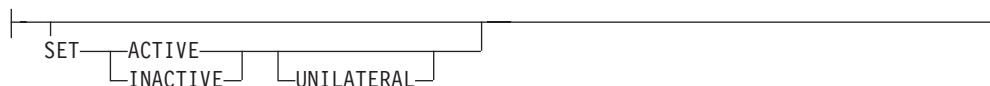
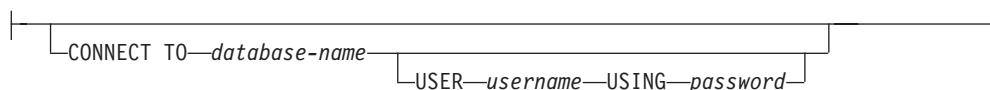
```

| D ( * ) | H ( * ) | M ( integer3 ) |
|         |         |         |
|         |         |         |
| integer1 | integer2 |
    
```

#### incremental update characteristics:

```

| UPDATE MINIMUM | minchanges |
    
```

**options:****index configuration options:****option-value:****activation options:****connection options:****Command parameters****ALTER INDEX** *index-name*

The schema and name of the index as specified in the **CREATE INDEX** command. It uniquely identifies the text search index in a database.

**UPDATE FREQUENCY**

Specifies the frequency with which index updates are made. The index will be updated, if the number of changes is at least the value set for **UPDATE MINIMUM**. The update frequency **NONE** indicates that no further index updates will be made. This can be useful for a text column in a table with data that will not change. It is also useful when the user intends to manually update the index (using the **UPDATE INDEX** command). Automatic updates can only be done if the **START FOR TEXT** command has been run and the DB2 Text Search instance services are running.

The default frequency value is taken from the view SYSIBMTS.TSDEFAULTS, where DEFAULTNAME='UPDATEFREQUENCY'.

**NONE**

No automatic updates will be applied to the text index. Any further index update will have to be started manually.

**D** The day(s) of the week when the index is updated.

\* Every day of the week.

## db2ts ALTER INDEX

*integer1*

Specific days of the week, from Sunday to Saturday: 0 to 6

**H** The hour(s) of the specified day(s) when the index is updated.

\* Every hour of the day.

*integer2*

Specific hours of the day, from midnight to 11 pm: 0 to 23

**M** The minute(s) of the specified hour(s) when the index is updated.

*integer3*

Specified as top of the hour (0), or in multiples of 5 minute increments after the hour: 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 or 55

If you do not specify the **UPDATE FREQUENCY** option, the frequency settings are left unchanged.

### **UPDATE MINIMUM** *minchanges*

Specifies the minimum number of changes to text documents that must occur before the index is incrementally updated. Multiple changes to the same text document are treated as separate changes. If you do not specify the **UPDATE MINIMUM** option, the setting is left unchanged.

### **INDEX CONFIGURATION** (*option-value*)

Starting with Version 9.7 Fix Pack 3 and later fix packs, this is an optional input argument of type VARCHAR(32K) that allows altering text index configuration settings. The following option is supported:

Table 53. Specifications for option-value

Option	Value	Data type	Description
SERIALUPDATE	<i>updatemode</i>	Integer	Specifies whether the update processing for a partitioned text search index should be run in parallel or in serial mode. In parallel mode the execution is distributed to the database partitions and executes independently on each node. In serial mode the execution is run without distribution and stops when a failure is encountered. Serial mode execution usually takes longer but requires significantly less resources. <ul style="list-style-type: none"> <li>• 0 = parallel mode</li> <li>• 1 = serial mode</li> </ul>

Table 53. Specifications for option-value (continued)

Option	Value	Data type	Description
UPDATEAUTOCOMMIT	<i>commitcount_number</i>	Integer	<p>Specifies the number of index updates after which a commit is executed to preserve the previous work automatically for either initial or incremental updates.</p> <ul style="list-style-type: none"> <li>For initial updates, the index update will process batches of documents from a base table after the trigger to capture data updates is activated. After the amount of documents updated reaches the COMMITCOUNT number, the server will do an interim commit. Log entries generated by unprocessed documents will be removed from the staging table. Using the UPDATEAUTOCOMMIT option for an initial text index update will lead to a significant increase of execution time.</li> <li>For incremental updates, log entries which have been processed will be removed correspondingly from staging table with each interim commit. COMMITCOUNT counts the number of documents updated, not the number of staging table entries.</li> </ul>

*activation options*

Starting with Version 9.7 Fix Pack 3 and later fix packs, this input argument of type integer sets the status of a text index.

**ACTIVE**

Sets the text index status to active

**INACTIVE**

Sets the text index status to inactive

**UNILATERAL**

Specifies a unilateral change which only affects the status of DB2 Text Search indexes. If this argument is specified, only the status of a DB2 Text Search index is changed to active or inactive. Without the UNILATERAL argument, the activation status of the DB2 Text Search and DB2 Net Search Extender indexes is jointly switched so that only one of the text indexes is active.

**CONNECT TO** *database-name*

This clause specifies the database to which a connection will be established. The database must be on the local system. If specified, this clause takes precedence over the environment variable DB2DBDFT. This clause can be omitted if the following are all true:

- The DB2DBDFT environment variable is set to a valid database name.
- The user running the command has the required authorization to connect to the database server.

**USER** *username* **USING** *password*

This clause specifies the username and password that will be used to establish the connection.

**Usage notes**

All limits and naming conventions, that apply to DB2 database objects and queries, also apply to DB2 Text Search features and queries. DB2 Text Search related

## db2ts ALTER INDEX

identifiers must conform to the DB2 naming conventions. In addition, there are some additional restrictions. For example, these identifiers can only be of the form:

```
[A-Za-z][A-Za-z0-9@#$_]*
```

or

```
"[A-Za-z ][A-Za-z0-9@#$_ ]*"
```

Note that multiple commands cannot be executed concurrently on a text search index if they may conflict. If this command is issued while a conflicting command is running, an error will occur and the command will fail, after which you can try to run the command again. Some of the conflicting commands are:

- **ALTER INDEX**
- **CLEAR EVENTS FOR INDEX**
- **DROP INDEX**
- **UPDATE INDEX**
- **DISABLE DATABASE FOR TEXT**

Changes to the database: Updates the DB2 Text Search catalog information.

The result of activating indexes depends on the original index status. The following table describes the results.

*Table 54. Status changes without invalid index:*

<b>Initial DB2 Text Search or Net Search Extender Status</b>	<b>Request Active</b>	<b>Request Active Unilateral</b>	<b>Request Inactive</b>	<b>Request Inactive Unilateral</b>
Active / Inactive	No change	No change	Inactive / Active	Inactive / Inactive
Inactive / Active	Active / Inactive	Error	No change	No change
Inactive / Inactive	Active / Inactive	Active / Inactive	Inactive / Active	No change

SQL20427N and CIE0379E error messages are returned for active index conflicts.

---

## db2ts CLEANUP FOR TEXT

Cleans up DB2 Text Search collections within an instance or within a database.

When a cleanup operation is executed for a database, invalid text indexes and their associated collections are dropped. When a cleanup operation is executed for the instance, obsolete collections are removed. A collection can become obsolete if a database containing text search indexes is dropped before DB2 Text Search has been disabled for the database.

**Note:** While the commands operate on text search indexes, text search server tools operate on text search collections. A text search collection refers to the underlying representation of a text search index. The relationship between a text search index and its associated collections is 1:1 in a non-partitioned setup and 1:n in a partitioned setup, where n is the number of data partitions. Query the SYSIBMTS.TSCOLLECTIONNAMES catalog table to determine the text search collections for a text search index. For additional information, see the topic about Administration Tool for DB2 Text Search.

For execution, the command needs to be prefixed with **db2ts** at the command line.

### Authorization

To issue the command on instance level, you must be the owner of the text search server process. For the integrated text search server, this is the instance owner.

To issue the command on database level, the privileges held by the authorization ID of the statement must include the SYSTS\_ADM role and the DBADM authority.

### Required connection

This command must be issued from the DB2 database server.

### Command syntax

Instance level

▶▶—CLEANUP FOR TEXT—▶▶

Database level

▶▶—CLEANUP FOR TEXT—connection-options—▶▶

### Command parameters

None

## db2ts CLEAR COMMAND LOCKS

Removes all command locks for a specific text search index or for all text search indexes in the database. A command lock is created at the beginning of a text search index command, and is destroyed when it is done. It prevents undesirable conflict between different commands.

Use of this command is required in the rare case that locks remain in place due to an unexpected system behavior, and need to be cleaned up explicitly.

For execution, the command needs to be prefixed with **db2ts** at the command line.

### Authorization

The privileges held by the authorization ID of the statement used to clear locks on the index must include both of the following authorities:

- SYSTS\_MGR role
- DBADM authority or CONTROL privilege on the base table on which the index is defined

The privileges held by the authorization ID of the statement used to clear locks on the database connection must include the SYSTS\_ADM role.

### Required connection

Database

### Command syntax

```

▶▶ CLEAR COMMAND LOCKS [FOR INDEX index-name] FOR TEXT
▶ | connection options |

```

### connection options:

```

| CONNECT TO database-name [USER username USING password]

```

### Command parameters

#### FOR INDEX *index-name*

The name of the index as specified in the **CREATE INDEX** command.

#### CONNECT TO *database-name*

This clause specifies the database to which a connection will be established.

The database must be on the local system. If specified, this clause takes precedence over the environment variable **DB2DBDFT**. This clause can be omitted if the following are all true:

- The **DB2DBDFT** environment variable is set to a valid database name.
- The user running the command has the required authorization to connect to the database server.



**USER** *username* **USING** *password*

This clause specifies the authorization name and password that will be used to establish the connection.

### Usage notes

You would invoke this command because the process owning the command lock is dead. In this case, the command (represented by the lock) may not have completed, and the index may not be operational. You need to take appropriate action. For example, the process executing the **DROP INDEX** command dies suddenly. It has deleted some index data, but not all the catalog and collection information. The command lock is left intact. After clearing the **DROP INDEX** command lock, you may want to re-execute the **DROP INDEX** command. In another example, the process executing the **UPDATE INDEX** command is interrupted. It has processed some documents, but not all, and the command lock is still in place. After reviewing the text search index status and clearing the **UPDATE INDEX** command lock, you can re-execute the **UPDATE INDEX** command.

When this command is issued, the content of the DB2 Text Search view SYSIBMTS.TSLOCKS is updated.

## db2ts CLEAR EVENTS FOR TEXT

This command deletes indexing events from an index's event table used for administration. The name of this table can be found in the view SYSIBMTS.TSINDEXES in column EVENTVIEWNAME.

Every index update operation that processes at least one document produces informational and, in some cases, error entries in the event table. For automatic updates, this table has to be regularly inspected. Document specific errors have to be corrected (by changing the document content). After correcting the errors, the events can be cleared (and should be, in order not to consume too much space).

For execution, the command needs to be prefixed with **db2ts** at the command line.

### Authorization

The privileges held by the authorization ID of the statement must include both of the following authorities:

- SYSTS\_MGR role
- DBADM with DATAACCESS authority or CONTROL privilege on the table on which the index is defined

### Required connection

Database

### Command syntax

```
▶▶ CLEAR EVENTS FOR INDEX—index-name—FOR TEXT—| connection options |▶▶
```

#### connection options:

```
|
| CONNECT TO—database-name—
| USER—username—USING—password—
|
```

### Command parameters

*index-name*

The name of the index as specified in the **CREATE INDEX** command. The index name must adhere to the naming restrictions for DB2 indexes.

**CONNECT TO** *database-name*

This clause specifies the database to which a connection will be established. The database must be on the local system. If specified, this clause takes precedence over the environment variable DB2DBDFT. This clause can be omitted if the following are all true:

- The DB2DBDFT environment variable is set to a valid database name.
- The user running the command has the required authorization to connect to the database server.

**USER** *username* **USING** *password*

This clause specifies the authorization name and password that will be used to establish the connection.

## Usage notes

All limits and naming conventions, that apply to DB2 database objects and queries, also apply to DB2 Text Search features and queries. DB2 Text Search related identifiers must conform to the DB2 naming conventions. In addition, there are some additional restrictions. For example, these identifiers can only be of the form:

```
[A-Za-z][A-Za-z0-9@#$_]*
```

or

```
"[A-Za-z ][A-Za-z0-9@#$_ ]*"
```

When regular updates are scheduled (see **UPDATE FREQUENCY** options in **CREATE INDEX** or **ALTER INDEX** commands), the event table should be regularly checked. To cleanup the DB2 Text Search event table for a text search index, use the **CLEAR EVENTS FOR INDEX** command after you have checked the reason for the event and removed the source of the error.

Be sure to make changes to all rows referenced in the event table. By changing the rows in the user table, you ensure that the next **UPDATE INDEX** attempt can be made to successfully re-index the once erroneous documents.

Note that multiple commands cannot be executed concurrently on a text search index if they may conflict. If this command is issued while a conflicting command is running, an error will occur and the command will fail, after which you can try to run the command again. Some of the conflicting commands are:

- **CLEAR EVENTS FOR INDEX**
- **UPDATE INDEX**
- **ALTER INDEX**
- **DROP INDEX**
- **DISABLE DATABASE FOR TEXT**

Changes to the database: The event table is cleared.

## db2ts CREATE INDEX

The **db2ts CREATE INDEX** command creates a text search index for a text column. You can then search the column data using text search functions.

The text search index does not contain any data until you run the text search **UPDATE INDEX** command, or the DB2 Administrative Task Scheduler runs the **UPDATE INDEX** command according to the defined update frequency for the index.

For execution, the command needs to be prefixed with **db2ts** at the command line.

### Authorization

The authorization ID of the statement must include the SYSTS\_MGR role and CREATETAB authority on the database and one of the following privileges or authority :

- CONTROL privilege on the table on which the index will be defined
- INDEX privilege on the table on which the index will be defined and one of the following authorities:
  - IMPLICIT\_SCHEMA authority on the database, if the implicit or explicit schema name of the index does not exist
  - CREATEIN privilege on the schema, if the schema name of the index exists
- DBADM authority

### Required connection

Database

### Command syntax

```

▶▶ CREATE INDEX index-name FOR TEXT ON schema-name table-name
▶ ( text-column-name )
  ( function-name ( text-column-name ) )
▶ | text default information | | update characteristics |
▶ | storage options | | index configuration options | | partition options |
▶ | connection options |

```

#### text default information:

```

| | | | | | |
| | CODEPAGE code-page | | LANGUAGE locale | | FORMAT format |
| | |

```

#### update characteristics:

```

| | | |
| | UPDATE FREQUENCY NONE |
| | | update frequency |
| | |

```



## db2ts CREATE INDEX

### **ON** *table-name*

The table name containing the text column. Text search indexes cannot be created on the following tables:

- federated tables
- materialized query tables
- views

### *text-column-name*

The column name of the column to be indexed. The column must be of one of the following data types: CHAR, VARCHAR, CLOB, DBCLOB, BLOB, GRAPHIC, VARGRAPHIC, or XML. If the data type of the column is not one of these data types, use a transformation function specified with *function-schema.function-name* to convert the column type to one of the valid types. Alternatively, you can specify a user-defined external function that accesses the text documents to be indexed. Only a single text search index can be created for a column.

### *function-name(text-column-name)*

Specifies the schema qualified name, conforming to DB2 naming conventions, of an external scalar function that accesses text documents in a column that is not of a supported type for text searching. Performs a data type conversion of that value and returns the value as one of the supported data types for text searching. Its task is to perform a column type conversion. This function must take only one parameter and return only one value.

### **CODEPAGE** *code-page*

Specifies the DB2 code page (CODEPAGE) to be used when indexing text documents. The default value is specified by the value in the view SYSIBMTS.TSDEFAULTS, where DEFAULTNAME='CODEPAGE'; (which happens to be the database code page). This argument only applies to binary data types, such as, the column type or return type from a transformation function must be BLOB or character-type FOR BIT DATA.

### **LANGUAGE** *locale*

Specifies the language to be used by DB2 Text Search for language-specific processing of a document during indexing. If you do not specify a locale, the database territory determines the default setting for **LANGUAGE**. To have your documents automatically scanned to determine the locale, specify *locale* as AUTO.

### **FORMAT** *format*

Specifies the format of text documents in the column. The supported formats include: TEXT, XML, HTML, and INSO. DB2 Text Search needs this information when indexing documents. If the format is not specified, the default value is used. The default value is in the view SYSIBMTS.TSDEFAULTS, where DEFAULTNAME='FORMAT';. For columns of data type XML, the default format 'XML'; is used, regardless of the value of DEFAULTNAME. Rich text support must be installed to use the INSO format.

### **UPDATE FREQUENCY**

Specifies the frequency with which index updates are made. The index is updated if the number of changes is at least the value set for **UPDATE MINIMUM**. The update frequency **NONE** indicates that no further index updates are made. An update frequency of **NONE** can be useful for a text column in a table with data that does not change. It is also useful when the user intends to manually update the index (using the **UPDATE INDEX** command). Automatic updates can only be done if the **START FOR TEXT** command has been run and the DB2 Text Search instance services are running.

The default frequency value is taken from the view SYSIBMTS.TSDEFAULTS, where DEFAULTNAME='UPDATEFREQUENCY';

**NONE**

No further index updates are made. The update has to be started manually.

**D** The days of the week when the index is updated.

- \* Every day of the week.

*integer1*

Specific days of the week, from Sunday to Saturday: 0 - 6

**H** The hours of the specified days when the index is updated.

- \* Every hour of the day.

*integer2*

Specific hours of the day, from midnight to 11 pm: 0 - 23

**M** The minutes of the specified hours when the index is updated.

*integer3*

Specified as top of the hour (0), or in multiples of 5 minute increments after the hour: 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 or 55

**UPDATE MINIMUM** *minchanges*

Specifies the minimum number of changes to text documents before the index is updated incrementally at the time specified in **UPDATE FREQUENCY**. Positive integer values only are allowed. The default value is taken from the view SYSIBMTS.TSDEFAULTS, where DEFAULTNAME='UPDATEMINIMUM'.

**Note:** This value is ignored during an **UPDATE INDEX** command (unless the **USING UPDATE MINIMUM** option is used there). A small value increases consistency between the table column and the text search index. However, it also causes additional load for the system.

**COLLECTION DIRECTORY** *directory*

Specifies the directory in which the text search index collection is stored. You must specify the absolute path, where the maximum length of the absolute path name is 215 characters. The process owner of the Text Search server instance service must have read and write access on this directory.

The **COLLECTION DIRECTORY** parameter is only supported for an integrated text search server setup. Review the usage notes for additional information about collection locations.

**ADMINISTRATION TABLES IN** *tablespace-name*

Specifies the name of an existing non-temporary table space for the administration tables created for the index.

For a non-partitioned databases, if no table space is specified, the table space of the base table for which you are creating the index is used.

For a partitioned database, you must use the **ADMINISTRATION TABLES** parameter explicitly when creating text indexes. The table space must be in the same partition group as the table space of the base table to ensure that the staging tables for the text search index are distributed in the same manner as the corresponding base table.

**INDEX CONFIGURATION** (*option-value*)

Specifies additional index-related values as option value string pairs. The following values are supported:

## db2ts CREATE INDEX

Table 55. Specifications for option-value

Option	Value	Data type	Description
<b>COMMENT</b>	<i>text</i>	String value less than 512 bytes	Adds a string comment value to the REMARKS column in the DB2 Text Search catalog view TSINDEXES. It also adds the string comment value as the description of the collection.
<b>UPDATEAUTOCOMMIT</b>	<i>commitcount_number</i>	Integer	<p>This option specifies the number of index updates after which a commit is executed to automatically preserve the previous work for either initial or incremental updates.</p> <ul style="list-style-type: none"> <li>For initial updates, the index update will process batches of documents from a base table after the trigger to capture data updates is activated. After the number of documents updated reaches the COMMITCOUNT number, the server does an interim commit. Log entries generated by unprocessed documents are removed from the staging table. Using the <b>UPDATEAUTOCOMMIT</b> option for an initial text index update leads to a significant increase of execution time.</li> <li>For incremental updates, log entries which have been processed are removed correspondingly from the staging table with each interim commit. COMMITCOUNT counts the number of documents updated, not the number of staging table entries.</li> </ul>
<b>AUXLOG</b>	<i>auxlog_value</i>	String	Controls the creation of the additional log infrastructure to capture changes that are not recognized by a trigger. The default setting for range partitioned tables is ON, otherwise it is OFF. The default value can be changed in the default table with a setting for AuxLogNorm for non-range-partitioned tables and AuxLogPart for range-partitioned tables.
<b>CJKSEGMENTATION</b>	<i>CJKSEGMENTATION_method</i>	String value less than 512 bytes	<p>Specifies the segmentation method for documents in Chinese, Japanese, and Korean languages (zh_CN, zh_TW, ja_JP, ko_KR locale sets), including such documents when automatic language detection is enabled (LANGUAGE AUTO). If no option is specified, the value for <b>CJKSEGMENTATION</b> in the defaults table is applied. Supported values are:</p> <ul style="list-style-type: none"> <li>MORPHOLOGICAL</li> <li>NGRAM</li> </ul> <p>The specified segmentation method is added to the SYSIBMTS.TSCONFIGURATION administrative view and cannot be changed after the text index is created.</p>

**Remember:** Non-numeric values, like comments, must be enclosed in single quotation marks. A single quote character within a string value must be represented by two consecutive single quotation marks.

**Example:**

```
INDEX CONFIGURATION (COMMENT 'Index on User''s Guide column')
```



**partition options**

Reserved for internal IBM use.

**CONNECT TO** *database-name*

This clause specifies the database to which a connection is established. The database must be on the local system. If specified, this clause takes precedence over the environment variable **DB2DBDFT**. This clause can be omitted if the following are all true:

- The **DB2DBDFT** environment variable is set to a valid database name.
- The user running the command has the required authorization to connect to the database server.

**USER** *username* **USING** *password*

This clause specifies the authorization name and password that are used to establish the connection.

**Usage notes**

All limits and naming conventions, that apply to DB2 database objects and queries, also apply to DB2 Text Search features and queries. DB2 Text related identifiers must conform to the DB2 naming conventions. In addition, there are some additional restrictions. For example, these identifiers can only be of the form:

[A-Za-z][A-Za-z0-9@#\$\_]\*

or

"[A-Za-z ][A-Za-z0-9@#\$\_ ]\*"

With the successful execution of the **CREATE INDEX** command:

- DB2 Text Search server data is updated. A collection of name *instance\_database-name\_index-identifier\_number* is created per database partition, as in the following example:  
tigertail\_MYTSDB\_TS250517\_0000

The collection name can be retrieved from the SYSIBMTS.TSCOLLECTIONNAMES view (column COLLECTIONNAME).

- DB2 Text Search catalog information is updated. An index staging table is created in the specified table space with appropriate DB2 indexes. In addition, an index event table is created in the specified table space. If the `auxlog on` option is set, a second staging table is created to capture changes via integrity processing.
- When DB2 Text Search coexists with DB2 Net Search Extender and an active Net Search Extender index exists for the table column, the new text index is set to inactive.
- The newly created text search index is not automatically populated. The **UPDATE INDEX** command must be executed either manually or automatically (as a result of an update schedule having been defined for the index through the specification of the **UPDATE FREQUENCY** option) for the text search index to be populated.
- If a frequency is specified, a schedule task is created for the DB2 Administrative Scheduler.

Usage restrictions:

## db2ts CREATE INDEX

- A primary key must be defined for the table. In DB2 Text Search, a multi-column DB2 primary key can be used without type limitations. The number of primary key columns is limited to two columns less than the number of primary key columns allowed by DB2.
- The total length of all primary key columns for a table with DB2 Text Search indexes is limited to 15 bytes less than the maximum total primary key length allowed by DB2. Refer to the restrictions of the DB2 CREATE INDEX statement.

Multiple commands cannot be executed concurrently on a text search index if they might conflict. If this command is issued while a conflicting command is running, an error will occur and the command will fail, after which you can try to run the command again. A conflicting command is:

- **DISABLE DATABASE FOR TEXT**

The auxiliary log property for a text index cannot be changed after creating the index.

The following conditions must be satisfied in a partitioned database environment:

- Text search index-specific administration tables, such as staging tables, and text search indexes are distributed in a manner like the corresponding base table. Use the **ADMINISTRATION TABLES IN** clause while creating text search indexes so that the specified table space is in the same partition group as the table space of the base table.

The **CJKSEGMENTATION** option applies to zh\_CN, zh\_TW, ja\_JP and ko\_KR locale sets for Chinese, Japanese, and Korean languages. The MORPHOLOGICAL or NGRAM option specified for the segmentation method is added to the SYSIBMTS.TSCONFIGURATION administration view.

Creating an index with the **LANGUAGE** parameter set to the AUTO option allows **CJKSEGMENTATION** specification as an option. The specified segmentation method applies to Chinese, Japanese, and Korean language documents. You cannot change the value set for the *CJKSEGMENTATION\_method* value once index creation is complete.

If a text search index is created with the **LANGUAGE** parameter set to AUTO and **CJKSEGMENTATION** set to MORPHOLOGICAL, then searches for valid strings on a morphological index might not return the expected results. In such a case, use the QUERYLANGUAGE option of the **CONTAINS** function to obtain the results, as shown in the following sample statement:

```
select bookname from ngrambooks where contains (story, '△△', 'QUERYLANGUAGE=zh_CN') = 1
```

## db2ts DISABLE DATABASE FOR TEXT

This command reverses the changes (for example, drops the text-search related tables and view) made by the command **ENABLE DATABASE FOR TEXT**.

When issued, this command:

- Disables the DB2 Text Search feature for the database
- Drops text search catalog tables and views and related database objects
- If the **FORCE** option is specified, all text index information is removed from the database and all associated collections are deleted. See the “db2ts DROP INDEX command” for reference.

For execution, the command needs to be prefixed with **db2ts** at the command line.

### Authorization

The privileges held by the authorization ID of the statement must include both of the following authorities:

- DBADM with DATAACCESS authority.
- SYSTS\_ADM role

### Required connection

Database

### Command syntax

```

▶▶—DISABLE DATABASE FOR TEXT—FORCE—| connection options |▶▶

```

### connection options:

```

|———|
|—CONNECT TO—database-name—|
|———|
|—USER—username—USING—password—|
|———|

```

### Command parameters

#### FORCE

Specifies that all text search indexes be forcibly dropped from the database.

If this option is not specified and text search indexes are defined for this database, the command will fail.

If this option is specified and DB2 Text Search service has not been started (the db2ts **START FOR TEXT** command has not been issued), the text search indexes (collections) are not dropped and need to be cleaned up manually with the **db2ts CLEANUP** command.

#### CONNECT TO *database-name*

This clause specifies the database to which a connection will be established. The database must be on the local system. If specified, this clause takes precedence over the environment variable DB2DBDFT. This clause can be omitted if the following are all true:

- The DB2DBDFT environment variable is set to a valid database name.

## db2ts DISABLE DATABASE FOR TEXT

- The user running the command has the required authorization to connect to the database server.

**USER** *username* **USING** *password*

This clause specifies the authorization name and password that will be used to establish the connection.

### Usage notes

This command does not influence the DB2 Net Search Extender enablement status of the database. It deletes the DB2 Text Search catalog tables and views that are created by the **ENABLE FOR TEXT** command.

Before dropping a DB2 database that has text search index definitions, issue this command and make sure that the text indexes and collections have been removed successfully.

If some indexes could not be deleted using the **FORCE** option, the collection names are written to the **db2diag** log file.

**Note:** The user is discouraged from usage that results in orphaned collections, such as, collections that remain defined on the text search server but are not used by DB2. Here are some cases that cause orphaned collections:

- When a **DROP DATABASE CLP** command is executed without running a **DISABLE DATABASE FOR TEXT** command
- When a **DISABLE DATABASE FOR TEXT** command is executed using the **FORCE** option.
- Some other error conditions.

Multiple commands cannot be executed concurrently on a text search index if they may conflict. If this command is issued while a conflicting command is running, an error will occur and the command will fail, after which you can try to run the command again. Some of the conflicting commands are:

- **DROP INDEX**
- **UPDATE INDEX**
- **CLEAR EVENTS FOR INDEX**
- **ALTER INDEX**
- **DISABLE DATABASE FOR TEXT**



### Usage notes

Multiple commands cannot be executed concurrently on a text search index if the command might conflict. If this command is issued while a conflicting command is running, an error occurs and the command fails, after which you can try to run the command again. The following commands are some common conflicting commands:

- **DROP INDEX**
- **UPDATE INDEX**
- **CLEAR EVENTS FOR INDEX**
- **ALTER INDEX**
- **DISABLE DATABASE FOR TEXT**

A **STOP FOR TEXT** command that runs in parallel with the **DROP** operation will not cause a conflicting command message, instead, if the text search server is shut down before **DROP** has removed the collection, an error will be returned that the text search server is not available.

After a text search index is dropped, text search is no longer possible on the corresponding text column. If you plan to create a new text search on the same text column, you must first disconnect from the database and then reconnect before creating the new text search index.

The **db2ts DROP INDEX FOR TEXT** command makes the following changes to the database:

- Updates the DB2 Text Search catalog information.
- Drops the index staging and event tables.
- Deletes triggers on the user text table.
- Destroys the collection associated with the DB2 Text Search index definition.

## db2ts ENABLE DATABASE FOR TEXT

The **db2ts ENABLE DATABASE FOR TEXT** command enables DB2 Text Search for the current database. It creates administrative tables and views, sets default values for parameters, and must run successfully before you can create text search indexes on columns in tables within the database. The command needs to be prefixed with **db2ts** at the command line.

After enabling the database, it is necessary to specify the connection information for the text search server in the `SYSIBMTS.TSSERVERS` view. The enable operation includes an attempt to populate the server data and will show a warning if the server configuration cannot be accessed. In any case, it is recommended to verify the connection information in the view. For details, see the topic about updating DB2 Text Search server information.

### Authorization

- The privileges held by the authorization ID of the statement must include the `SYSTS_ADM` role and the `DBADM` authority.

### Required connection

Database

### Command syntax

```

▶▶—ENABLE DATABASE FOR TEXT—————▶▶
▶——▶
|  ADMINISTRATION TABLES IN tablespace-name  |
|_____|
▶——▶
|  AUTOGRANT | connection options |
|_____|

```

### connection options:

```

|_____|
|  CONNECT TO—database-name—
|_____|
|  USER—username—USING—password—
|_____|

```

### Command parameters

#### ADMINISTRATION TABLES IN *tablespace-name*

Specifies the name of an existing regular table space for administration tables created while enabling the database for DB2 Text Search. It is recommended that the table space is in the database partition group `IBMCATGROUP`. For a partitioned database, the bufferpool and table space should be defined with 32 KB page size.

If the clause is not specified, `SYSTOOLSPACE` is used as default table space. In this case, ensure that `SYSTOOLSPACE` already exists. If it does not exist, the `SYSPROC.SYSINSTALLOBJECTS` procedure may be used to create it.

**Note:** Use quotation marks to specify a case-sensitive table space name.

## db2ts ENABLE DATABASE FOR TEXT

### AUTOGRANT

This option has been deprecated and does not grant privileges to the instance owner anymore. Its use is no longer suggested and might be removed in a future release.

### CONNECT TO *database-name*

This clause specifies the database to which a connection is established. The database must be on the local system. If specified, this clause takes precedence over the environment variable DB2DBDFT. This clause can be omitted if the following statements are all true:

- The DB2DBDFT environment variable is set to a valid database name.
- The user running the command has the required authorization to connect to the database server.

### USER *username* USING *password*

This clause specifies the authorization name and password used to establish the connection.

## Example

Example 1: Enable a database for DB2 Text Search creating administration tables in table space named tsspace and return any error messages in English.

```
CALL SYSPROC.SYSTS_ENABLE('ADMINISTRATION TABLES IN tsspace', 'en_US', ?)
```

The following is an example of output from this query.

```
Value of output parameters
-----
Parameter Name : MESSAGE
Parameter Value : Operation completed successfully.

Return Status = 0
```

## Usage notes

When executed successfully, this command does the following actions:

- Enables the DB2 Text Search feature for the database.
- Establishes DB2 Text Search database configuration default values in the view SYSIBMTS.TSDEFAULTS.
- Creates the following DB2 Text Search administrative views in the SYSIBMTS schema:
  - SYSIBMTS.TSDEFAULTS
  - SYSIBMTS.TSLOCKS
  - SYSIBMTS.TSINDEXES
  - SYSIBMTS.TSCONFIGURATION
  - SYSIBMTS.TSCOLLECTIONNAMES
  - SYSIBMTS.TSSERVERS



## db2ts HELP

**db2ts HELP** displays the list of available DB2 Text Search commands, or the syntax of an individual command.

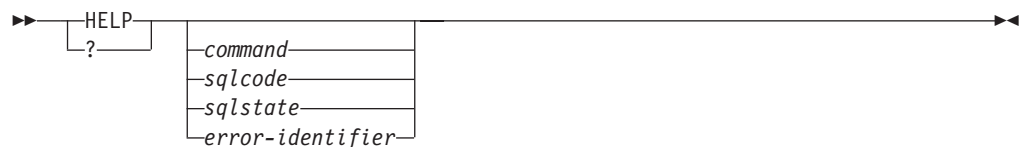
Use the **db2ts HELP** command to get help on specific error messages as well.

For execution, the command needs to be prefixed with **db2ts** at the command line.

### Authorization

None.

### Command syntax



### Command parameters

#### HELP | ?

Provides help information for a command or a reason code.

#### *command*

The first keywords that identify a DB2 Text Search command:

- ALTER
- CLEANUP
- CLEAR (for both CLEAR COMMAND LOCKS and CLEAR EVENTS FOR INDEX)
- CREATE
- DISABLE
- DROP
- ENABLE
- RESET PENDING
- START
- STOP
- UPDATE

*sqlcode* SQLCODE for message returned by db2ts command (within or outside the administration stored procedure) or text search query.

*sqlstate* Sqlstate returned by command, administration stored procedure, or text search query.

#### *error-identifier*

An identifier is part of the *text-search-error-msg* that is embedded in error messages. This identifier starts with 'CIE' and is of the form CIE $nnnnn$  where  $nnnnn$  is a number. This identifier represents the specific error that is returned upon an error during text search. It may also be returned in an informational message upon completion of a text search command or in the message printed at the completion of a text search administration procedure. If the identifier does not start with 'CIE', then **db2ts help**

## db2ts HELP

cannot provide information about the *error-identifier*. For example, db2ts cannot provide help for a message with an *error-identifier* such as IQQR0012E.

### Usage notes

When using a UNIX shell, it might be necessary to supply the arguments to **db2ts** using double quotation marks, as in the following example:

```
db2ts "? CIE00323"
```

Without the quotation marks, the shell tries to match the wildcard with the contents of the working directory and it may give unexpected results.

If the first keyword of any db2ts command is specified, the syntax of the identified command is displayed. For the two db2ts commands that share the same first keyword (**CLEAR COMMAND LOCKS** and **CLEAR EVENTS FOR INDEX**), the syntax of both commands will be displayed when db2ts help clear is issued, but each command may be specifically displayed by adding the second keyword to distinguish them, for example db2ts help clear events. If a parameter is not specified after **?** or **HELP**, db2ts lists all available db2ts commands.

Specifying a *sqlcode*, *sqlstate*, or CIE *error-identifier* will return information about that code, state, or error identifier. For example,

```
db2ts help SQL20423
```

or

```
db2ts ? 38H10
```

or

```
db2ts ? CIE00323
```

## db2ts START FOR TEXT

The **db2ts START FOR TEXT** command starts the DB2 Text Search instance services that support other DB2 Text Search administration commands and the ability to reference text search indexes in SQL queries.

The **db2ts START FOR TEXT** command also includes starting processes for rich text support on the host machine running the DB2 Text Search server, if the server is configured for rich text support.

This command must be issued from the DB2 database server.

To start instance services in a partitioned database environment using an integrated text search setup, you must run the command on the integrated text search server host machine. By default, the integrated text search server host machine is the host of the lowest-numbered database partition server.

### Authorization

Instance owner. No database privilege is required

### Command syntax

```

▶▶—START FOR TEXT—┌STATUS┐
                   └VERIFY┘
  
```

### Command parameters

#### STATUS

Verifies the status of the DB2 Text Search server. A verbose informational message is returned indicating the STARTED or STOPPED status of the server.

#### VERIFY

Verifies the started status of the DB2 Text Search server and exits with a standard message and return code 0 indicating that the operation is successful. A non-zero code is returned for any other state of the text search server or if the status cannot be verified.

### Examples

- Check that the text search server is started.

```

Linux/UNIX:
$ db2ts START FOR TEXT VERIFY
CIE00001 Operation completed successfully.
  
```

```

$ echo $?
0
  
```

```

Windows:
C:\> db2ts START FOR TEXT VERIFY
CIE00001 Operation completed successfully.
  
```

```

C:\> echo %ERRORLEVEL%
0
  
```

### Usage notes

- In a partitioned database environment, the **db2ts START FOR TEXT** command with the **STATUS** and **VERIFY** parameters can be issued on any one of the partition

## db2ts START FOR TEXT

server hosts. To start the instance services, you must run the **db2ts START FOR TEXT** command on the integrated text search server host machine. The integrated text search server host machine is the host of the lowest-numbered database partition server. If custom collection directories are used, ensure that no lower numbered partitions are created later. This restriction is especially relevant for Linux and UNIX platforms. If you configure DB2 Text Search when creating an instance, the configuration initially determines the integrated text search server host. That configuration must always remain the host of the lowest-numbered database partition server.

- On Windows platforms, there is a Windows service associated with each DB2 instance for DB2 Text Search. The service name can be determined by issuing the following command:

```
DB2TS - <instance name>[-<partition number>]
```

. Apart from the using the **db2ts START FOR TEXT** command, you can also start the service using the Control Panel or the **NET START** command.

## db2ts STOP FOR TEXT

The **db2ts STOP FOR TEXT** command stops DB2 Text Search instance services. If the running services include processes for rich text support then those services are stopped as well.

This command must be issued from the DB2 database server.

When running this command from the command line, prefix the command with db2ts at the DB2 command line.

This command provides the convenience of stopping a stand-alone text search server which can also be achieved in its own installation environment using the provided script. If the instance services are already stopped, the command only checks and reports its status to the user.

### Authorization

Instance owner. No database privilege is required

### Command syntax

```

▶▶ STOP FOR TEXT [STATUS] [VERIFY]

```

### Command parameters

#### STATUS

Verifies the status of the DB2 Text Search servers. A verbose informational message is returned indicating the STARTED or STOPPED status of the servers.

#### VERIFY

Verifies the stopped status of the DB2 Text Search server. It exits with the standard message and return code 0 to indicate the command ran successfully. Otherwise, the text search server returns a non-zero code to indicate failure.

### Usage notes

- To avoid interrupting the execution of currently running commands, ensure no other administrative commands like the **db2ts UPDATE INDEX FOR TEXT** command are still active before issuing the **db2ts STOP FOR TEXT** command.
- In a partitioned database environment, the **db2ts START FOR TEXT** command with the **STATUS** and **VERIFY** parameters can be issued on any one of the partition server hosts.
- In a partitioned database environment on Windows platforms using an integrated text search server, stop the instance services by issuing the **db2ts STOP FOR TEXT** command on the integrated text search server host machine. By default, the integrated text search server host machine is the host of the lowest-numbered database partition server. Running the command on the integrated text search server host machine ensures that all processes and services are stopped. If the command is run on a different partition server host, the DB2TS service must be stopped separately using a command such as NET STOP.

## db2ts UPDATE INDEX

The **db2ts UPDATE INDEX** command updates the text search index (collection in DB2 Text Search) to reflect the current contents of the text column with which the index is associated. While the update is being performed, a search is possible. Until completion of the update, the search operates on a partially updated index.

For execution, the command needs to be prefixed with **db2ts** at the command line.

### Authorization

The privileges held by the authorization ID of the statement must include the SYSTS\_MGR role and at least one of the following authorities:

- DATAACCESS authority
- CONTROL privilege on the table on which the text index is defined
- INDEX with SELECT privilege on the base table on which the text index is defined

In addition, for an initial update the authorization requirements apply as outlined in the **CREATE TRIGGER** statement.

### Required connection

Database

### Command syntax

```

>> UPDATE INDEX index-name FOR TEXT UPDATE OPTIONS
connection options

```

#### connection options:

```

CONNECT TO database-name USER username USING password

```

### Command parameters

#### UPDATE INDEX *index-name*

Specifies the name of the text search index to be updated. The index name must adhere to the naming restrictions for DB2 indexes.

#### UPDATE OPTIONS

An input argument of type VARCHAR(32K) that specifies update options. If no options are specified the update is started unconditionally. The possible values are:

UPDATE OPTIONS value	Description
USING UPDATE MINIMUM	This option enforces the use of the UPDATE MINIMUM value that is defined for the text search index and processes updates if the specified minimum number of changes have occurred.

UPDATE OPTIONS value	Description
FOR DATA REDISTRIBUTION	This option specifies that a text search index in a partitioned database needs to be refreshed after data partitions were added or removed and a subsequent data redistribution operation completed. Search results may be inconsistent until the text search index was updated with the FOR DATA REDISTRIBUTION option.
ALLROWS	This option specifies that an initial update should be attempted unconditionally.
UPDATEAUTOCOMMIT	<p>This option specifies the number of index updates after which a commit is executed to preserve the previous work automatically for either initial or incremental updates.</p> <ul style="list-style-type: none"> <li>For initial updates, the index update will process batches of documents from a base table after the trigger to capture data updates is activated. After the number of documents updated reaches the COMMITCOUNT number, the server does an interim commit. Log entries generated by unprocessed documents are removed from the staging table. Using the UPDATEAUTOCOMMIT option for an initial text index update leads to a significant increase of execution time.</li> <li>For incremental updates, log entries which have been processed are removed correspondingly from staging table with each interim commit. COMMITCOUNT counts the number of documents updated, not the number of staging table entries.</li> </ul>

**CONNECT TO** *database-name*

This clause specifies the database to which a connection is established. The database must be on the local system. If specified, this clause takes precedence over the environment variable DB2DBDFT. This clause can be omitted if the following statements are all true:

- The DB2DBDFT environment variable is set to a valid database name.
- The user running the command has the required authorization to connect to the database server.

**USER** *username* **USING** *password*

This clause specifies the authorization name and password that are used to establish the connection.

**Usage notes**

All limits and naming conventions, that apply to DB2 database objects and queries, also apply to DB2 Text Search features and queries. DB2 Text Search related identifiers must conform to the DB2 naming conventions. In addition, there are some additional restrictions. For example, these identifiers can only be of the form:

[A-Za-z][A-Za-z0-9@#\$\_]\*

## db2ts UPDATE INDEX

or

```
"[A-Za-z ] [A-Za-z0-9@#$_ ]*"
```

If synonym dictionaries have been created for a text index, executing the ALLROWS and FOR DATA REDISTRIBUTION update options removes dictionaries from existing collections. There can be new collections associated with the text index after database partitions are added. The synonym dictionaries for all associated collections have to be added again.

This command does not return until all index update processing is completed. The duration depends on the number of documents to be indexed and the number of documents already indexed. The collection name for the index can be retrieved from the SYSIBMTS.TSCOLLECTIONNAMES view (column COLLECTIONNAME).

Multiple commands cannot be executed concurrently on a text search index if they might conflict. If you run this command while a conflicting command is running, an error occurs and the command fails, after which you can try to run the command again. The following commands are some of the common conflicting commands:

- **UPDATE INDEX**
- **CLEAR EVENTS FOR INDEX**
- **ALTER INDEX**
- **DROP INDEX**
- **DISABLE DATABASE FOR TEXT**

**Note:** In cases of individual document errors, the documents must be corrected. The primary keys of the erroneous documents can be looked up in the event table for the index. The next **UPDATE INDEX** command reprocesses these documents if the corresponding rows in the user table are modified.

Changes to the database:

- Insert rows to the event table (including parser error information from DB2 Text Search).
- Delete from the index staging table in case of incremental updates.
- Before first update, create triggers on the user text table.
- The collection is updated.
- New or changed documents are parsed and indexed.
- Deleted documents are discarded from the index.



---

## Appendix A. Naming conventions

---

### Naming conventions

The naming conventions for naming database manager objects, such as databases and tables.

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) which, as a communications protocol, is no longer supported. 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 previously.

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 bytes in length. Column names can be 1 to 128 bytes 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 128 bytes in length.
- Group IDs cannot exceed 128 bytes in length.

---

## **Appendix B. File type modifiers and delimiters**

### File type modifiers for the export, import and load utilities

The links below will take you to the top of a command topic, where you'll then find a quick link to the respective file type modifier section.

[“EXPORT” on page 182](#)

[“IMPORT” on page 265](#)

[“LOAD” on page 387](#)

## Delimiter considerations for moving data

When moving delimited ASCII (DEL) files, it is important to ensure that the data being moved is not unintentionally altered because of problems with delimiter character recognition. To help prevent these errors, DB2 enforces several restrictions and provides a number of file type modifiers.

### Delimiter restrictions

There are a number of restrictions in place that help prevent the chosen delimiter character from being treated as a part of the data being moved. First, delimiters are mutually exclusive. Second, a delimiter cannot be binary zero, a line-feed character, a carriage-return, or a blank space. As well, the default decimal point (.) cannot be a string delimiter. Finally, in a DBCS environment, the pipe (|) character delimiter is not supported.

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:
  - string delimiter: "(0x22, double quotation mark)
  - column delimiter: ,(0x2c, comma)
- Default delimiters for data files in EBCDIC SBCS code pages are:
  - string delimiter: "(0x7F, double quotation mark)
  - column delimiter: ,(0x6B, comma)
- 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 ...
```

### Issues with delimiters during data movement

#### Double character delimiters

By default, for character-based fields of a DEL file, any instance of the character delimiter found within the field is represented by double character delimiters. For example, assuming that the character delimiter is the double quote, if you export the text I am 6" tall., the output text in the DEL file reads "I am 6"" tall." Conversely, if the input text in a DEL file reads "What a ""nice"" day!", the text is imported as What a "nice" day!

#### **nodouble1**

Double character delimiter behavior can be disabled for the import, export, and load utilities by specifying the `nodouble1` file type modifier. However, be aware that double character delimiter behavior exists in order to avoid parsing errors. When you use `nodouble1` with export, the character delimiter is not doubled if it is present in

## Delimiter considerations for moving data

character fields. When you use `nodoubledel` with `import` and `load`, the double character delimiter is not interpreted as a literal instance of the character delimiter.

### **nochardel**

When you use the `nochardel` file type modifier with `export`, the character fields are not surrounded by character delimiters. When `nochardel` is used `import` and `load`, the character delimiters are not treated as special characters and are interpreted as actual data.

### **chardel**

Other file type modifiers can be used to manually prevent confusion between default delimiters and the data. The `chardel` file type modifier specifies `x`, a single character, as the character string delimiter to be used instead of double quotation marks (as is the default).

### **coldel**

Similarly, if you wanted to avoid using the default comma as a column delimiter, you could use `coldel`, which specifies `x`, a single character, as the column data delimiter.

### **delprioritychar**

Another concern in regards to moving DEL files is maintaining the correct precedence order for delimiters. The default priority for delimiters is: row, character, column. However, some applications depend on the priority: character, row, column. For example, using the default priority, the DEL data file:

```
"Vincent <row delimiter> is a manager",<row delimiter>
```

would be interpreted as having two rows: Vincent, and is a manager, since `<row delimiter>` takes precedence over the character delimiter (`"`). Using `delprioritychar` gives the character delimiter (`"`) precedence over the row delimiter (`<row delimiter>`), meaning that the same DEL file would be interpreted (correctly) as having one row: Vincent is a manager.

---

## Appendix C. Overview of the DB2 technical information

DB2 technical information is available in multiple formats that can be accessed in multiple ways.

DB2 technical information is available through the following tools and methods:

- DB2 Information Center
  - Topics (Task, concept and reference topics)
  - Sample programs
  - Tutorials
- DB2 books
  - PDF files (downloadable)
  - PDF files (from the DB2 PDF DVD)
  - printed books
- Command-line help
  - Command help
  - Message help

**Note:** The DB2 Information Center topics are updated more frequently than either the PDF or the hardcopy 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](http://ibm.com).

You can access additional DB2 technical information such as technotes, white papers, and IBM Redbooks® publications online at [ibm.com](http://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 to improve the DB2 documentation, send an email to [db2docs@ca.ibm.com](mailto: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 email 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.

---

### 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/e-business/linkweb/publications/servlet/pbi.wss](http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss). English and translated DB2 Version 10.1 manuals in PDF format can be downloaded from [www.ibm.com/support/docview.wss?rs=71&uid=swg2700947](http://www.ibm.com/support/docview.wss?rs=71&uid=swg2700947).

Although the tables identify books available in print, the books might not be available in your country or region.

The form number increases each time a manual is updated. Ensure that you are reading the most recent version of the manuals, as listed below.

**Note:** The *DB2 Information Center* is updated more frequently than either the PDF or the hard-copy books.

*Table 56. DB2 technical information*

Name	Form Number	Available in print	Last updated
<i>Administrative API Reference</i>	SC27-3864-00	Yes	April, 2012
<i>Administrative Routines and Views</i>	SC27-3865-00	No	April, 2012
<i>Call Level Interface Guide and Reference Volume 1</i>	SC27-3866-00	Yes	April, 2012
<i>Call Level Interface Guide and Reference Volume 2</i>	SC27-3867-00	Yes	April, 2012
<i>Command Reference</i>	SC27-3868-00	Yes	April, 2012
<i>Database Administration Concepts and Configuration Reference</i>	SC27-3871-00	Yes	April, 2012
<i>Data Movement Utilities Guide and Reference</i>	SC27-3869-00	Yes	April, 2012
<i>Database Monitoring Guide and Reference</i>	SC27-3887-00	Yes	April, 2012
<i>Data Recovery and High Availability Guide and Reference</i>	SC27-3870-00	Yes	April, 2012
<i>Database Security Guide</i>	SC27-3872-00	Yes	April, 2012
<i>DB2 Workload Management Guide and Reference</i>	SC27-3891-00	Yes	April, 2012
<i>Developing ADO.NET and OLE DB Applications</i>	SC27-3873-00	Yes	April, 2012
<i>Developing Embedded SQL Applications</i>	SC27-3874-00	Yes	April, 2012
<i>Developing Java Applications</i>	SC27-3875-00	Yes	April, 2012



Table 56. DB2 technical information (continued)

<b>Name</b>	<b>Form Number</b>	<b>Available in print</b>	<b>Last updated</b>
<i>Developing Perl, PHP, Python, and Ruby on Rails Applications</i>	SC27-3876-00	No	April, 2012
<i>Developing User-defined Routines (SQL and External)</i>	SC27-3877-00	Yes	April, 2012
<i>Getting Started with Database Application Development</i>	GI13-2046-00	Yes	April, 2012
<i>Getting Started with DB2 Installation and Administration on Linux and Windows</i>	GI13-2047-00	Yes	April, 2012
<i>Globalization Guide</i>	SC27-3878-00	Yes	April, 2012
<i>Installing DB2 Servers</i>	GC27-3884-00	Yes	April, 2012
<i>Installing IBM Data Server Clients</i>	GC27-3883-00	No	April, 2012
<i>Message Reference Volume 1</i>	SC27-3879-00	No	April, 2012
<i>Message Reference Volume 2</i>	SC27-3880-00	No	April, 2012
<i>Net Search Extender Administration and User's Guide</i>	SC27-3895-00	No	April, 2012
<i>Partitioning and Clustering Guide</i>	SC27-3882-00	Yes	April, 2012
<i>pureXML Guide</i>	SC27-3892-00	Yes	April, 2012
<i>Spatial Extender User's Guide and Reference</i>	SC27-3894-00	No	April, 2012
<i>SQL Procedural Languages: Application Enablement and Support</i>	SC27-3896-00	Yes	April, 2012
<i>SQL Reference Volume 1</i>	SC27-3885-00	Yes	April, 2012
<i>SQL Reference Volume 2</i>	SC27-3886-00	Yes	April, 2012
<i>Text Search Guide</i>	SC27-3888-00	Yes	April, 2012
<i>Troubleshooting and Tuning Database Performance</i>	SC27-3889-00	Yes	April, 2012
<i>Upgrading to DB2 Version 10.1</i>	SC27-3881-00	Yes	April, 2012
<i>What's New for DB2 Version 10.1</i>	SC27-3890-00	Yes	April, 2012
<i>XQuery Reference</i>	SC27-3893-00	No	April, 2012

## DB2 technical library in hardcopy or PDF format

Table 57. DB2 Connect-specific technical information

Name	Form Number	Available in print	Last updated
DB2 Connect Installing and Configuring DB2 Connect Personal Edition	SC27-3861-00	Yes	April, 2012
DB2 Connect Installing and Configuring DB2 Connect Servers	SC27-3862-00	Yes	April, 2012
DB2 Connect User's Guide	SC27-3863-00	Yes	April, 2012

---

## Displaying SQL state help from the command line processor

DB2 products return an SQLSTATE value for conditions that can be the result of an SQL statement. SQLSTATE help explains the meanings of SQL states and SQL state class codes.

### Procedure

To start 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.

---

## Accessing different versions of the DB2 Information Center

Documentation for other versions of DB2 products is found in separate information centers on [ibm.com](http://ibm.com)<sup>®</sup>.

### About this task

For DB2 Version 10.1 topics, the *DB2 Information Center* URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v10r1>.

For DB2 Version 9.8 topics, the *DB2 Information Center* URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/>.

For DB2 Version 9.7 topics, the *DB2 Information Center* URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/>.

For DB2 Version 9.5 topics, the *DB2 Information Center* URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9r5>.

For DB2 Version 9.1 topics, the *DB2 Information Center* URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9/>.

For DB2 Version 8 topics, go to the *DB2 Information Center* URL at: <http://publib.boulder.ibm.com/infocenter/db2luw/v8/>.

### Updating the DB2 Information Center installed on your computer or intranet server

A locally installed DB2 Information Center must be updated periodically.

#### Before you begin

A DB2 Version 10.1 Information Center must already be installed. For details, see the “Installing the DB2 Information Center using the DB2 Setup wizard” topic in *Installing DB2 Servers*. All prerequisites and restrictions that applied to installing the Information Center also apply to updating the Information Center.

#### About this task

An existing DB2 Information Center can be updated automatically or manually:

- Automatic updates update existing Information Center features and languages. One benefit of automatic updates is that the Information Center is unavailable for a shorter time compared to during a manual update. In addition, automatic updates can be set to run as part of other batch jobs that run periodically.
- Manual updates can be used to update existing Information Center features and languages. Automatic updates reduce the downtime during the update process, however you must use the manual process when you want to add features or languages. For example, a local Information Center was originally installed with both English and French languages, and now you want to also install the German language; a manual update will install German, as well as, update the existing Information Center features and languages. However, a manual update requires you to manually stop, update, and restart the Information Center. The Information Center is unavailable during the entire update process. In the automatic update process the Information Center incurs an outage to restart the Information Center after the update only.

This topic details the process for automatic updates. For manual update instructions, see the “Manually updating the DB2 Information Center installed on your computer or intranet server” topic.

#### Procedure

To automatically update the DB2 Information Center installed on your computer or intranet server:

1. On Linux operating systems,
  - a. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the `/opt/ibm/db2ic/V10.1` directory.
  - b. Navigate from the installation directory to the `doc/bin` directory.
  - c. Run the `update-ic` script:  
`update-ic`
2. On Windows operating systems,
  - 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 `<Program Files>\IBM\DB2 Information Center\Version 10.1` directory, where `<Program Files>` represents the location of the Program Files directory.
  - c. Navigate from the installation directory to the `doc\bin` directory.

## Updating the DB2 Information Center installed on your computer or intranet server

- d. Run the update-ic.bat file:  
update-ic.bat

### Results

The DB2 Information Center restarts automatically. If updates were available, the Information Center displays the new and updated topics. If Information Center updates were not available, a message is added to the log. The log file is located in `doc\ eclipse\ configuration` directory. The log file name is a randomly generated number. For example, `1239053440785.log`.

---

## Manually updating the DB2 Information Center installed on your computer or intranet server

If you have installed the DB2 Information Center locally, you can obtain and install documentation updates from IBM.

### About this task

Updating your locally installed *DB2 Information Center* manually 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 apply updates. The Workstation version of the DB2 Information Center always runs in stand-alone mode. .
2. Use the Update feature to see what updates are available. If there are updates that you must install, you can use the Update feature to obtain and install them

**Note:** If your environment requires installing the *DB2 Information Center* updates on a machine that is not connected to the internet, mirror the update site to a local file system by using a machine that is connected to the internet and has the *DB2 Information Center* installed. If many users on your network will be installing the documentation updates, you can reduce the time required for individuals to perform the updates by also mirroring the update site locally and creating a proxy for the update site.

If update packages are available, use the Update feature to get the packages. However, the Update feature is only available in stand-alone mode.

3. Stop the stand-alone Information Center, and restart the *DB2 Information Center* on your computer.

**Note:** On Windows 2008, Windows Vista (and higher), the commands listed later in this section must be run as an administrator. To open a command prompt or graphical tool with full administrator privileges, right-click the shortcut and then select **Run as administrator**.

### Procedure

To update the *DB2 Information Center* installed on your computer or intranet server:

1. Stop the *DB2 Information Center*.
  - On Windows, click **Start > Control Panel > Administrative Tools > Services**. Then right-click **DB2 Information Center** service and select **Stop**.
  - On Linux, enter the following command:  
`/etc/init.d/db2icdv10 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 *Program\_Files\IBM\DB2 Information Center\Version 10.1* directory, where *Program\_Files* represents the location of the Program Files directory.
    - c. Navigate from the installation directory to the *doc\bin* directory.
    - d. Run the *help\_start.bat* file:

```
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/V10.1* directory.
    - b. Navigate from the installation directory to the *doc/bin* directory.
    - c. Run the *help\_start* script:

```
help_start
```

The systems default Web browser opens to display the stand-alone Information Center.

3. Click the **Update** button (🔄). (JavaScript must be enabled in your browser.) On the right panel of the Information Center, click **Find Updates**. A list of updates for existing documentation displays.
4. To initiate the installation process, check that the selections you want to install, then click **Install Updates**.
5. After the installation process has completed, click **Finish**.
6. Stop the stand-alone Information Center:
  - On Windows, navigate to the *doc\bin* directory within the installation directory, and run the *help\_end.bat* file:

```
help_end.bat
```

**Note:** The *help\_end* batch file contains the commands required to safely stop the processes that were started with the *help\_start* batch file. Do not use Ctrl-C or any other method to stop *help\_start.bat*.
  - On Linux, navigate to the *doc/bin* directory within the installation directory, and run the *help\_end* script:

```
help_end
```

**Note:** The *help\_end* script contains the commands required to safely stop the processes that were started with the *help\_start* script. Do not use any other method to stop the *help\_start* script.
7. Restart the *DB2 Information Center*.
  - On Windows, click **Start > Control Panel > Administrative Tools > Services**. Then right-click **DB2 Information Center** service and select **Start**.
  - On Linux, enter the following command:

```
/etc/init.d/db2icdv10 start
```

## Results

The updated *DB2 Information Center* displays the new and updated topics.

### DB2 tutorials

The DB2 tutorials help you learn about various aspects of DB2 database 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/db2luw/v10r1/>.

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 the title.

##### **"pureXML®" in pureXML Guide**

Set up a DB2 database to store XML data and to perform basic operations with the native XML data store.

---

### DB2 troubleshooting information

A wide variety of troubleshooting and problem determination information is available to assist you in using DB2 database products.

#### DB2 documentation

Troubleshooting information can be found in the *Troubleshooting and Tuning Database Performance* or the Database fundamentals section of the *DB2 Information Center*, which contains:

- Information about how to isolate and identify problems with DB2 diagnostic tools and utilities.
- Solutions to some of the most common problem.
- Advice to help solve other problems you might encounter with your DB2 database products.

#### IBM Support Portal

See the IBM Support Portal 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 IBM Support Portal at [http://www.ibm.com/support/entry/portal/Overview/Software/Information\\_Management/DB2\\_for\\_Linux,\\_UNIX\\_and\\_Windows](http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows)

---

### Terms and conditions

Permissions for the use of these publications are granted subject to the following terms and conditions.

**Applicability:** These terms and conditions are in addition to any terms of use for the IBM website.

**Personal use:** You may reproduce these publications for your personal, noncommercial 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.

**Rights:** 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.

**IBM Trademarks:** IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)





---

## Appendix D. Notices

This information was developed for products and services offered in the U.S.A. Information about non-IBM products is based on information available at the time of first publication of this document and is subject to change.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about 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 grant 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 character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan, Ltd.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 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, changes, or both in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to websites not owned by IBM are provided for convenience only and do not in any manner serve as an endorsement of those

## Notices

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

Licenses 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  
U59/3600  
3600 Steeles Avenue East  
Markham, Ontario L3R 9Z7  
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 contains 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample 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**

IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

The following terms are trademarks or registered trademarks of other companies

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle, its affiliates, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Intel, Intel logo, Intel Inside, Intel Inside logo, Celeron, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.



---

# Index

## Special characters

- / command 680
- . command 678
- @ command 681
- @@ command 682

## A

- abnormal termination
  - RESTART DATABASE command 544
- ACCEPT command 683
- access paths
  - optimizing 582
- action precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- ACTIVATE DATABASE command
  - details 64
- Active Directory
  - schema extension command 1156
- ADD CONTACT command
  - without using ADMIN\_CMD 66
- ADD CONTACTGROUP command
  - without using ADMIN\_CMD 67
- ADD DBPARTITIONNUM command 68
- ADD XMLSCHEMA DOCUMENT command 71
- ALTER INDEX Text Search command 1244
- anyorder file type modifier 387
- APPEND command 685
- ARCHIVE LOG command 73
- ASC import file type 265
- ATTACH command
  - details 76
- audit facility
  - db2audit command 797
- AUTOCONFIGURE command
  - without using ADMIN\_CMD 78
- autostart DAS command 760
- autostart instance command 942

## B

- BACKUP DATABASE command
  - without using ADMIN\_CMD 81
- backups
  - snapshot backup objects 775
- Benchmark Tool command 806
- binarynumerics file type modifier
  - LOAD command 387
- BIND command
  - details 90
- bind files
  - description tool command 816
- bind variables
  - CLPPlus 49
- bindfile precompile option 440
- binding
  - errors 134
  - implicitly created schemas 90, 440

- blocking precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- BREAK command
  - description 686
- BTITLEcommand
  - description 687

## C

- call level interface (CLI)
  - configuration
    - retrieving 202
- CALL statement
  - CLP 21
- case sensitivity
  - commands 2
  - naming conventions 1276
- CATALOG DATABASE command
  - details 110
- CATALOG DCS DATABASE command 113
- CATALOG LDAP DATABASE command 115
- CATALOG LDAP NODE command 118
- CATALOG LOCAL NODE command 119
- CATALOG NAMED PIPE NODE command 121
- CATALOG ODBC DATA SOURCE command 123
- CATALOG TCPIP/TCPIP4/TCPIP6 NODE command 125

- cataloging
  - databases 110
  - host databases 113
- CCSIDG precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- CCSIDM precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- CCSIDS precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- CHANGE command 689
- CHANGE DATABASE COMMENT command 129
- change database partition server configuration
  - command 1027
- CHANGE ISOLATION LEVEL command 131
- chardel file type modifier
  - EXPORT command 182
  - IMPORT command 265
  - LOAD command 387
- charsub precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- check backup command 833
- CLEANUP FOR TEXT Text Search command 1249
- CLEAR command 691
- CLEAR COMMAND LOCKS Text Search command 1250
- CLEAR EVENTS FOR INDEX Text Search command 1252
- CLI/ODBC static package binding tool command 821
- CLIPKG precompile/bind option 90
- CLOSE statement
  - issuing in CLP 21

- CLPPlus
  - CREATE DATABASE 43
  - LDAP support 39
- cnulreqd precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- code page file type modifier 387
- code pages
  - EXPORT command 182
  - IMPORT command 265
- codel file type modifier
  - EXPORT command 182
  - IMPORT command 265
  - LOAD command 387
- collection precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- COLUMN command 693
- command line processor (CLP)
  - back-end process 635
  - commands
    - syntax 8
  - help 8, 19
  - invocation command 8
  - line-continuation character 2
  - options 10
  - overview 2
  - quitting 8, 477
  - return codes 18
  - shell command 8
  - SQL statements 21
  - terminating 8, 635
- command line processor plus (CLPPlus)
  - commands
    - ! 679
    - / 680
    - . 678
    - @ 681
    - @@ 682
    - ACCEPT 683
    - APPEND 685
    - BREAK 686
    - BTITLE 687
    - CHANGE 689
    - CLEAR 691
    - CLPPLUS 31
    - COLUMN 693
    - COMPUTE 698
    - CONNECT 700
    - COPY 702
    - DEFINE 704
    - DEL 705
    - DESCRIBE 707
    - DISCONNECT 712
    - EDIT 713
    - EXECUTE 714
    - EXIT 717
    - EXPLAIN PLAN 719
    - GET 720
    - HELP 721
    - HOST 722
    - IMPORT 723
    - INPUT 725
    - LIST 726
    - PAUSE 728
    - PRINT 729
    - PROMPT 730

- command line processor plus (CLPPlus) *(continued)*
  - commands *(continued)*
    - QUIT 731
    - REMARK 732
    - REORGCHK 733
    - REPFOOTER 735
    - REPHEADER 736
    - RUN 738
    - SAVE 739
    - SET 740
    - SHOW 749
    - SPOOL 748
    - START 750
    - TTITLE 751
    - UNDEFINE 753
    - WHENEVER OSERROR 754
    - WHENEVER SQLERROR 756
  - comments 46
  - console types
    - UTF-8 support 35
  - CREATE DATABASE 43
  - interruptskip 45
  - LDAP support 39
- Command Line Processor Plus (CLPPlus)
  - bind variables 49
  - commands
    - summary 677
    - supported 54
  - connect
    - DSN aliases 35
    - KerberosDSN aliases 37
    - SERVER\_ENCRYPTDSN aliases 38
  - console types 33
  - escape characters 47
  - installing on IBM data server clients 29
  - log records 54
  - overview 29
  - restrictions 55
  - scripts 44
  - starting 30
  - traces 54
  - troubleshooting 56
  - variables
    - shellenvironment 53
- commands
  - ACTIVATE DATABASE 64
  - ADD CONTACT 66
  - ADD CONTACTGROUP 67
  - ADD DBPARTITIONNUM 68
  - ARCHIVE LOG 73
  - ATTACH 76
  - AUTOCONFIGURE 78
  - BACKUP DATABASE 81
  - BIND 90
  - CATALOG DATABASE 110
  - CATALOG DCS DATABASE 113
  - CATALOG LDAP DATABASE 115
  - CATALOG LDAP NODE 118
  - CATALOG LOCAL NODE 119
  - CATALOG NAMED PIPE NODE 121
  - CATALOG ODBC DATA SOURCE 123
  - CATALOG TCP/IP NODE 125
  - CHANGE DATABASE COMMENT 129
  - CHANGE ISOLATION LEVEL 131
  - CLPPlus
    - ! 679
    - / 680

commands (continued)

CLPPlus (continued)

. 678  
 @ 681  
 @@ 682  
 ACCEPT 683  
 APPEND 685  
 BREAK 686  
 BTITLE 687  
 CHANGE 689  
 CLEAR 691  
 COLUMN 693  
 COMPUTE 698  
 CONNECT 700  
 COPY 702  
 DEFINE 704  
 DEL 705  
 DESCRIBE 707  
 DISCONNECT 712  
 EDIT 713  
 EXECUTE 714  
 EXIT 717  
 EXPLAIN PLAN 719  
 GET 720  
 HELP 721  
 HOST 722  
 IMPORT 723  
 INPUT 725  
 LIST 726  
 PAUSE 728  
 PRINT 729  
 PROMPT 730  
 QUIT 731  
 REMARK 732  
 REORGCHK 733  
 REPFOOTER 735  
 REPHEADER 736  
 RUN 738  
 SAVE 739  
 SET 740  
 SHOW 749  
 SPOOL 748  
 START 750  
 summary 677  
 TTITLE 751  
 UNDEFINE 753  
 WHENEVER OSERROR 754  
 WHENEVER SQLERROR 756  
 CREATE DATABASE 134  
 CREATE TOOLS CATALOG 152  
 dasauto 760  
 dasprt 761  
 dasdrop 762  
 dasmigr  
 migrating DAS 763  
 dasupdt 765  
 db2 8  
 db2\_deinstall  
 details 767  
 db2\_install 770  
 db2\_local\_ps 773  
 db2acsutil 775  
 db2addicons 779  
 db2admin 780  
 db2adutl  
 details 782  
 db2advis 790

commands (continued)

db2audit 797  
 db2batch 806  
 db2bfd 816  
 db2caem 817  
 db2cap 821  
 db2cfexp 826  
 db2cfimp 828  
 db2chglbpath 829  
 db2chgpath 831  
 db2ckbkp 833  
 db2cklog 837  
 db2ckrst 839  
 db2ckupgrade  
 details 841  
 db2cli 844  
 db2cmd 857  
 db2cptsa 858  
 db2dart  
 details 859  
 db2daslevel 867  
 db2dclgn 868  
 db2diag  
 details 871  
 db2drdat  
 details 887  
 db2drvmp 889  
 db2empfa 891  
 db2envar.bat 892  
 db2evmon 893  
 db2evtbl 895  
 db2exmig  
 details 900  
 db2expln 901  
 db2extsec 910  
 db2flsn 912  
 db2fm 917  
 db2fmcu 919  
 db2fodc 921  
 db2fopt 932  
 db2fs  
 details 934  
 db2gcf 935  
 db2gov  
 details 938  
 db2govlg 940  
 db2gpmap 941  
 db2iauto 942  
 db2iclus 943  
 db2icrt 946  
 db2IdentifyType1  
 identifying type-1 indexes 958  
 db2idrop 955  
 db2ilist 961  
 db2inidb 962  
 db2inspf  
 details 965  
 db2iprune 966  
 db2isetup 968  
 db2iupdt 970  
 db2iupgrade  
 details 978  
 db2jdbcbind 981  
 db2ldcfg 983  
 db2level  
 details 984

commands (continued)

db2licm  
 details 985  
 db2listvolumes 988  
 db2logsforrwd 989  
 db2look  
 details 990  
 db2ls  
 details 1002  
 db2move 1005  
 db2mqsln 1014  
 db2mscs 1018  
 db2mtrk 1022  
 db2nchg 1027  
 db2ncrt 1029  
 db2ndrop 1031  
 db2nrcfg 1032  
 db2nrupdt 1033  
 db2nrupgrade 1034  
 db2osconf  
 details 1035  
 db2pd  
 details 1038  
 db2pdcfg  
 details 1130  
 db2perfc 1136  
 db2perfi 1138  
 db2perfr 1139  
 db2rbind 1140  
 db2relocatedb 1142  
 db2rfe  
 details 1148  
 db2rfpen 1150  
 db2rmicons 1151  
 db2sampl  
 details 1153  
 db2schex 1156  
 db2set 1158  
 db2setup  
 details 1163  
 db2snapcore 1165  
 db2start 1167  
 db2stat 1168  
 db2stop 625  
 details 1169  
 db2support  
 details 1170  
 db2swtch 1188  
 db2sync 1190  
 db2systray 1191  
 db2tapemgr 1193  
 db2tbtst 1196  
 db2tdbmgr  
 details 1197  
 db2top 1198  
 db2trc  
 details 1203  
 db2trcoff  
 details 1218  
 db2trcon  
 details 1219  
 db2ts ALTER INDEX 1244  
 db2ts CLEANUP FOR TEXT 1249  
 db2ts CLEAR COMMAND LOCKS 1250  
 db2ts CLEAR EVENTS FOR INDEX 1252  
 db2ts CREATE INDEX 1254  
 db2ts DISABLE DATABASE FOR TEXT 1261

commands (continued)

db2ts DROP INDEX 1263  
 db2ts ENABLE DATABASE FOR TEXT 1265  
 db2ts HELP 1267  
 db2ts START FOR TEXT 1269  
 db2ts STOP FOR TEXT 1271  
 db2ts UPDATE INDEX 1272  
 db2unins 1221  
 db2untag 1223  
 db2updserv 1224  
 db2xdbmig 1227  
 db2xpvt 1228  
 DEACTIVATE DATABASE 154  
 DECOMPOSE XML DOCUMENT 156  
 DECOMPOSE XML DOCUMENTS 158  
 DEREGISTER 161  
 DESCRIBE  
 details 162  
 DETACH 172  
 disable\_MQFunctions 1229  
 doce\_deinstall  
 details 1230  
 doce\_install 1231  
 DROP CONTACT 173  
 DROP CONTACTGROUP 174  
 DROP DATABASE 175  
 DROP DBPARTITIONNUM VERIFY 177  
 DROP TOOLS CATALOG 178  
 ECHO 180  
 EDIT 181  
 enable\_MQFunctions 1233  
 EXPORT 182  
 FORCE APPLICATION 192  
 GET ADMIN CONFIGURATION 194  
 GET ALERT CONFIGURATION 196  
 GET CLI CONFIGURATION 202  
 GET CONNECTION STATE 204  
 GET CONTACTGROUP 205  
 GET CONTACTGROUPS 206  
 GET CONTACTS 207  
 GET DATABASE CONFIGURATION 208  
 GET DATABASE MANAGER CONFIGURATION 221  
 GET DATABASE MANAGER MONITOR SWITCHES 230  
 GET DESCRIPTION FOR HEALTH INDICATOR 233  
 GET HEALTH NOTIFICATION CONTACT LIST 234  
 GET HEALTH SNAPSHOT 235  
 GET INSTANCE 238  
 GET MONITOR SWITCHES 239  
 GET RECOMMENDATIONS 242  
 GET ROUTINE 245  
 GET SNAPSHOT  
 details 247  
 HELP 263  
 HISTORY 264  
 IMPORT 265  
 INGEST 294  
 INITIALIZE TAPE 337  
 INSPECT  
 details 338  
 installDSDriver 1235  
 installFixPack 1236  
 LIST ACTIVE DATABASES 346  
 LIST APPLICATIONS 348  
 LIST COMMAND OPTIONS 351  
 LIST DATABASE DIRECTORY 352  
 LIST DATABASE PARTITION GROUPS 355  
 LIST DBPARTITIONNUMS 357



commands (*continued*)

- LIST DCS APPLICATIONS 358
- LIST DCS DIRECTORY 360
- LIST DRDA INDOUBT TRANSACTIONS 362
- LIST HISTORY 364
- LIST INDOUBT TRANSACTIONS 369
- LIST NODE DIRECTORY 371
- LIST ODBC DATA SOURCES 374
- LIST PACKAGES/TABLES 375
- LIST TABLESPACE CONTAINERS 378
- LIST TABLESPACES 380
- LIST UTILITIES 385
- LOAD 387
- LOAD QUERY 431
- Microsoft Cluster Server 943
- MIGRATE DATABASE 437
- MQListener 1014
- PING 438
- PRECOMPILE 440
- PRUNE HISTORY/LOGFILE 466
- PUT ROUTINE 468
- QUERY CLIENT 469
- QUIESCE 470
- QUIESCE TABLESPACES FOR TABLE 474
- QUIT 477
- REBIND 478
- RECOVER DATABASE 482
- redirecting output 2
- REDISTRIBUTE DATABASE PARTITION GROUP 489
- REFRESH LDAP 497
- REGISTER 499
- REORG INDEXES/TABLE 506
- REORGCHK 523
- RESET ADMIN CONFIGURATION 534
- RESET ALERT CONFIGURATION 536
- RESET DATABASE CONFIGURATION 538
- RESET DATABASE MANAGER CONFIGURATION 540
- RESET MONITOR 542
- RESTART DATABASE 544
- RESTORE DATABASE 547
- REWIND TAPE 569
- ROLLFORWARD DATABASE 570
- RUNCMD 581
- RUNSTATS
  - details 582
- SET CLIENT 598
- SET RUNTIME DEGREE 601
- SET SERVEROUTPUT 603
- SET TABLESPACE CONTAINERS 605
- SET TAPE POSITION 608
- SET UTIL\_IMPACT\_PRIORITY 609
- SET WORKLOAD
  - details 611
- SET WRITE 612
- setup 1241
- START DATABASE MANAGERcommandscommands
  - START DBMdb2start 614
- START HADR 623
- STOP DATABASE MANAGER 625
- STOP DBM 625
- STOP HADR 629
- syntax help 59
- TAKEOVER HADR 631
- TERMINATE 635
- UNCATALOG DATABASE 636
- UNCATALOG DCS DATABASE 637
- UNCATALOG LDAP DATABASE 638

commands (*continued*)

- UNCATALOG LDAP NODE 639
- UNCATALOG NODE 640
- UNCATALOG ODBC DATA SOURCE 641
- UNQUIESCE 642
- UPDATE ADMIN CONFIGURATION 644
- UPDATE ALERT CONFIGURATION 646
- UPDATE ALTERNATE SERVER FOR DATABASE 651
- UPDATE ALTERNATE SERVER FOR LDAP
  - DATABASE 653
- UPDATE CLI CONFIGURATION 654
- UPDATE COMMAND OPTIONS 656
- UPDATE CONTACT 658
- UPDATE CONTACTGROUP 659
- UPDATE DATABASE CONFIGURATION 660
- UPDATE DATABASE MANAGER CONFIGURATION 664
- UPDATE HEALTH NOTIFICATION CONTACT LIST 666
- UPDATE HISTORY FILE 667
- UPDATE LDAP NODE 669
- UPDATE MONITOR SWITCHES 671
- UPDATE XMLSCHEMA 673
- UPGRADE DATABASE
  - details 674
- COMPLETE XMLSCHEMA command 133
- compound file type modifier
  - IMPORT command 265
- COMPUTE command
  - description 698
- configuration
  - administration
    - resetting to default 534
    - sample 194
  - CLI 202
  - database manager 221
  - databases
    - resetting to default 538
    - sample 208
    - UPDATE DATABASE CONFIGURATION
      - command 660
- configure DB2 database for problem determination behavior
  - command 1130
- configure LDAP environment command 983
- CONNECT command 700
- connect precompile option 440
- CONNECT statement
  - run through the CLP 21
- connectivity configuration export tool command 826
- connectivity configuration import tool command 828
- continuation character
  - command line processor (CLP) 2
- Control DB2 instance command 935
- COPY command 702
- CREATE DATABASE command
  - details 134
- CREATE INDEX Text Search command 1254
- create instance command 946
- create main menu entries for DB2 tools command 779
- create sample database command 1153
- CREATE TOOLS CATALOG command 152
- cursor stability (CS)
  - changing 131

## D

- dasauto command 760
- dasrct command 761
- dasdrop command 762

- dasmigr command
  - details 763
- dasupdt command 765
- data representation
  - fragmentation 506
  - integrity
    - isolation levels 131
  - redistribution
    - REDISTRIBUTE DATABASE PARTITION GROUP command 489
- database analysis and reporting tool command
  - details 859
- database configuration file
  - resetting values 538
  - retrieving values 208
  - sample 208
  - updating 660
- Database Connection Services (DCS) directory
  - removing entries 637
- database directories
  - changing comments 129
  - details 352
  - sample content 352
- database manager
  - monitor switches
    - GET DATABASE MANAGER MONITOR SWITCHES command 230
    - GET MONITOR SWITCHES command 239
  - starting 614
  - statistics 247
  - stopping 625
- database manager configuration file
  - retrieving values 221
  - sample file 221
- database movement tool command 1005
- database system monitor
  - GET DATABASE MANAGER MONITOR SWITCHES command 230
  - GET MONITOR SWITCHES command 239
  - GET SNAPSHOT command 247
  - RESET MONITOR command 542
  - UPDATE MONITOR SWITCHES command 671
- databases
  - backups
    - history file 466
  - cataloging
    - CATALOG DATABASE command 110
  - creating
    - CREATE DATABASE command 134
  - deleting
    - Database Connection Services (DCS) 637
    - DROP DATABASE command 175
    - system database directory 636
  - dropping
    - DROP DATABASE command 175
  - exporting from table into file
    - EXPORT command 182
  - home directory entry 352
  - importing from file into table
    - IMPORT command 265
  - indirect directory entry 352
  - information collection 247
  - loading data into tables 387
  - migrating
    - MIGRATE DATABASE command 437
  - monitoring
    - db2top command 1198
- databases (*continued*)
  - monitoring (*continued*)
    - RESET MONITOR command 542
  - rebuilding
    - RESTORE DATABASE command 547
  - recovering
    - ROLLFORWARD DATABASE command 570
  - remote directory entry 352
  - reorganizing 523
  - restarting 544
  - restoring 547
  - rollforward recovery
    - ROLLFORWARD DATABASE command 570
  - statistics 582
  - uncataloging
    - Database Connection Services (DCS) 637
    - system database directory 636
  - upgrading
    - UPGRADE DATABASE command 674
- dateformat file type modifier
  - IMPORT command 265
  - LOAD command 387
- DB2 administration server (DAS)
  - configuring 194
  - creating 761
  - dropping 780
  - managing 780
- db2 command
  - details 8
- DB2 Connect
  - connections supported 113
- DB2 copies
  - validation tool 1225
- DB2 fault monitor command 917
- DB2 Governor
  - querying log files 940
  - starting 938
  - stopping 938
- DB2 Information Center
  - updating 1285, 1286
  - versions 1284
- DB2 process status for Linux/UNIX command 773
- DB2 process status for Windows command 1168
- DB2 profile registry command 1158
- DB2 servers
  - administering
    - configuration file changes 644
    - configuration file value reset 534
    - configuration file value retrieval 194
- DB2 statistics and DDL extraction tool command 990
- DB2 Text Search
  - ALTER INDEX command 1244
  - CLEAR COMMAND LOCKS command 1250
  - CLEAR EVENTS FOR INDEX command 1252
- commands
  - ALTER INDEX 1244
  - CLEANUP FOR TEXT 1249
  - CLEAR COMMAND LOCKS 1250
  - CLEAR EVENTS FOR INDEX 1252
  - CREATE INDEX 1254
  - DISABLE DATABASE FOR TEXT 1261
  - DROP INDEX 1263
  - ENABLE DATABASE FOR TEXT 1265
  - HELP 1267
  - START FOR TEXT 1269
  - STOP FOR TEXT 1271
  - UPDATE INDEX 1272

DB2 Text Search (*continued*)

- CREATE INDEX command 1254
- DISABLE DATABASE FOR TEXT command 1261
- DROP INDEX command 1263
- ENABLE DATABASE FOR TEXT command 1265
- HELP command 1267
- START FOR TEXT command 1269
- STOP FOR TEXT command 1271
- UPDATE INDEX command 1272

DB2 workload management

- SET WORKLOAD command 611

db2\_deinstall command

- details 767

db2\_install command

- details 770

db2\_local\_ps command 773

db2acsutil command 775

db2addicons command 779

db2admin command 780

db2adutl command

- details 782

db2advis command 790

db2audit command 797

db2batch command

- details 806

db2bfd command

- details 816

db2caem command

- event monitoring 817

db2cap command 821

db2cat command 823

db2cfexp command 826

db2cfimp command 828

db2chglbpath command 829

db2chgpath command 831

db2ckbkp command 833

db2cklog command

- troubleshooting 837

db2ckrst command 839

db2ckupgrade command

- details 841

db2cli command 844

db2cmd command 857

db2cptsa command 858

db2dart command

- details 859

db2daslevel command 867

db2dclgn command

- details 868

db2diag command

- details 871

db2drdat command

- details 887

db2drvmp command 889

db2empfa command 891

db2envar.bat command

- details 892

db2evmon command

- description 893

db2evtbl command 895

db2exfmt command

- details 897

db2exmig command

- details 900

db2expln command

- details 901

db2extsec command 910

db2flsn command 912

db2fm command

- details 917

db2fmcu command 919

db2fodc command 921

db2fopt command

- description 932

db2fs command 934

db2gcf command 935

db2gov command

- details 938

db2govlg command 940

db2gpmap command 941

db2iauto command 942

db2iclus command 943

db2icrt command

- details 946

db2IdentifyType1 command

- details 958

db2idrop command

- details 955

db2ilist command

- details 961

db2inidb command

- details 962

db2inspf command

- details 965

db2iprune command

- details 966

db2isetup command

- details 968

db2iupdt command

- details 970

db2iupgrade command

- details 978

db2jdbcbind command 981

db2ldcfg command

- details 983

db2level command

- details 984

db2licm command

- details 985

db2listvolumes command 988

db2logsforrwd command 989

db2look command

- details 990

db2ls command

- details 1002

db2move command

- details 1005

db2mqlsn command

- details 1014

db2mscs command

- details 1018

db2mtrk command

- details 1022

db2nchg command

- details 1027

db2ncrt command

- details 1029

db2ndrop command

- details 1031

db2nrcfg command 1032

db2nrupdt command 1033

db2nrupgrade command 1034

DB2OPTIONS environment variable

- setting CLP options 10

- db2osconf command
  - details 1035
- db2pd command
  - details 1038
- db2pdcfg command
  - details 1130
- db2perfc command
  - details 1136
- db2perfi command
  - details 1138
- db2perfr command
  - details 1139
- db2rbind command
  - details 1140
- db2relocatedb command
  - details 1142
- db2rfe command
  - details 1148
- db2rfpen command
  - details 1150
- db2rmicons command
  - details 1151
- db2rspgn command
  - details 1152
- db2sampl command 1153
- db2schex command 1156
- db2set command
  - details 1158
- db2setup command
  - details 1163
- db2snapcore command 1165
- db2start command
  - details 1167
- db2stat command 1168
- db2stop command
  - details 1169
- db2swtch command 1188
- db2sync command 1190
- db2systray command 1191
- db2tapemgr command
  - details 1193
- db2tbst command 1196
- db2tdbmgr command
  - details 1197
- db2top command
  - details 1198
- db2trc command
  - details 1203
- db2trcoff command
  - details 1218
- db2trcon command
  - details 1219
- db2ts commands
  - ALTER INDEX 1244
  - CLEANUP FOR TEXT 1249
  - CLEAR COMMAND LOCKS 1250
  - CLEAR EVENTS FOR INDEX 1252
  - CREATE INDEX 1254
  - DISABLE DATABASE FOR TEXT 1261
  - DROP INDEX 1263
  - ENABLE DATABASE FOR TEXT 1265
  - HELP 1267
  - START FOR TEXT 1269
  - STOP FOR TEXT 1271
  - UPDATE INDEX 1272
- db2unins command 1221
- db2untag command 1223
- db2updserv command 1224
- db2val command
  - details 1225
- db2xdbmig command 1227
- db2xpvt command 1228
- DCLGEN command 868
- DEACTIVATE DATABASE command
  - details 154
- declaration generator command 868
- DECLARE CURSOR statement
  - running in CLP 21
- DECOMPOSE XML DOCUMENT command 156
- DECOMPOSE XML DOCUMENTS command
  - details 158
- decplusblank file type modifier
  - EXPORT command 182
  - IMPORT command 265
  - LOAD command 387
- decpt file type modifier
  - EXPORT command 182
  - IMPORT command 265
  - LOAD command 387
- default configuration resetting
  - databases 538
  - DB2 administration server (DAS) 534
- DEFINE command 704
- DEL command 705
- delimiters
  - modifying 1279
  - restrictions on moving data 1279
- delprioritychar file type modifier
  - IMPORT command 265
  - LOAD command 387
- DEREGISTER command 161
- DESCRIBE CLPPlus command 707
- DESCRIBE command
  - details 162
- Design Advisor
  - details 790
- DETACH command
  - details 172
- directories
  - Database Connection Services (DCS)
    - deleting entries 637
  - local database
    - changing comments 129
  - node
    - deleting entries 640
  - system database
    - changing comments 129
    - deleting entries 636
    - uncataloging databases (API) 636
- DISABLE DATABASE FOR TEXT Text Search command 1261
- disable\_MQFunctions command 1229
- DISCONNECT command 712
- display GUIDs for all disk volumes command 988
- doce\_deinstall command
  - details 1230
- doce\_install command
  - details 1231
- documentation
  - overview 1281
  - PDF files 1282
  - printed 1282
  - terms and conditions of use 1288
- DRDA trace command 887

- DROP CONTACT command
  - details
    - without using ADMIN\_CMD 173
- DROP CONTACTGROUP command
  - details
    - without using ADMIN\_CMD 174
- DROP DATABASE command 175
- drop database partition server from an instance
  - command 1031
- DROP DBPARTITIONNUM VERIFY command 177
- DROP INDEX Text Search command 1263
- DROP TOOLS CATALOG command 178
- dumpfile file type modifier 387
- dumping trace to file
  - db2trc command 1203
- DYNAMICRULES precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440

## E

- ECHO command 180
- EDIT CLPPlus command 181
- EDIT command 713
- embedded SQL applications
  - runtime library search path 829
- ENABLE DATABASE FOR TEXT Text Search command 1265
- enable\_MQFunctions command 1233
- environment variables
  - DB2OPTIONS 10
- errors
  - checksum
    - database configuration file 538, 660
    - database manager configuration file 534
- event monitoring
  - capture activity event monitor data 817
  - db2caemdb2support 817
- event monitors
  - db2evmon command 893
- EXECUTE command 714
- exit codes in CLP 18
- EXIT command 717
- explain tables
  - formatting contents 897
- EXPORT command
  - details
    - without using ADMIN\_CMD 182
- exporting
  - data
    - EXPORT command 182
    - file type modifiers 182
- Extract Data Server Driver component command 1235

## F

- fastparse file type modifier
  - LOAD command 387
- fault monitor
  - db2fmcu command 919
- federated precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- federated\_asynchrony precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440

- FETCH statement
  - running in CLP 21
- file formats
  - exporting table to file 182
  - importing file to table 265
- file type modifiers
  - EXPORT utility 182
  - IMPORT command 265
  - LOAD command 387
- find log sequence number command 912
- First Steps
  - db2fs command 934
- FODC
  - db2fodc command 921
- FORCE APPLICATION command
  - without using ADMIN\_CMD 192
- forcein file type modifier
  - IMPORT command 265
  - LOAD command 387
- format inspect results command 965
- format trap file command 1228
- funcpath precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440

## G

- generate event monitor target table definitions command 895
- generatedignore file type modifier
  - IMPORT command 265
  - LOAD command 387
- generatedmissing file type modifier
  - IMPORT command 265
  - LOAD command 387
- generatedoverride file type modifier
  - LOAD command 387
- generic precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- GET ADMIN CONFIGURATION command 194
- GET ALERT CONFIGURATION command 196
- GET CLI CONFIGURATION command 202
- GET command 720
- GET CONNECTION STATE command 204
- GET CONTACTGROUP command 205
- GET CONTACTGROUPS command 206
- GET CONTACTS command 207
- GET DATABASE CONFIGURATION command 208
- GET DATABASE MANAGER CONFIGURATION
  - command 221
- GET DATABASE MANAGER MONITOR SWITCHES
  - command 230
- GET DESCRIPTION FOR HEALTH INDICATOR
  - command 233
- get distribution map command 941
- GET HEALTH NOTIFICATION CONTACT LIST
  - command 234
- GET HEALTH SNAPSHOT command 235
- GET INSTANCE command 238
- GET MONITOR SWITCHES command 239
- GET RECOMMENDATIONS command 242
- GET ROUTINE command 245
- GET SNAPSHOT command
  - details 247
  - effect on UPDATE MONITOR SWITCHES command 671
- get table space state command 1196
- GRANT BIND option 90

GRANTGROUP bind option 90  
GRANTUSER BIND option 90

## H

help  
  commands 19  
  messages 19  
  SQL statements 1284  
HELP command  
  CLPPlus 721  
  DB2 Database for Linux, UNIX, and Windows 263  
  Text Search 1267  
highlighting conventions viii  
HISTORY command 264  
HOST command 722  
host systems  
  cataloging databases 113  
  connections supported by DB2 Connect  
    CATALOG DCS DATABASE command 113  
  removing DCS catalog entries 637  
how this book is structured vii

## I

IBM data server clients  
  installing  
    CLPPlus 29  
identityignore file type modifier  
  IMPORT command 265  
  LOAD command 387  
identitymissing file type modifier  
  IMPORT command 265  
  LOAD command 387  
identityoverride file type modifier  
  LOAD command 387  
images  
  reducing size 966  
implieddecimal file type modifier  
  IMPORT command 265  
  LOAD command 387  
IMPORT command  
  details  
    without using ADMIN\_CMD 265  
importing  
  data 265  
indexes  
  REORGCHK command 523  
  statistics  
    RUNSTATS command 582  
indexfreespace file type modifier  
  LOAD command 387  
indexixf file type modifier  
  IMPORT command 265  
indexschema file type modifier  
  IMPORT command 265  
indoubt transaction field 369  
INGEST command 294  
initialize a mirrored database command 962  
INITIALIZE TAPE command  
  without using ADMIN\_CMD 337  
INPUT command 725  
insert precompile/bind option  
  BIND command 90  
  PRECOMPILE command 440

INSPECT command  
  details 338  
install DB2 command  
  db2iprune 966  
  db2setup 1163  
  setup 1241  
install DB2 Information Center command 1231  
install DB2 product command 770  
install or update DB2 HA scripts command 858  
installation images  
  reducing size 966  
installDSDriver command  
  description 1235  
installFixPack command  
  details 1236  
interactive CLI command 844  
IPX/SPX nodes  
  uncataloging 640  
isolation levels  
  CHANGE ISOLATION LEVEL command 131  
isolation precompile/bind option 90, 440

## J

JDBC  
  package binder utility command 981

## K

keepblanks file type modifier  
  IMPORT command 265  
  LOAD command 387

## L

LANGLEVEL precompile option  
  SQL92E 440  
level precompile option 440  
licenses  
  License Management Tool command 985  
line continuation character  
  command line processor (CLP) 2  
LIST ACTIVE DATABASES command 346  
LIST APPLICATIONS command 348  
LIST COMMAND OPTIONS command 351  
LIST DATABASE DIRECTORY command 352  
LIST DATABASE PARTITION GROUPS command  
  details 355  
LIST DBPARTITIONNUMS command 357  
LIST DCS APPLICATIONS command  
  details 358  
LIST DCS DIRECTORY command 360  
LIST DRDA INDOUBT TRANSACTIONS command  
  details 362  
LIST HISTORY command 364  
LIST INDOUBT TRANSACTIONS command  
  details 369  
list installed DB2 products and features command 1002  
list instances command 961  
list logs required for rollforward recovery command 989  
LIST NODE DIRECTORY command 371  
LIST ODBC DATA SOURCES command 374  
LIST PACKAGES command  
  details 375  
LIST PACKAGES/TABLES command 375

- LIST TABLES command
  - details 375
- LIST TABLESPACE CONTAINERS command
  - details 378
- LIST TABLESPACES command
  - details 380
- LIST UTILITIES command
  - details 385
- LOAD command
  - details
    - without using ADMIN\_CMD 387
- LOAD QUERY command
  - details 431
- load utility
  - temporary files
    - space requirements 387
- loading
  - file to database table 387
  - file type modifiers 387
- lobsinfile file type modifier
  - EXPORT command 182
  - IMPORT command 265
  - LOAD command 387
- local database directory
  - changing comments 129
- locks
  - resetting maximum to default 538
- log records
  - Command Line Processor Plus (CLPPlus) 54
- logs
  - listing during rollforward 570
- longerror precompile option 440

## M

- manage DB2 snapshot backup objects command 775
- manage logs on tape command 1193
- map DB2 database drive command 889
- memory tracker command
  - details 1022
- messages
  - help 8, 19
  - precompile/bind option 90, 440
- Microsoft Cluster Server (MSCS)
  - command 943
- MIGRATE DATABASE command
  - details 437
- migrate explain tables command 900
- migrate the DB2 Administration Server command 763
- migrate tools catalog database command 1197
- migrate XSR objects command 1227
- migration
  - DB2 Administration Server
    - command 763
  - explain tables
    - command 900
  - tools catalog database 1197
  - XSR objects 1227
- monitoring
  - databases 230, 239
  - db2pd command 1038
- moving data
  - between databases 265
  - delimiter restrictions 1279
- MQListener command 1014
- multi-page file allocation command 891

## N

- naming conventions
  - database manager objects 1276
- no commit (NC) 131
- nochecklengths file type modifier
  - IMPORT command 265
  - LOAD command 387
- node directories
  - deleting entries 640
- nodefaults file type modifier
  - IMPORT command 265
- nodes
  - SOCKS
    - CATALOG TCPIP/TCPIP4/TCPIP6 NODE
      - command 125
- nodoubledel file type modifier
  - EXPORT command 182
  - IMPORT command 265
  - LOAD command 387
- noeofchar file type modifier
  - IMPORT command 265
  - LOAD command 387
- noheader file type modifier
  - LOAD command 387
- NOLINEMACRO precompile option 440
- non-root installations
  - configuring 1032
  - enabling root-based features 1148
  - updating instances 1033
- norowwarnings file type modifier
  - LOAD command 387
- notices 1291
- notypeid file type modifier
  - IMPORT command 265
- NULL
  - SQL value
    - command line processor representation 2
    - string 2
- nullindchar file type modifier
  - IMPORT command 265
  - LOAD command 387

## O

- off trace options 1218
- on trace options 1219
- open DB2 Command Window command 857
- OPEN statement
  - running in CLP 21
- optimization
  - REORG INDEXES/TABLE command 506
- optlevel precompile option 440
- output precompile option 440
- owner precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440

## P

- packages
  - precompile option 440
  - re-creating 478
- packeddecimal file type modifier 387
- pagefreespace file type modifier 387

- passwords
  - changing
    - ATTACH command 76
    - CONNECT statement 21
- PAUSE command 728
- performance
  - indexes
    - REORGCHK command 523
  - tables
    - reorganizing 506
    - REORGCHK command 523
  - Windows
    - Performance Monitor registration tool command 1139
- performance counters registration utility command 1138
- phantom quiesce 474
- PING command 438
- PRECOMPILE command
  - details 440
- PREP command 440
- PRINT command 729
- privileges
  - databases
    - granted when creating database 134
- problem determination
  - diagnostic tools
    - db2fodc command 921
    - db2snapcore 1165
    - db2support 1170
  - information available 1288
  - tutorials 1288
- PROMPT command 730
- proxy nodes
  - Tivoli Storage Manager (TSM)
    - db2adutl command 782
- PRUNE HISTORY/LOGFILE command
  - without using ADMIN\_CMD 466
- PUT ROUTINE command 468

## Q

- qualifier precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- QUERY CLIENT command 469
- queryopt precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- QUIESCE command 470
- QUIESCE TABLESPACES FOR TABLE command
  - without using ADMIN\_CMD 474
- quiescing
  - table spaces 474
- QUIT CLPPLus command 731
- QUIT command 477

## R

- read stability (RS)
  - changing 131
- rebind all packages command 1140
- REBIND command
  - details 478
- reclen file type modifier
  - IMPORT command 265
  - LOAD command 387

- RECOVER DATABASE command
  - details 482
- recovery
  - databases
    - RESTORE DATABASE command 547
    - with rollforward 570
    - without roll forward 547
- REDISTRIBUTE DATABASE PARTITION GROUP command
  - without using ADMIN\_CMD 489
- REFRESH LDAP command 497
- REGISTER command 499
- REGISTER XMLSCHEMA command 502
- REGISTER XSROBJECT command 504
- release container tag command 1223
- release precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- relocate database command 1142
- REMARK command 732
- remove a DB2 Administration Server command 762
- remove instance command 955
- remove main menu entries for DB2 tools command 1151
- REORG INDEXES command
  - without using ADMIN\_CMD 506
- REORG TABLE command
  - without using ADMIN\_CMD 506
- REORGCHK command 523
- repeatable read (RR)
  - changing 131
- REPFOOTER command
  - description 735
- REPHEADER command
  - description 736
- RESET ADMIN CONFIGURATION command 534
- RESET ALERT CONFIGURATION command
  - without using ADMIN\_CMD 536
- RESET DATABASE CONFIGURATION command
  - without using ADMIN\_CMD 538
- RESET DATABASE MANAGER CONFIGURATION command
  - without using ADMIN\_CMD 540
- reset database performance values command 1136
- RESET MONITOR command 542
- reset rollforward pending state command 1150
- response files
  - generator
    - db2rspgn command 1152
- RESTART DATABASE command
  - details 544
- RESTORE DATABASE command
  - details 547
- restoring
  - earlier versions of DB2 databases 547
- return codes
  - command line processor (CLP) 18
- REWIND TAPE command
  - without using ADMIN\_CMD 569
- ROLLFORWARD DATABASE command
  - details 570
- RUN command 738
- RUNCMD command 581
- RUNSTATS command
  - details
    - without using ADMIN\_CMD 582

## S

- SAVE command 739



- search paths
  - changing 831
- SELECT statement
  - EXPORT command 182
  - running in CLP 21
- SET CLIENT command 598
- SET command 740
- set permissions for DB2 objects command 910
- SET RUNTIME DEGREE command 601
- SET SERVEROUTPUT command 603
- SET TABLESPACE CONTAINERS command 605
- SET TAPE POSITION command
  - without using ADMIN\_CMD 608
- set up Windows Failover Utility command 1018
- SET UTIL\_IMPACT\_PRIORITY command 609
- SET WORKLOAD command
  - details 611
- SET WRITE command
  - details 612
- setup command 1241
- SHOW command 749
- show current DAS level command 867
- show DB2 service level command 984
- show product updates command 1224
- SIGALRM signal
  - starting database manager 614
- SIGINT signal
  - starting database manager 614
- snapcore command 1165
- SOCKS
  - nodes
    - parameter 125
- SPOOL command 748
- SQL and XQuery Explain command 901
- SQL statements
  - command line 21
  - help
    - accessing 8
    - displaying 1284
- sqlca precompile option 440
- sqlerror precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- sqlflag precompile option 440
- sqlrules precompile option 440
- sqlwarn precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- START CLPPlus command 750
- START DATABASE MANAGER command
  - details 614
- start DB2 command 1167
- start DB2 Synchronizer command 1190
- start DB2 system tray command 1191
- START FOR TEXT Text Search command 1269
- START HADR command
  - details 623
- start instance creation interface command 968
- starting
  - DB2 1167
- statistics
  - database 582
  - database manager 247
  - reorganizing indexes 523
  - REORGCHK command 523
- STOP DATABASE MANAGER command
  - details 625
- stop DB2 command 1169
- STOP FOR TEXT Text Search command 1271
- STOP HADR command
  - details 629
- stopping
  - DB2 1169
- storage
  - physical 506
- strdel precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- striptblanks file type modifier
  - IMPORT command 265
  - LOAD command 387
- striptnulls file type modifier
  - IMPORT command 265
  - LOAD command 387
- subtableconvert file type modifier 387
- switch default DB2 copy and database client interface copy
  - command 1188
- syncpoint precompile option 440
- syntax diagrams
  - reading ix
- system catalogs
  - analyzing 823
- system database directory
  - changing comments 129
  - deleting entries 636
  - uncataloging 636

## T

- tables
  - exporting to files 182
  - importing files 265
  - loading 387
  - reorganization
    - determining need for 523
    - REORG INDEXES/TABLE command 506
  - statistics
    - details 582
- TAKEOVER HADR command
  - details 631
- tape backups
  - BACKUP DATABASE command 81
- target precompile option 440
- TCP/IP
  - uncataloging nodes 640
- temporary files
  - load utility
    - space requirements 387
- TERMINATE command 635
- termination
  - abnormal 544
  - command line processor back-end process 635
  - normal 625
- terms and conditions
  - publications 1288
- text precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- timeformat file type modifier
  - IMPORT command 265
  - LOAD command 387
- timestampformat file type modifier
  - IMPORT command 265
  - LOAD command 387

- Tivoli Storage Manager (TSM)
  - archived images 782
- totalfreespace file type modifier
  - LOAD command 387
- trace command 1203
- traces
  - activating 1203, 1218, 1219
  - Command Line Processor Plus (CLPPlus) 54
- transform group precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440
- troubleshooting
  - db2pd command 1038
  - log filesdb2cklog command 837
  - online information 1288
  - tutorials 1288
- true type font requirement for CLP 2
- TTITLE command
  - description 751
- turning trace off
  - db2trcoff command 1218
- turning trace on
  - db2trcon command 1219
- tutorials
  - list 1288
  - problem determination 1288
  - pureXML 1288
  - troubleshooting 1288
- Type-1 index identification tool command 958

## U

- UNCATALOG DATABASE command 636
- UNCATALOG DCS DATABASE command 637
- UNCATALOG LDAP DATABASE command 638
- UNCATALOG LDAP NODE command 639
- UNCATALOG NODE command 640
- UNCATALOG ODBC DATA SOURCE command 641
- uncataloging
  - database entries 636
  - host DCS database entries 637
  - system database directory 636
- uncommitted read (UR) isolation level
  - changing 131
- UNDEFINE command 753
- uninstall DB2 Information Center command 1230
- uninstall DB2 products command 1221
- uninstall DB2 products, features, or languages command 767
- UNQUIESCE command 642
- UPDATE ADMIN CONFIGURATION command 644
- UPDATE ALERT CONFIGURATION command
  - without using ADMIN\_CMD 646
- UPDATE ALTERNATE SERVER FOR DATABASE
  - command 651
- UPDATE ALTERNATE SERVER FOR LDAP DATABASE
  - command 653
- UPDATE CLI CONFIGURATION command 654
- UPDATE COMMAND OPTIONS command 656
- UPDATE CONTACT command
  - without using ADMIN\_CMD 658
- UPDATE CONTACTGROUP command
  - without using ADMIN\_CMD 659
- UPDATE DATABASE CONFIGURATION command
  - without using ADMIN\_CMD 660
- UPDATE DATABASE MANAGER CONFIGURATION
  - command
  - without using ADMIN\_CMD 664

- UPDATE HEALTH NOTIFICATION CONTACT LIST
  - command
  - without using ADMIN\_CMD 666
- UPDATE HISTORY FILE command 667
- UPDATE INDEX Text Search command 1272
- update installed DB2 products command 1236
- update instances command 970
- UPDATE LDAP NODE command 669
- UPDATE MONITOR SWITCHES command 671
- UPDATE XMLSCHEMA command 673
- updates
  - DB2 Information Center 1285, 1286
  - update DAS command 765
- UPGRADE DATABASE command
  - details 674
- upgrade instance command 978
- upgrade non-root instance command 1034
- usedefaults file type modifier
  - IMPORT command 265
  - LOAD command 387
- utility for kernel parameter values command 1035

## V

- validate precompile/bind option
  - BIND command 90
  - PRECOMPILE command 440

## W

- WCHARTYPE precompiler option
  - details 440
- WHENEVER OSERROR command 754
- WHENEVER SQLERROR command 756
- who should use this book vii
- workstations
  - remote
    - cataloging databases 110
    - uncataloging databases 636
    - uncataloging nodes 640

## X

- X/Open Backup Services API (XBSA)
  - BACKUP DATABASE command option 81
- XML schema repository (XSR)
  - ADD XMLSCHEMA DOCUMENT command 71
  - COMPLETE XMLSCHEMA command 133
  - REGISTER XMLSCHEMA command 502
  - REGISTER XSROBJECT command 504
  - UPDATE XMLSCHEMA command 673

## Z

- zoned decimal file type modifier
  - LOAD command 387





Printed in USA

SC27-3868-00



Spine information:

IBM DB2 10.1 for Linux, UNIX, and Windows

Command Reference

