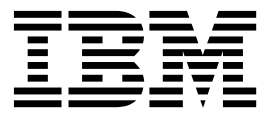


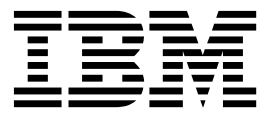
DB2 10 for z/OS

Command Reference



DB2 10 for z/OS

Command Reference



Notes

Before using this information and the product it supports, be sure to read the general information under “Notices” at the end of this information.

October 11, 2017 edition

This edition applies to DB2 10 for z/OS (product number 5605-DB2), DB2 10 for z/OS Value Unit Edition (product number 5697-P31), and to any subsequent releases until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Specific changes are indicated by a vertical bar to the left of a change. A vertical bar to the left of a figure caption indicates that the figure has changed. Editorial changes that have no technical significance are not noted.

© Copyright IBM Corporation 1983, 2017.

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

Contents

About this information	ix
Who should read this information	ix
DB2 Utilities Suite for z/OS	ix
Terminology and citations	x
Accessibility features for DB2 10 for z/OS	x
How to send your comments	xi
How to read syntax diagrams	xi

Part 1. About DB2 and related commands 1

Chapter 1. Privileges and authorization IDs for DB2 commands. 3

Chapter 2. Types of commands. 5

The DSN command and its subcommands	5
DB2 commands	7
IMS commands	8
CICS attachment facility commands	8
Administrative task scheduler commands.	9
z/OS IRLM commands	9
TSO CLISTS	10

Chapter 3. Syntax rules for DB2 commands 11

Chapter 4. Syntax rules for DSN subcommands 15

Chapter 5. Scope of commands in data sharing 17

Chapter 6. Output from DB2 commands 19

Chapter 7. Command line processor 21

Tutorial: Using the command line processor	21
Running the command line processor.	25
Command line processor ADD XMLSCHEMA DOCUMENT command	26
Command line processor BIND command	28
Command line processor CHANGE ISOLATION command	29
Command line processor CHANGE MAXCOLUMNWIDTH command	29
Command line processor CHANGE MAXLINESFROMSELECT command	30
Command line processor COMMIT command	30
Command line processor COMPLETE XMLSCHEMA command.	31
Command line processor CONNECT command	32
Command line processor DESCRIBE CALL command	33
Command line processor DESCRIBE TABLE command.	35
Command line processor DISCONNECT command	36
Command line processor DISPLAY RESULT SETTINGS command	36
Command line processor ECHO command	36
Command line processor help command	37
Command line processor LIST COMMAND OPTIONS command	38
Command line processor LIST TABLES command	38
Command line processor properties file	39
Command line processor REGISTER XMLSCHEMA command	42
Command line processor REMOVE XMLSCHEMA command	44
Command line processor ROLLBACK command	44
Command line processor SQL statements	45

	Command line processor start command	46
	Command line processor TERMINATE command	48
	Command line processor z/OS UNIX System Services command	48
	Command line processor UPDATE COMMAND OPTIONS command	49
	Chapter 8. DB2 command text is recorded in the DB2 recovery log	51
<hr/>		
	Part 2. DB2 and related commands	53
	Chapter 9. ABEND (DSN)	55
	Chapter 10. -ACCESS DATABASE (DB2)	57
	Chapter 11. -ALTER BUFFERPOOL (DB2)	61
	Chapter 12. -ALTER GROUPBUFFERPOOL (DB2)	67
	Chapter 13. -ALTER UTILITY (DB2).	71
	Chapter 14. -ARCHIVE LOG (DB2)	75
	Chapter 15. BIND PACKAGE (DSN).	81
	Chapter 16. BIND PLAN (DSN).	91
	Chapter 17. BIND QUERY (DSN)	97
	Chapter 18. BIND and REBIND options for packages and plans	101
	ACQUIRE bind option	101
	ACTION bind option.	102
	APCOMPARE bind option	103
	APRETAINDUP bind option	104
	APREUSE bind option	105
	CACHESIZE bind option	106
	COLLID bind option	107
	CONCURRENTACCESSRESOLUTION bind option	107
	COPY bind option.	108
	CURRENTDATA bind option	110
	CURRENTSERVER bind option	111
	DBPROTOCOL bind option	112
	DEFER and NODEFER bind options.	113
	DEGREE bind option.	114
	DEPLOY bind option.	115
	DISCONNECT bind option.	116
	DYNAMICRULES bind option	116
	ENABLE and DISABLE bind options	122
	ENCODING bind option	124
	EXPLAIN bind option	125
	EXTENDEDINDICATOR bind option	127
	FILTER bind option	128
	FLAG bind option.	128
	GENERIC bind option	129
	GETACCELARCHIVE bind option	130
	IMMEDWRITE bind option	131
	ISOLATION bind option	133
	KEEPDYNAMIC bind option	134
	LIBRARY bind option	135

MEMBER bind option	136
OPTHINT bind option	137
OPTIONS bind option	138
OWNER bind option	138
PACKAGE bind option	139
PATH bind option	141
PATHDEFAULT bind option	142
PKLIST and NOPKLIST bind options	143
PLAN bind option.	144
PLANMGMT bind option	144
QUALIFIER bind option.	146
QUERYACCELERATION bind option	146
QUERYID bind option	148
RELEASE bind option	149
REOPT bind option	151
ROUNDING bind option	152
SQLERROR bind option.	153
SQLRULES bind option	154
SWITCH bind option.	155
VALIDATE bind option	156
Chapter 19. Specification of DB2 for z/OS bind options from IBM Data Server clients	
and drivers	157
Chapter 20. -CANCEL THREAD (DB2)	159
Chapter 21. /CHANGE (IMS)	165
Chapter 22. DCLGEN (DECLARATIONS GENERATOR) (DSN)	167
Chapter 23. /DISPLAY (IMS)	177
Chapter 24. -DISPLAY ACCEL (DB2).	181
Chapter 25. -DISPLAY ARCHIVE (DB2).	185
Chapter 26. -DISPLAY BUFFERPOOL (DB2)	187
Chapter 27. -DISPLAY DATABASE (DB2).	201
Chapter 28. -DISPLAY DDF (DB2).	221
Chapter 29. -DISPLAY FUNCTION SPECIFIC (DB2)	225
Chapter 30. -DISPLAY GROUP (DB2)	229
Chapter 31. -DISPLAY GROUPBUFFERPOOL (DB2)	233
Chapter 32. -DISPLAY LOCATION (DB2)	241
Chapter 33. -DISPLAY LOG (DB2).	245
Chapter 34. -DISPLAY PROCEDURE (DB2)	247
Chapter 35. -DISPLAY PROFILE (DB2).	251

Chapter 36. -DISPLAY RLIMIT (DB2).	253
Chapter 37. -DISPLAY THREAD (DB2)	255
Chapter 38. -DISPLAY TRACE (DB2).	271
Chapter 39. -DISPLAY UTILITY (DB2)	281
Chapter 40. DSN (TSO)	285
Chapter 41. DSNH (TSO CLIST).	289
Chapter 42. END (DSN)	319
Chapter 43. FREE PACKAGE (DSN)	321
Chapter 44. FREE PLAN (DSN)	325
Chapter 45. FREE QUERY (DSN)	327
Chapter 46. MODIFY admtproc,APPL=SHUTDOWN	331
Chapter 47. MODIFY admtproc,APPL=TRACE	333
Chapter 48. -MODIFY DDF (DB2)	335
Chapter 49. MODIFY irlmproc,ABEND (z/OS IRLM)	341
Chapter 50. MODIFY irlmproc,DIAG (z/OS IRLM)	343
Chapter 51. MODIFY irlmproc,PURGE (z/OS IRLM)	345
Chapter 52. MODIFY irlmproc,SET (z/OS IRLM)	347
Chapter 53. MODIFY irlmproc,STATUS (z/OS IRLM)	351
Chapter 54. -MODIFY TRACE (DB2)	357
Chapter 55. REBIND PACKAGE (DSN).	361
Chapter 56. REBIND PLAN (DSN).	367
Chapter 57. REBIND TRIGGER PACKAGE (DSN)	373
Chapter 58. -RECOVER BSDS (DB2)	377
Chapter 59. -RECOVER INDOUBT (DB2)	379
Chapter 60. -RECOVER POSTPONED (DB2)	383
Chapter 61. -REFRESH DB2,EARLY (DB2)	387
Chapter 62. -RESET GENERICLU (DB2)	389

Chapter 63. -RESET INDOUBT (DB2)	391
Chapter 64. RUN (DSN)	395
Chapter 65. -SET ARCHIVE (DB2).	399
Chapter 66. -SET LOG (DB2)	403
Chapter 67. -SET SYSPARM (DB2)	409
Chapter 68. SPUFI (DSN)	413
Chapter 69. /SSR (IMS)	415
Chapter 70. -START ACCEL (DB2)	417
Chapter 71. START admtproc	421
Chapter 72. /START (IMS)	423
Chapter 73. -START DATABASE (DB2).	425
Chapter 74. -START DB2 (DB2)	435
Chapter 75. -START DDF (DB2).	439
Chapter 76. -START FUNCTION SPECIFIC (DB2)	441
Chapter 77. START irlmproc (z/OS IRLM)	445
Chapter 78. -START PROCEDURE (DB2).	451
Chapter 79. -START PROFILE (DB2).	455
Chapter 80. -START RLIMIT (DB2).	457
Chapter 81. -START TRACE (DB2)	459
Chapter 82. -STOP ACCEL (DB2)	477
Chapter 83. STOP admtproc (z/OS)	479
Chapter 84. /STOP (IMS).	481
Chapter 85. -STOP DATABASE (DB2)	483
Chapter 86. -STOP DB2 (DB2)	491
Chapter 87. -STOP DDF (DB2)	493
Chapter 88. -STOP FUNCTION SPECIFIC (DB2)	497
Chapter 89. STOP irlmproc (z/OS IRLM)	501

Chapter 90. -STOP PROCEDURE (DB2)	503
Chapter 91. -STOP PROFILE (DB2)	507
Chapter 92. -STOP RLIMIT (DB2)	509
Chapter 93. -STOP TRACE (DB2)	511
Chapter 94. -TERM UTILITY (DB2).	523
Chapter 95. /TRACE (IMS)	527
Chapter 96. TRACE CT (z/OS IRLM)	529
<hr/>	
Part 3. Appendixes	533
Information resources for DB2 10 for z/OS and related products	535
Notices	537
Programming interface information	538
Trademarks	539
Terms and conditions for product documentation	539
Privacy policy considerations	540
Glossary	541
Index	543

About this information

This information describes numerous commands that system administrators, database administrators, and application programmers use. The commands are in alphabetical order for quick retrieval.

Throughout this information, “DB2®” means “DB2 10 for z/OS®”. References to other DB2 products use complete names or specific abbreviations.

Important: To find the most up to date content, always use IBM® Knowledge Center, which is continually updated as soon as changes are ready. PDF manuals are updated only when new editions are published, on an infrequent basis.

This information assumes that your DB2 subsystem is running in DB2 10 new-function mode.

Availability of new function in DB2 10

Generally, new SQL capabilities, including changes to existing functions, statements, and limits, become available only in new-function mode, unless explicitly stated otherwise. Exceptions to this general statement include optimization and virtual storage enhancements, which are also available in conversion mode unless stated otherwise. In DB2 Version 8 and DB2 9, most utility functions were available in conversion mode. However, for DB2 10, most utility functions become available in new-function mode.

Who should read this information

This information presents reference information for people involved in system administration, database administration, and operation. It presents detailed information on commands, including syntax, option descriptions, and examples for each command.

DB2 Utilities Suite for z/OS

Important: In DB2 10, the DB2 Utilities Suite for z/OS is available as an optional product. You must separately order and purchase a license to such utilities, and discussion of those utility functions in this publication is not intended to otherwise imply that you have a license to them.

DB2 Utilities Suite for z/OS can work with DB2 Sort for z/OS and the DFSORT program. You are licensed to use DFSORT in support of the DB2 utilities even if you do not otherwise license DFSORT for general use. If your primary sort product is not DFSORT, consider the following informational APARs mandatory reading:

- II14047/II14213: USE OF DFSORT BY DB2 UTILITIES
- II13495: HOW DFSORT TAKES ADVANTAGE OF 64-BIT REAL ARCHITECTURE

These informational APARs are periodically updated.

Related concepts:

 [DB2 utilities packaging \(DB2 Utilities\)](#)

Terminology and citations

When referring to a DB2 product other than DB2 for z/OS, this information uses the product's full name to avoid ambiguity.

The following terms are used as indicated:

DB2 Represents either the DB2 licensed program or a particular DB2 subsystem.

Tivoli® OMEGAMON® XE for DB2 Performance Expert on z/OS

Refers to any of the following products:

- IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS
- IBM Tivoli OMEGAMON XE for DB2 Performance Monitor for z/OS
- IBM DB2 Performance Expert for Multiplatforms and Workgroups
- IBM DB2 Buffer Pool Analyzer for z/OS

C, C++, and C language

Represent the C or C++ programming language.

CICS® Represents CICS Transaction Server for z/OS.

IMS™ Represents the IMS Database Manager or IMS Transaction Manager.

MVS™ Represents the MVS element of the z/OS operating system, which is equivalent to the Base Control Program (BCP) component of the z/OS operating system.

RACF®

Represents the functions that are provided by the RACF component of the z/OS Security Server.

Accessibility features for DB2 10 for z/OS

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in z/OS products, including DB2 10 for z/OS. These features support:

- Keyboard-only operation.
- Interfaces that are commonly used by screen readers and screen magnifiers.
- Customization of display attributes such as color, contrast, and font size

Tip: The IBM Knowledge Center (which includes information for DB2 for z/OS) and its related publications are accessibility-enabled for the IBM Home Page Reader. You can operate all features using the keyboard instead of the mouse.

Keyboard navigation

For information about navigating the DB2 for z/OS ISPF panels using TSO/E or ISPF, refer to the *z/OS TSO/E Primer*, the *z/OS TSO/E User's Guide*, and the *z/OS ISPF User's Guide*. These guides describe how to navigate each interface, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Related accessibility information

IBM and accessibility

See the *IBM Accessibility Center* at <http://www.ibm.com/able> for more information about the commitment that IBM has to accessibility.

How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 for z/OS documentation.

Send your comments by email to db2zinfo@us.ibm.com and include the name of the product, the version number of the product, and the number of the book. If you are commenting on specific text, please list the location of the text (for example, a chapter and section title or a help topic title).

How to read syntax diagrams

Certain conventions apply to the syntax diagrams that are used in IBM documentation.

Apply the following rules when reading the syntax diagrams that are used in DB2 for z/OS documentation:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.
 - The \blacktriangleright — symbol indicates the beginning of a statement.
 - The — \blacktriangleright symbol indicates that the statement syntax is continued on the next line.
 - The \blacktriangleright — symbol indicates that a statement is continued from the previous line.
 - The — \blacktriangleleft symbol indicates the end of a statement.
- Required items appear on the horizontal line (the main path).

\blacktriangleright —*required_item*— \blacktriangleleft

- Optional items appear below the main path.

\blacktriangleright —*required_item*— \blacktriangleleft
 └─*optional_item*—┘

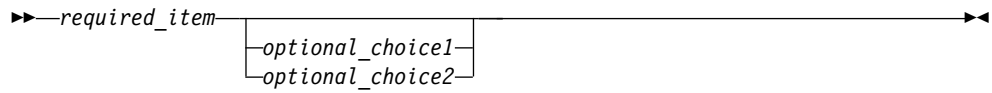
If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

\blacktriangleright —*required_item*— \blacktriangleleft
 ┌─*optional_item*—┐

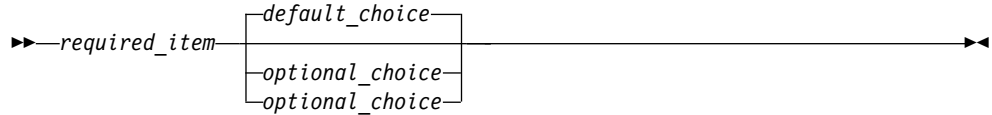
- If you can choose from two or more items, they appear vertically, in a stack. If you *must* choose one of the items, one item of the stack appears on the main path.

\blacktriangleright —*required_item*— \blacktriangleleft
 ┌─*required_choice1*—┐
 └─*required_choice2*—┘

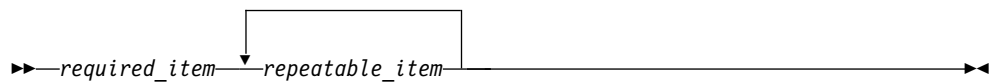
If choosing one of the items is optional, the entire stack appears below the main path.



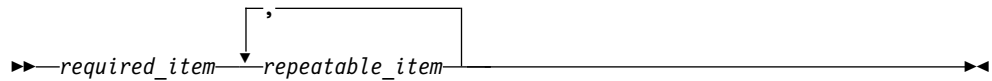
If one of the items is the default, it appears above the main path and the remaining choices are shown below.



- An arrow returning to the left, above the main line, indicates an item that can be repeated.



If the repeat arrow contains a comma, you must separate repeated items with a comma.



A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.




fragment-name:




- With the exception of XPath keywords, keywords appear in uppercase (for example, FROM). Keywords must be spelled exactly as shown. XPath keywords are defined as lowercase names, and must be spelled exactly as shown. Variables appear in all lowercase letters (for example, *column-name*). They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

Related concepts:

Syntax rules for DB2 commands

 [DB2 online utilities \(DB2 Utilities\)](#)

 [DB2 stand-alone utilities \(DB2 Utilities\)](#)

Related information:

DB2 and related commands

Part 1. About DB2 and related commands


Use the DB2 for z/OS and related commands to execute database administrative functions.

These topics provide detailed reference information for DB2 and related commands, including the environment in which each command is issued, the privileges and authorities that are required to issue each command, syntax and option descriptions, usage information, and examples.

Syntax diagrams for commands

For information about the conventions for syntax diagrams in IBM documentation, see [How to read syntax diagrams](#) ().

Related tasks:

 [Controlling DB2 operations by using commands \(DB2 Administration Guide\)](#)

Related information:

[DB2 and related commands](#)

Chapter 1. Privileges and authorization IDs for DB2 commands

A command can be issued by an individual user, by a program that runs in batch mode, or by an IMS or CICS transaction. The term *process* describes any of these initiators.

DB2 processes are represented by a set of identifiers (IDs), which are called authorization IDs. What the process can do with DB2 is determined by the privileges and authorities that are held by its identifiers.



If RACF is active, IDs that issue commands from logged-on MVS consoles or from TSO SDSF must have appropriate RACF authorization for DB2 commands, or the primary authorization IDs must have DB2 authorization to issue commands.

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

You can use DB2 authorization to check DB2 commands that are issued from a DSN session under TSO or DB2I by using primary authorization IDs, secondary authorization IDs, and role, if the commands are running in a trusted context with an associated role.

SQL IDs do not affect most DB2 and related commands.

Related concepts:

-  [Authorization IDs \(Managing Security\)](#)
-  [Privileges and authorities \(Managing Security\)](#)



Chapter 2. Types of commands

DB2 supports several different types of commands that you can use to complete database administration tasks.

The commands are divided into the following categories:

The DSN command and its subcommands	DSN is the DB2 command processor and executes as a TSO command processor.
DB2 commands	You can use DB2 commands to control most of the operational environment.
IMS commands	You can use IMS commands to control IMS connections as well as to start and stop connections to DB2 and display activity on the connections.
CICS attachment facility commands	You can use CICS commands to control CICS connections as well as to start and stop connections to DB2 and display activity on the connections.
“Administrative task scheduler commands” on page 9	You can use administrative task scheduler commands to start, stop, and change the administrative task scheduler.
z/OS IRLM commands	You can use z/OS Internal Resource Lock Manager (IRLM) commands to start, stop, and change the IRLM.
TSO CLISTs	You can use Time Sharing Option (TSO) commands to perform TSO tasks such as prepare and execute programs under TSO.

Related tasks:

-  [Controlling DB2 operations by using commands \(DB2 Administration Guide\)](#)
-  [Submitting work to DB2 \(DB2 Administration Guide\)](#)

The DSN command and its subcommands

DSN is the DB2 command processor and executes as a TSO command processor.

Introductory concepts

Commands for controlling DB2 and related facilities (Introduction to DB2 for z/OS)


TSO attachment facility (Introduction to DB2 for z/OS)

Environment

All of the DSN subcommands, except SPUFI, run under DSN in either the foreground or background, and all, except END, also run under DB2 Interactive (DB2I). SPUFI runs only in the foreground under ISPF.

ABEND (DSN)	The DSN subcommand ABEND causes the DSN session to terminate with abend completion code X'04E' and reason code of X'00C50101'. Important: The ABEND subcommand is used for diagnostic purposes only, and is intended to be used only under the direction of IBM Software Support. Use it only when diagnosing a problem with DSN or DB2.
BIND PACKAGE (DSN)	The DSN subcommand BIND PACKAGE builds an application package. DB2 records the description of the package in the catalog tables and saves the prepared package in the directory.
BIND PLAN (DSN)	The DSN subcommand BIND PLAN builds an application plan. All DB2 programs require an application plan to allocate DB2 resources and support SQL requests made at run time.
BIND QUERY (DSN)	The DSN subcommand BIND QUERY reads the statement text, default schema, and a set of bind options from every row of DSN_USERQUERY_TABLE, and information from correlated EXPLAIN table rows. When LOOKUP(NO) is in effect, DB2 inserts the pertinent data into certain catalog tables.
DSN (TSO)	The TSO command DSN starts a DSN session.
END (DSN)	The DSN subcommand END is used to end the DSN session and return to TSO.
FREE PACKAGE (DSN)	The DSN subcommand FREE PACKAGE can be used to delete a specific version of a package, all versions of a package, or whole collections of packages.
FREE PLAN (DSN)	The DSN subcommand FREE PLAN deletes application plans from DB2.
FREE QUERY (DSN)	The DSN subcommand FREE QUERY removes from certain catalog tables for one or more queries. If any of the specified queries are in the dynamic statement cache, FREE QUERY purges them from the dynamic statement cache.
DCLGEN (DECLARATIONS GENERATOR) (DSN)	The declarations generator (DCLGEN) produces an SQL DECLARE TABLE statement and a COBOL, PL/I, or C data declaration for a table or a view named in the catalog.
REBIND PACKAGE (DSN)	The DSN subcommand REBIND PACKAGE rebinds an application package when you make changes that affect the package, but have not changed the SQL statements in the program.
REBIND PLAN (DSN)	The DSN subcommand REBIND PLAN rebinds an application plan when you make changes to the attributes of the plan, such as the package list.
REBIND TRIGGER PACKAGE (DSN)	The DSN subcommand REBIND TRIGGER PACKAGE rebinds a package that was created when DB2 executed a CREATE TRIGGER statement.
RUN (DSN)	The DSN subcommand RUN executes an application program, which can contain SQL statements.
SPUFI (DSN)	The DSN subcommand SPUFI executes the SQL processor using file input.

Related concepts:

 [DSN command processor \(DB2 Application programming and SQL\)](#)

Related tasks:

 [Issuing commands from TSO terminals \(DB2 Administration Guide\)](#)

DB2 commands

You can use DB2 commands to control most of the operational environment.

Environment

The command START DB2 can be issued only from a z/OS console or TSO SDSF. All other DB2 commands can be issued from the following environments:

- z/OS consoles
- TSO terminals, by any of the following methods:
 - Issuing the DSN command from the TSO READY prompt
 - Entering commands in the DB2 Commands ISPF panel of DB2I
- IMS terminals
- Authorized CICS terminals

You can issue many commands from the background within batch programs, such as the following types of programs:

- z/OS application programs
- Authorized CICS programs
- IMS programs
- APF-authorized programs, such as a terminal monitor program (TMP)
- IFI application programs

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

For detailed descriptions of the DB2 commands, see the commands with names that are preceded by the recognition character “-” and identified by “(DB2)” under Part 2, “DB2 and related commands,” on page 53.

Extended MCS consoles

The extended MCS console feature of z/OS lets a z/OS system have more than 99 consoles. Because DB2 supports extended MCS consoles, messages returned from a DB2 command are routed to the extended MCS console that issued the command.

Completion messages

Message DSN9022I indicates the normal end of DB2 command processing; DSN9023I indicates the abnormal end of DB2 command processing.


Related concepts:

[Syntax rules for DB2 commands](#)

Related information:

[About DB2 and related commands](#)

[DB2 and related commands](#)

 [DSN9022I \(DB2 Messages\)](#)



IMS commands

You can use IMS commands to control IMS connections as well as to start and stop connections to DB2 and display activity on the connections.

Environment: You can issue an IMS command from an IMS terminal or you can invoke an IMS transaction or command by using the DB2-supplied stored procedures `DSNAIMS` or `DSNAIMS2`. `DSNAIMS2` has the same functions as `DSNAIMS` but also provides multi-segment input support for IMS transactions.

<code>/CHANGE (IMS)</code>	The IMS command <code>/CHANGE</code> resets an indoubt unit of recovery as identified by the <code>OASN</code> keyword of the <code>/DISPLAY</code> command. That command deletes the item from the standpoint of IMS, but it does not communicate to DB2.
<code>/DISPLAY (IMS)</code>	The IMS command <code>/DISPLAY</code> displays the status of the connection between IMS and an external subsystem (as well as all application programs communicating with the external subsystem), or the outstanding recovery units that are associated with the subsystem.
<code>/SSR (IMS)</code>	The IMS <code>/SSR</code> command allows the IMS operator to enter an external subsystem command.
<code>/START (IMS)</code>	The IMS <code>/START</code> command (with the <code>SUBSYS</code> parameter) makes the connection between IMS and the specified external subsystem available. Establishing the connection allows application programs to access resources managed by the external subsystem.
<code>/STOP (IMS)</code>	The IMS <code>/STOP</code> command (with the <code>SUBSYS</code> parameter) prevents application programs from accessing external subsystem resources.
<code>/TRACE (IMS)</code>	The IMS <code>/TRACE</code> command directs and controls the IMS capabilities for tracing internal IMS events. It also starts, stops, and defines the activity to be monitored by the IMS Monitor.

Related reference:






-  [DSNAIMS stored procedure \(DB2 SQL\)](#)
-  [DSNAIMS2 stored procedure \(DB2 SQL\)](#)

CICS attachment facility commands

You can use CICS commands to control CICS connections as well as to start and stop connections to DB2 and display activity on the connections.

Environment: Each CICS attachment facility command can be issued from a CICS terminal.

Related information:

-  [Issuing commands to DB2 using the DSNB transaction \(CICS Transaction Server for z/OS\)](#)
-  [DSNB DISCONNECT \(CICS Transaction Server for z/OS\)](#)
-  [DSNB DISPLAY \(CICS Transaction Server for z/OS\)](#)
-  [DSNB STOP \(CICS Transaction Server for z/OS\)](#)
-  [DSNB STRT \(CICS Transaction Server for z/OS\)](#)


Administrative task scheduler commands

You can use administrative task scheduler commands to start, stop, and change the administrative task scheduler.

Environment: Each administrative task scheduler command can be issued from a z/OS console.

Command	Description
MODIFY admtproc,APPL=SHUTDOWN	The MODIFY <i>admtproc</i> , APPL=SHUTDOWN command stops the administrative task scheduler from accepting requests and starting new task executions. It also shuts down the administrative task scheduler.
MODIFY admtproc,APPL=TRACE	The MODIFY <i>admtproc</i> , APPL=TRACE command starts or stops traces in the administrative task scheduler.
START admtproc	The START <i>admtproc</i> command starts the scheduler that is specified in the <i>admtproc</i> parameter
STOP admtproc (z/OS)	The STOP <i>admtproc</i> command stops the administrative task scheduler that is specified in the <i>admtproc</i> parameter.

Related tasks:

 [Scheduling administrative tasks \(DB2 Administration Guide\)](#)

z/OS IRLM commands

You can use z/OS Internal Resource Lock Manager (IRLM) commands to start, stop, and change the IRLM.

Environment: Each z/OS IRLM command can be issued from a z/OS console.

MODIFY irlmproc,ABEND (z/OS IRLM)	The MODIFY <i>irlmproc</i> , ABEND command terminates IRLM abnormally. IRLM processes this command even if a DB2 subsystem is identified to it.
MODIFY irlmproc,DIAG (z/OS IRLM)	The MODIFY <i>irlmproc</i> , DIAG command initiates diagnostic dumps for IRLM subsystems.)
MODIFY irlmproc,PURGE (z/OS IRLM)	The MODIFY irlmproc,PURGE command releases IRLM locks retained due to a DB2, IRLM, or system failure.
MODIFY irlmproc,SET (z/OS IRLM)	The MODIFY irlmproc,SET command dynamically sets various IRLM operational parameters
MODIFY irlmproc,STATUS (z/OS IRLM)	The MODIFY irlmproc,STATUS command displays information for one or more subsystems connected to the IRLM that is specified using <i>irlmproc</i> .

TSO CLISTS

You can use Time Sharing Option (TSO) commands to perform TSO tasks such as prepare and execute programs under TSO.


Chapter 41, "DSNH (TSO CLIST)," on page 289

The DSNH command procedure (a TSO CLIST) is a powerful yet easy method of preparing an application program for execution.

DSNU

The DSNU command procedure can be executed directly or by using DB2L.

Related tasks:

 Invoking a DB2 utility by using the DSNU CLIST command in TSO (DB2 Utilities)

Chapter 3. Syntax rules for DB2 commands

Standard syntax rules apply to DB2 commands.

General conventions for syntax diagrams

For information about the conventions for syntax diagrams in IBM documentation, see [How to read syntax diagrams](#) ().

Conventions for DB2 commands syntax

For the complete syntax of specific commands, see the “Syntax” section for each command under Part 2, “DB2 and related commands,” on page 53.

The following figure illustrates the overall pattern of DB2 command syntax.

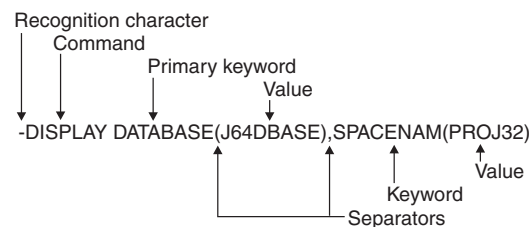


Figure 1. The format and parts of a DB2 command

A DB2 command contains the following parts:

Recognition character

Shown as a hyphen throughout this information, with the following exceptions:

- If the command is issued from a z/OS console, the recognition character must be the *command prefix*.

The command prefix can be up to eight characters. The default is '-DSN1'. However, the majority of examples in this information assume that the command prefix has been defined as a hyphen (-). Examples involving members of a data sharing group demonstrate the use of multi-character command prefixes, such as -DB1G.

Inserting a space between the command prefix and the command is optional. For example, you can use either one of the following formats:

```
-DB1GDIS THREAD(*)
```

```
-DB1G DIS THREAD(*)
```

Using a space makes it easier for users to identify the command, especially when the command prefix has multiple characters.

The command prefix can be defined at installation time.

- If the command is issued from an IMS terminal, the recognition character must be the command recognition character (CRC). The command recognition character is defined in the IMS SSM PROCLIB member.

- If the command is issued from a CICS terminal or under the DSN command processor, the recognition character must be a hyphen.

Command name

The name of the command. Command names have abbreviations, which are provided in the description of each command.

Operands

Combinations of keywords and parameters that can be specified for the command.

Keywords

Sometimes called command options. Keywords can be required or optional. They must be entered exactly as shown in the descriptions of the commands.

Parameters

A keyword can have zero or more parameters. A parameter list, if present, must be enclosed in parentheses.

Separators

These can be one or more blanks or commas. An open parenthesis marks the beginning of a parameter list; no separator is needed. Optionally, an equal sign can be used to separate a single parameter from its keyword without using parentheses.

Characters with special meanings in DB2 commands

The following characters have special meanings when they are used in DB2 commands:

Blank or blanks ()

A separator. Single blanks and multiple blanks are equivalent, except in strings that are enclosed between apostrophes.

Comma (,)

A comma is a separator.

Apostrophe (')

The usual SQL string constant delimiter, and marks the beginning or end of a string constant in SQL. (In COBOL programs only, the QUOTESQL precompiler option allows you to choose the quotation mark as the SQL string delimiter; the apostrophe is then the SQL escape character.)

Letters that are not in string constants are changed to uppercase. Two successive apostrophes in a string constant are changed to one apostrophe. Blanks, commas, equal signs, and parentheses in string constants are treated as literal characters, and are not recognized as separators.

Use apostrophes to enclose options that must have lowercase characters. Beware of commands that do not convert lowercase characters to uppercase because entering lowercase letters might cause a JCL error or an abend. Similarly, entering uppercase letters where lowercase is required (UNIX Services, for example) might produce incorrect results. For more information, see Starting a system task from a console (MVS System Commands).

There is an exception to the rule about changing letters to uppercase. If the CODED CHARACTER SET option is set to 930 or 5026 during installation, the letters are not folded to uppercase, whether in an SQL string constant or not.

If a keyword value contains leading or following asterisk (*) or underscore (_) pattern-matching characters, and the characters in the keyword value are enclosed in apostrophes, the leading or following pattern-matching characters must also be enclosed in those apostrophes.

Quotation mark (")

The SQL escape character, and marks the beginning or end of an SQL delimited identifier. (In COBOL programs only, the QUOTESQL precompiler option allows you to choose the apostrophe as the SQL escape character; the double quotation mark is then the SQL string delimiter.)

Within a string delimited by quotation marks, two successive quotation marks are changed to one. Other rules are the same as for SQL string constants.

Equal sign (=)

Separates a single parameter from a keyword. Thus, an equal sign is used as a separator for keywords that have only one parameter. An equal sign can be used for keywords with multiple parameters when only one member of the parameter list is specified.

open parenthesis (()

An open parenthesis marks the beginning of a parameter list.

Close parenthesis ())

The end of a parameter list.

Colon (:)

An inclusive range. For example, (A:D) means the same as (A,B,C,D); (1:5) means (1,2,3,4,5). The colon can be used this way only in commands where this operation is specifically permitted.

Asterisk (*)

The meaning depends on the context:

* A single asterisk as a *keyword-value* indicates "all". For example:
-DISPLAY UTILITY (*)

**keyword-value*

An asterisk as the first character of a *keyword-value* indicates that a match for the value will be satisfied when all characters following the * are the same. For example: (*BCD)

*beginning-of-keyword-value*end-of-keyword-value*

An intermediate asterisk indicates that a match for the value will be satisfied when all characters preceding and all characters following the asterisk are the same. For example: (ABC*EFG)

*keyword-value**

An asterisk as the final character of a *keyword-value* indicates that a match will for the value will be satisfied when all characters preceding the asterisk are the same. For example: (ABC*)

*beginning-of-keyword-value*middle-of-keyword-value*end-of-keyword-value** Asterisks used as the first, intermediate and final characters in a string are also valid. For example: (*BCD*FGH*)

For example, DISPLAY UTILITY (*) displays the status of all utilities; DISPLAY UTILITY (R2*) displays the status of all utilities whose identifiers begin with R2.

The asterisk pattern-matching character is available to all DB2 commands, but not all DB2 commands support it. The asterisk can be used this way only in commands in which the pattern-matching operation is specifically supported.

| **Underscore (_)**

| An underscore indicates that at the position in the keyword value where
| the underscore occurs, any character is a match for that value. For
| example, A_C matches any three-character keyword value with A as the
| first character and C as the third character.

| **The two-character string NO**

Negates the keyword that follows.

A negated keyword means the opposite of the keyword itself, and is often used to override a keyword default. In keywords that have no opposite meaning, the initial characters "NO" can be merely part of the keyword itself; for example, in NODE.

| **Related reference:**

DB2 commands

Chapter 4. Syntax rules for DSN subcommands

The syntax rules for DSN subcommands conform to standard TSO command parsing conventions.


For general information about the syntax of TSO/E commands and subcommands, see TSO/E commands and subcommands.

To continue a subcommand on the next line in the DSN processor, type either a hyphen (-) or a plus sign (+) at the end of the current line. If you use a plus sign, precede it by at least one blank character to prevent the concatenation of character strings from line to line. Using a plus sign causes TSO/E to delete leading delimiters (blanks, commas, tabs, and comments) on the continuation line, and reduces the overall size of the command.

Important: The names of the DSN command and its subcommands cannot be abbreviated. Abbreviations for some keywords are supported for compatibility with prior DB2 releases. However, avoid abbreviating keywords in the DSN command and its subcommand to avoid potential problems.

Related reference:

The DSN command and its subcommands

 [How to read syntax diagrams \(\)](#)

Chapter 5. Scope of commands in data sharing

In a data sharing environment, the *scope* of a command is the breadth of its impact.

All commands have one of the following scopes:

Member or Local

Many commands used in a data sharing environment have *member* (or *local*) scope because they affect only the DB2 subsystem for which they are issued. For example, a CANCEL THREAD command cancels the specified threads for the member that is identified by the command prefix.

Group Other commands have *group* scope because they affect an object in such a way that affects all members of the group. For example, a STOP DATABASE command issued from any member of the group stops that database for all members of the group.

The following commands have group scope:



ALTER GROUPBUFFERPOOL (DB2)	FREE PLAN (DSN)
BIND PACKAGE (DSN)	MODIFY irlmproc,DIAG (z/OS IRLM)
BIND PLAN (DSN)	REBIND PACKAGE (DSN)
DCLGEN (DSN)	REBIND PLAN (DSN)
DISPLAY DATABASE (DB2)	REBIND TRIGGER PACKAGE (DSN)
DISPLAY GROUP (DB2)	START DATABASE (DB2)
DISPLAY GROUPBUFFERPOOL (DB2)	STOP DATABASE (DB2)
FREE PACKAGE (DSN)	

The following commands have either group or member scope, depending on the options that you specify:

ARCHIVE LOG (DB2)	START FUNCTION SPECIFIC (DB2)
DISPLAY FUNCTION SPECIFIC (DB2)	START PROCEDURE (DB2)
DISPLAY PROCEDURE (DB2)	START TRACE (DB2)
DISPLAY THREAD (DB2)	STOP FUNCTION SPECIFIC (DB2)
DISPLAY TRACE (DB2)	STOP PROCEDURE (DB2)
DISPLAY UTILITY (DB2)	STOP TRACE (DB2)
MODIFY irlmproc,SET (z/OS IRLM)	TERM UTILITY (DB2)
MODIFY irlmproc,STATUS (z/OS IRLM)	

All other commands have member scope. For information about the specific data sharing scope of each command, see the “Environment” sections in the topics under DB2 and related commands .

Related concepts:

-  Commands for data sharing environments (DB2 Data Sharing Planning and Administration)
-  Command scope (DB2 Data Sharing Planning and Administration)

Chapter 6. Output from DB2 commands

Several factors affect the amount of output you can receive from a DB2 command.

The amount of output that you receive from a DB2 command is always less than 256 KB. The following factors determine the maximum amount of output that is returned:

- The amount of storage available to your DB2 subsystem or to an individual command.
- The environment from which you issue the DB2 command.
For example, if you issue a DB2 command from an IMS console, you can receive no more than 32 KB of output.
- For DISPLAY DATABASE, the value of the LIMIT keyword.
- For DISPLAY THREAD, the number of lines of output. DISPLAY THREAD displays at most 254 lines of output.

Related reference:

-DISPLAY DATABASE (DB2)

-DISPLAY THREAD (DB2)

Related information:

DB2 and related commands

Chapter 7. Command line processor


The command line processor is a program that runs under z/OS UNIX System Services. You can use the command line processor to execute SQL statements, bind DBRMs that are stored in HFS files into packages, call stored procedures, and perform XML schema repository operations.


The command line processor is an alternative to using SPUFI or DSNTEP2 to execute SQL statements.

You can use the command line processor in one of the following ways:

- In interactive mode, in which you request a single operation, and the command line processor executes that operation and returns the result.
- In batch mode, in which you pass a file to the command processor that contains all of the operations that you want to execute.

Related tasks:

 Binding a DBRM that is in an HFS file to a package or collection (DB2 Application programming and SQL)

 Setting up your system to use the DB2 command line processor (DB2 Installation and Migration)

Related reference:

 Statements (DB2 SQL)

Tutorial: Using the command line processor

You can interactively query and modify DB2 data, call stored procedures, and issue z/OS UNIX System Services commands through the command line processor. This lesson demonstrates how to perform those tasks.

Before you start this tutorial, make sure that your system administrator has customized your environment appropriately:

- Your z/OS UNIX System Services .profile file needs to be set up so that you can use the command line processor.
- The command line processor properties file needs to include the alias MYALIAS01 for your connection information. A line of the following form must exist in the command line processor properties file.
`MYALIAS01=connection-url,user-id,password`
“Command line processor CONNECT command” on page 32 describes the syntax for *connection-url*, *user-id*, and *password*.
- An alias of db2 for the command that starts the command line processor must be created:
`alias db2="java com.ibm.db2.clp.db2"`
- The database server to which you connect must not be a data sharing system.

See Setting up your system to use the DB2 command line processor (DB2 Installation and Migration) for more information.

This tutorial demonstrates how to perform the same tasks in the command line processor that you can perform in SPUFI or DSNTEP2. In addition, this tutorial demonstrates how to call stored procedures from the command line processor.

In this tutorial, you will learn how to do the following command line processor tasks:

- Start and stop the command line processor
- List and modify command line processor options
- Connect to a database server
- Execute SQL statements to create, populate, query and drop a table
- Create a stored procedure
- Call a stored procedure

Conventions used in this tutorial: Although command line processor input is mostly case insensitive, SQL statements and command line processor commands are shown in uppercase for consistency with other DB2 operations.

1. Start the command line processor, and set command line processor options.

a. Start the command line processor by typing the predefined alias:

```
db2
```

The command line processor starts, and the following prompt is displayed:

```
db2 =>
```

b. Connect to a DB2 database server using the predefined connection alias:

```
CONNECT TO MYALIAS01
```

The connection is established, and the following information about the connection is displayed:

```
Database Connection Information
Database server      =Db2 server version
SQL authorization ID =User ID
JDBC Driver         =JDBC driver name and version
DSNC101I : The "CONNECT" command completed successfully.
db2 =>
```

Information about the JDBC driver is displayed because the command line processor is a Java™ database application.

c. Check the current settings of the command line processor options to determine whether they are right for your command line processor session:

```
LIST COMMAND OPTIONS
```

A list of command line processor options and their current settings is displayed:

Option	Description	Current Setting
-a	Display SQLCA	OFF
-c	Auto-Commit	ON
-f	Read from input file	OFF
-o	Display output	ON
-s	Stop execution on command error	OFF
-t	Set statement termination character	OFF
-v	Echo current command	OFF
-x	Suppress printing of column headings	OFF
-z	Save all output to output file	OFF

```
DSNC101I : The "LIST COMMAND OPTIONS" command completed successfully.
```

```
db2 =>
```

The options are described in Table 2 on page 46.

- d. Suppose that you want to perform commit operations explicitly, so you want to disable autocommit. To do that, you need to set the c option to OFF:

```
UPDATE COMMAND OPTIONS USING c OFF
```

The following message is displayed, which indicates that you have successfully disabled the autocommit option:

```
DSNC101I : The "UPDATE COMMAND OPTIONS" command completed successfully.
```

```
db2 =>
```

- 2. Run SQL statements to query and modify data.

- a. Run an SQL CREATE statement to create a copy of sample employee table:

```
CREATE TABLE MY_EMP LIKE DSN8A10.EMP
```

- b. Run an SQL INSERT statement to populate your copy of the sample employee table from the sample employee table:

```
INSERT INTO MY_EMP SELECT * FROM DSN8A10.EMP
```

- c. Commit the previously run SQL operations:

```
COMMIT
```

- d. To make the output easier to read on the screen, set the maximum width of any column data that is returned to 12:

```
CHANGE MAXCOLUMNWIDTH TO 12
```

- e. Run this SELECT statement, to ensure that the table was created and the rows were inserted successfully:

```
SELECT FIRSTNME, MIDINIT, LASTNAME FROM MY_EMP ORDER BY LASTNAME
```

Rows from table MY_EMP are displayed:

FIRSTNME	MIDINIT	LASTNAME
BRUCE		ADAMSON
ROY	R	ALONZO
DAVID		BROWN
JOHN	B	GEYER
JASON	R	GOUNOT
...		
MICHAEL	L	THOMPSON
JAMES	H	WALKER
HELENA		WONG
KIYOSHI		YAMAMOTO
MASATOSHI	J	YOSHIMURA

42 record(s) selected

```
db2 =>
```

- 3. Terminate the command line processor, and restart it with a different set of options.

- a. Terminate the command line processor:

```
Terminate the command line processor:
```

```
TERMINATE
```

Control returns to z/OS UNIX System Services.

- b. Start the command line processor with the statement termination character set to the pound sign (#), so that you can terminate statements within an SQL PL procedure with semicolons (;):

```
db2 -td#
```

```
db2 =>
```

- c. Change the command line processor option for saving output to a file. Issue the following command to direct the command line processor to save the output to a file named clplog.txt, and to display the output on the screen:
UPDATE COMMAND OPTIONS USING z ON clplog.txt#
- d. Connect to a DB2 database server using the predefined connection alias:
CONNECT to MYALIAS01#

4. Create and call a stored procedure. Save the output to a file.

- a. Create stored procedure CALC_SAL.

Type the following lines on the display. Press Enter after you type each line. The command line processor considers all text up to the pound sign (#) to be a single statement.

```
CREATE PROCEDURE CALC_SAL(INOUT EMPNO_IN CHAR(6),
  IN PCT_RAISE DECIMAL(2,2),
  OUT EMPNAME VARCHAR(15),
  OUT SAL_CALC DECIMAL(9,2))
LANGUAGE SQL
BEGIN
  DECLARE CURRENT_SALARY DECIMAL(9,2) DEFAULT 0;
  SELECT LASTNAME, SALARY INTO EMPNAME, CURRENT_SALARY FROM MY_EMP
  WHERE EMPNO=EMPNO_IN;
  SET SAL_CALC=CURRENT_SALARY*PCT_RAISE+CURRENT_SALARY;
END#
```

- b. Call the CALC_SAL stored procedure to find out what the salary of employee '000100' will be if you give the employee a 5% raise:

```
CALL CALC_SAL('000100',.05,?,?)#
```

The following output is displayed:

```
Value of output parameters
-----
Parameter Name : EMPNO_IN
Parameter Value : 000100
Parameter Name : EMPNAME
Parameter Value : SPENSER
Parameter Name : SAL_CALC
Parameter Value : 27457.50
```

DSNC101I : The "CALL" command completed successfully.

db2 =>

- c. Run the z/OS UNIX System Services cat command from within the command line processor to view the contents of the clplog.txt file. The exclamation mark (!) at the beginning of a statement indicates that it is a z/OS UNIX System Services command.

```
!cat clplog.txt#
```

The contents of the clplog.txt file are displayed:

DSNC101I : The "UPDATE COMMAND OPTIONS" command completed successfully.

```
Value of output parameters
-----
Parameter Name : EMPNO_IN
Parameter Value : 000100
Parameter Name : EMPNAME
Parameter Value : SPENSER
Parameter Name : SAL_CALC
Parameter Value : 27457.50
```

DSNC101I : The "CALL" command completed successfully.

db2 =>

5. Clean up all objects that you created for this tutorial, and exit the command line processor.
 - a. Change the command line processor option for saving output to a file back to its original setting of OFF:
UPDATE COMMAND OPTIONS USING z OFF#
 - b. Drop the SQL objects that you created, and commit the changes.
DROP TABLE MY_EMP#
DROP PROCEDURE CALC_SAL#
COMMIT#
 - c. Delete the file that contains command line processor output.
!rm clplog.txt#
 - d. Terminate the command line processor session:
TERMINATE#
The command line processor session ends.

Running the command line processor

If you use z/OS UNIX System Services, you can run the command line processor.

Before you can use the command line processor, your z/OS UNIX System Services session needs to be set up to locate the command line processor code and the properties file that contains command line processor start-up options. See Setting up your system to use the DB2 command line processor (DB2 Installation and Migration) for details.

To run the command line processor:


1. Start the command line processor. See Command line processor start command.
2. Connect to a DB2 server using the CONNECT command.
You can disconnect from one server and connect to another server during the command line processor session by using the DISCONNECT command and another CONNECT command.
3. Run command line processor commands. The command line processor lets you perform the following types of tasks:

Task	Command
Perform COMMIT, ROLLBACK, change isolation level, or DESCRIBE operations	COMMIT ROLLBACK CHANGE ISOLATION DESCRIBE CALL DESCRIBE TABLE
Execute other SQL statements, including CALL	Command line processor SQL statements
Perform XML schema repository operations	REGISTER XMLSCHEMA ADD XMLSCHEMA COMPLETE XMLSCHEMA REMOVE XMLSCHEMA
Bind packages	BIND
List the tables on the DB2 server	LIST TABLES

Task	Command
View or modify command line processor options	LIST COMMAND OPTIONS UPDATE COMMAND OPTIONS
View or modify the format of the command line processor output	DISPLAY RESULT SETTINGS CHANGE MAXCOLUMNWIDTH CHANGE MAXLINESFROMSELECT
Add explanatory information to the output	ECHO
Run z/OS UNIX System Services commands	Command line processor z/OS UNIX System Services command
Display command line processor help	Command line processor help command

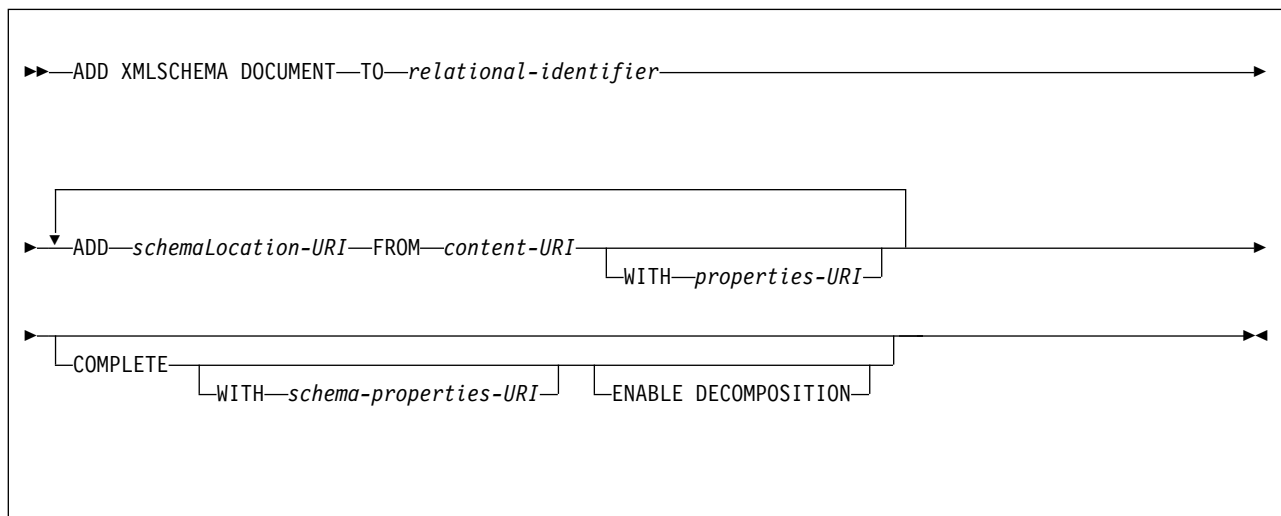
4. Stop the command line processor by running the TERMINATE command.

Related tasks:

 Setting up your system to use the DB2 command line processor (DB2 Installation and Migration)

Command line processor ADD XMLSCHEMA DOCUMENT command

The command line processor ADD XMLSCHEMA DOCUMENT command adds one or more XML schema documents to an existing but incomplete XML schema.



TO relational-identifier

Specifies the relational name of a registered, but incomplete, XML schema to which additional schema documents are to be added.

The schema of the relational identifier must be SYSXSR. If you specify a qualified name, the qualifier must be SYSXSR, such as, SYSXSR.MYSHEMA. If you specify an unqualified name, you need to execute the SQL statement SET SCHEMA="SYSXSR" before you execute the command. A name that is enclosed in quotation marks, such as "SYSXSR.MYSHEMA," is considered an unqualified name. Qualifiers or schema names that are enclosed in quotation marks are not folded to uppercase. Qualifiers or schema names that are not enclosed in quotation marks are folded to uppercase.

|
| **ADD *schemaLocation-URI***

| Specifies the schema location URI of an XML schema document that is to be
| added to this schema, as the document is likely to be referenced from another
| XML document.

|
| **FROM *content-URI***

| Specifies the URI where the actual content of an XML schema document is
| located. The command line processor supports only file URIs, which begin
| with file://.

|
| **WITH *properties-URI***

| Specifies that the URI of a properties document is to be associated with a
| schema document. The command line processor supports only file URIs, which
| begin with file://.

|
| **COMPLETE**

| Indicates that no more XML schema documents need to be associated with this
| XML schema, other than those that are indicated in this ADD XML SCHEMA
| DOCUMENT command. Validation of the schema is to be performed, and, if
| no errors are found, the schema is to be marked as usable. Until XML schema
| registration is completed, the schema is not visible to or usable in other SQL or
| XML commands.

|
| **WITH *schema-properties-URI***

| Specifies the URI of a properties document to be associated with this XML
| schema. The command line processor supports only file URIs, which begin
| with file://. A schema properties document can be specified only when the
| XML schema is also being completed.

|
| **ENABLE DECOMPOSITION**

| Indicates that this schema is to be used for the purpose of decomposing XML
| documents. This option can be specified only when the XML schema is also
| being completed.

|
| **Examples**

| **Example: Adding documents to an XML schema and completing schema
| registration**

| An XML schema that is referred to by the relational name
| JOHNDOE.PRODSHEMA1 already is registered. Two more documents
| need to be added to the list of schema documents associated with the
| schema. The first document is `http:// mycompany.com/xml/schema/
| order.xsd`, which is stored locally in `/u/temp`. The second document is
| `http:// othercompany.com/external/po.xsd`, which is also stored locally in
| `/u/temp`. Neither document has an associated properties file, and you are
| certain that no other documents are referenced by the schema. Issue these
| commands to add the documents and complete registration:

```
| ADD XMLSCHEMA DOCUMENT TO JOHNDOE.PRODSHEMA1  
| ADD http://mycompany.com/xml/schema/order.xsd  
| FROM file:///u/temp/order.xsd  
| ADD http://othercompany.com/external/po.xsd  
| FROM file:///u/temp/po.xsd  
| COMPLETE
```


| **Example: Adding documents to an XML schema without completing schema
| registration**

| An XML schema that is referred to by the relational name JOHNDOE.TEST
| is registered, and you need to add a single document without completing

the registration. The schema document is `http://johndoe.com/test/test3.xsd` and it is stored locally in `/usr/jdoe/test/test3.xsd`. Issue this command to add the document:

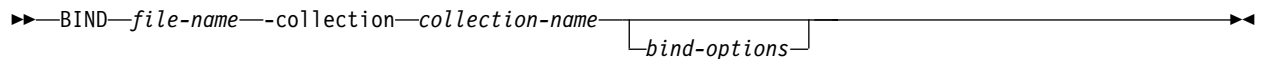
```
ADD XMLSCHEMA DOCUMENT TO JOHNDOE.TEST
ADD http://johndoe.com/test/test3.xsd
FROM file:///usr/jdoe/test/test3.xsd
```

Related concepts:

 XML schema management with the XML schema repository (XSR) (DB2 Programming for XML)

Command line processor BIND command

The DB2 command line processor BIND command binds DBRMs that are in hierarchical file system (HFS) files into a package.



```
▶ BIND file-name -collection collection-name bind-options ▶
```

file-name

The absolute or relative path name of the file that contains a DBRM.

collection-name

The name of the package collection into which you want to bind the DBRM.

bind-options

Specifies one or more of the bind options that are listed in Chapter 18, “BIND and REBIND options for packages and plans,” on page 101. Bind options must be separated by spaces.

Examples

Example: Binding a package with the command line processor

Suppose that you compiled a C language DB2 application whose full path name is `/u/usrt005/APP01.c`. The compilation process created a DBRM named `/u/usrt005/APP01.dbrm`. You want to bind the DBRM into package collection PKG01, with bind options ISOLATION(CS) and RELEASE(DEALLOCATE). Use this command to perform the bind:

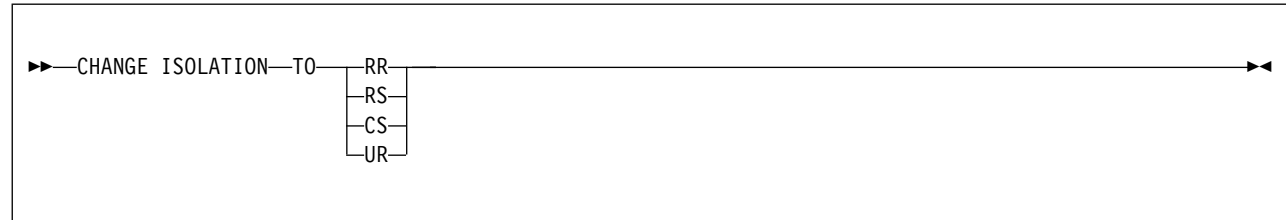
```
BIND /u/usrt005/APP01.dbrm -collection PKG01 ISOLATION(CS) RELEASE(DEALLOCATE)
```

Related reference:

BIND and REBIND options for packages and plans

Command line processor CHANGE ISOLATION command

The CHANGE ISOLATION command changes the isolation level for SQL statements that the command line processor executes.



See ISOLATION bind option for the meanings of the isolation levels.

During the command line processor session, the CHANGE ISOLATION command overrides the isolation value that is set in the IsolationLevel property in the command line processor properties file. The CHANGE ISOLATION command cannot be executed if a transaction is active.

Related tasks:

 [Choosing an ISOLATION option \(DB2 Performance\)](#)

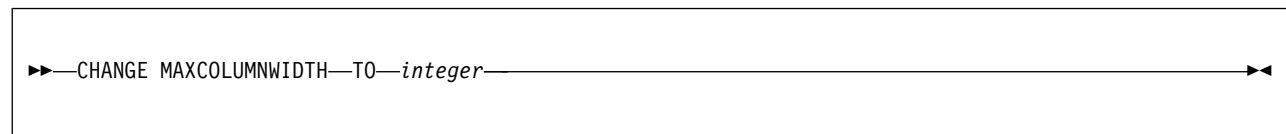
Related reference:

ISOLATION bind option

Command line processor properties file

Command line processor CHANGE MAXCOLUMNWIDTH command

The CHANGE MAXCOLUMNWIDTH command changes the number of bytes in a table column that the command line processor returns.



integer

The maximum number of bytes in any column that is displayed after execution of:

- An SQL SELECT statement
- An SQL CALL statement that returns a result set
- A command line processor LIST TABLES, DESCRIBE TABLE, or DESCRIBE CALL command

This value overrides the value of the MaxColumnWidth command line processor property during the command line processor session.

Related reference:

Command line processor properties file

Command line processor **CHANGE MAXLINESFROMSELECT** command

The **CHANGE MAXLINESFROMSELECT** command changes the number of rows that the command line processor returns.

```
▶▶—CHANGE MAXLINESFROMSELECT—TO—integer—◀◀
```

integer

The maximum number of rows that are displayed after execution of:

- An SQL **SELECT** statement
- An SQL **CALL** statement that returns a result set
- A command line processor **LIST TABLES**, **DESCRIBE TABLE**, or **DESCRIBE CALL** command

This value overrides the value of the `MaxLinesFromSelect` command line processor property during the command line processor session.

Related reference:

Command line processor properties file

Command line processor **COMMIT** command

The command line processor **COMMIT** command commits all SQL work since the previous commit or rollback operation, and releases any database locks that are currently held by the active connection.

The DB2 command line processor **COMMIT** command performs the same function as the SQL **COMMIT** statement.

```
▶▶—COMMIT—◀◀
```

Notes[®]

Implicit commit operation

By default, *autocommit mode* is enabled. This means that the command line processor implicitly issues a **COMMIT** statement after execution of every SQL statement. If an active connection exists, and you issue a new **CONNECT** command, the command line processor closes the currently active connection before making the new connection. When *autocommit mode* is enabled, the new connection can be established. However, if *autocommit mode* is disabled, you need to issue a **COMMIT** or **ROLLBACK** command before you issue the new **CONNECT** command. Otherwise, an error occurs.

Related reference:

 [COMMIT \(DB2 SQL\)](#)

Command line processor start command

Command line processor COMPLETE XMLSCHEMA command

The command line processor COMPLETE XMLSCHEMA command completes the registration of an XML schema.

```
▶▶—COMPLETE XMLSCHEMA—relational-identifier—┐  
└—WITH—schema-properties-URI—◀◀
```

relational-identifier

Specifies the relational name that was assigned to an XML schema in a previous REGISTER XMLSCHEMA command.

The schema of the relational identifier must be SYSXSR. If you specify a qualified name, the qualifier must be SYSXSR; for example, SYSXSR.MYSCHEMA. If you specify an unqualified name, you need to execute the SQL statement SET SCHEMA="SYSXSR" before you execute the command. A name that is enclosed in quotation marks, such as "SYSXSR.MYSCHEMA", is considered an unqualified name. Qualifiers or schema names that are enclosed in quotation marks are not folded to uppercase. Qualifiers or schema names that are not enclosed in quotation marks are folded to uppercase.

WITH *schema-properties-URI*


Specifies the URI of a properties document that is to be associated with this XML schema. The command line processor supports only file URIs, which begin with file://.

Example: Adding documents to an XML schema and completing schema registration

Suppose that a schema with relational name PRODSHEMA is already registered. The schema has additional properties that are stored locally in file /u/temp/CUSTPROP.xml. Issue this command to complete registration:

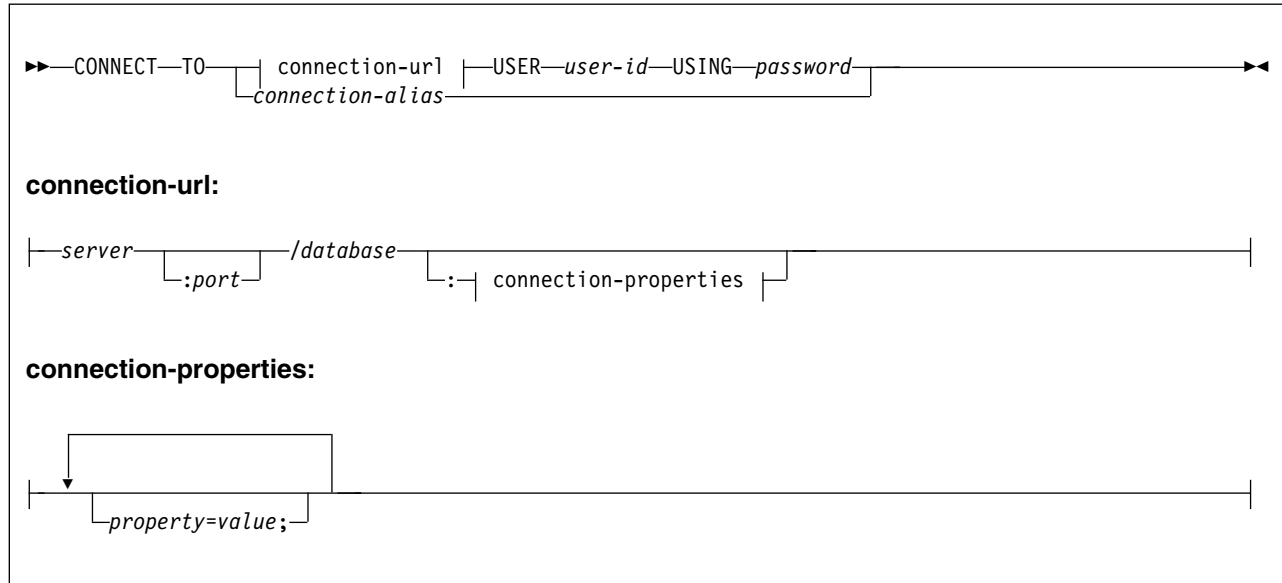
```
COMPLETE XMLSCHEMA PRODSHEMA WITH 'file:///u/temp/CUSTPROP.xml'
```

Related concepts:

 XML schema management with the XML schema repository (XSR) (DB2 Programming for XML)

Command line processor CONNECT command

The DB2 command line processor CONNECT command establishes a connection to a DB2 server.



TO connection-url

Specifies the DB2 database system to which the connection is made. It consists of the following parts:

server

The domain name or IP address of the DB2 database system.

port

The TCP/IP server port number that is assigned to the DB2 database system. This is an integer between 0 and 65535. The default is 446.

database

A name for the DB2 database system. *database* is the DB2 location name that is defined during installation. All characters in the DB2 location name must be uppercase characters.

You can determine the location name by executing the following SQL statement on the DB2 database system to which you want to connect:

```
SELECT CURRENT SERVER FROM SYSIBM.SYSDUMMY1;
```

connection-properties

A list of IBM Data Server Driver for JDBC and SQLJ properties and their values. You can specify one or more property and value pairs. Each property and value pair, including the last one, must end with a semicolon (;). Do not include spaces or other white space characters anywhere within the list of property and value strings.

Specify connection properties only if your system administrator tells you to do so.

USER user-id

The authorization name to use when connecting to the DB2 server.

This name is case insensitive. However, if you run the LIST TABLES command to see your own tables, the case of *user-id* must match the case of the schema name for your tables. For example, if you connect to the DB2 database server with a user ID of myid, LIST TABLES does not list tables with a schema name of MYID.

USING password

The password for the authorization name that is used when connecting to the DB2 server.

connection-alias

A name that corresponds to an entry in the properties file that is defined by the CLPPROPERTIESFILE environment variable. That entry specifies the connection URL, user ID, and password for the connection.

Example: Making a connection for the command line processor by specifying a connection URL, ID and password

Suppose that your system administrator told you that the DB2 server against which you want to use the command line processor has the connection URL syszos1.abc.com:5021/ABCLOC1. Your ID is myid01 and your password is mypw01. Use this command to make a connection to that server:

```
CONNECT TO syszos1.abc.com:5021/ABCLOC1 USER myid01 USING mypw01
```

Example: Making a connection for the command line processor by specifying a connection alias


Suppose that the CLPPROPERTIESFILE environment variable specifies a file named clp.properties. In that file, the following connection alias is defined:

```
MYALIAS01=syszos1.abc.com:5021/ABCLOC1,myid01,mypw01
```

Use this command to make a connection using the connection alias:

```
CONNECT TO MYALIAS01
```

Related reference:

 Properties for the IBM Data Server Driver for JDBC and SQLJ (DB2 Application Programming for Java)
Command line processor properties file

Command line processor DESCRIBE CALL command

The DESCRIBE CALL command returns information about the parameters of a stored procedure.

For each parameter, DESCRIBE CALL returns the following information:

- Parameter name
- Parameter type: IN, OUT, or INOUT
- Data type
- Length
- Scale
- Nullability: yes, no, or unknown

```

  ►► DESCRIBE CALL schema-name.procedure-name

```

schema-name

The schema name of the stored procedure for which parameter information is returned. The default is the user ID under which the command line processor session is running.

procedure-name

The name of the stored procedure for which parameter information is returned.

Example: Displaying parameter information for the SYSPROC.DSNUTILU stored procedure

Suppose that you want to display information about the DB2-supplied SYSPROC.DSNUTILU stored procedure. You do not want the output to wrap on your display, so you want the column widths in the output to be at most 12 bytes. Use these commands to set the column width and display the parameter information:

```

CHANGE MAXCOLUMNWIDTH TO 12
DISPLAY CALL SYSPROC.DSNUTILU

```

The following information is displayed:

COLUMN_NAME	COLUMN_TYPE	TYPE_NAME	PRECISION	LENGTH	SCALE	NULLABLE
UTILITY_ID	1	VARCHAR	16	16	<null>	0
RESTART	1	VARCHAR	8	8	<null>	0
UTSMT	1	VARCHAR	32704	32704	<null>	0
RETCODE	4	INTEGER	10	4	<null>	0

Mapping information for COLUMN_TYPE


- 0 - Unknown
- 1 - IN parameter
- 2 - INOUT parameter
- 3 - Result column in ResultSet
- 4 - OUT parameter
- 5 - Procedure return value

Mapping information for NULLABLE

- 0 - Does not allow NULL values
- 1 - Allow NULL values
- 2 - Nullability unknown

Related reference:

Command line processor CHANGE MAXCOLUMNWIDTH command

 [CALL \(DB2 SQL\)](#)

Command line processor DESCRIBE TABLE command

The DESCRIBE TABLE command returns information about the columns of a table.

For each column, DESCRIBE TABLE returns the following information:

- Column name
- Table qualifier
- Data type
- Length
- Scale
- Nullability: yes, no, or unknown

```
►► DESCRIBE TABLE table-creator.table-name ◄◄
```

table-creator

The user ID of the table creator. The default is the user ID under which the command line processor session is running.

table-name

The name of the table for which column information is returned.

Example: Displaying column information for the DSN8A10.DEPT table

Suppose that you want to display information about DB2 sample table DSN8A10.DEPT. You do not want the output to wrap on your display, so you want the column widths in the output to be at most 14 bytes. Use these commands to set the column width and display the parameter information:

```
CHANGE MAXCOLUMNWIDTH TO 14
DESCRIBE TABLE DSN8A10.EMP
```

The following information is displayed:

COLUMN_- NAME	TABLE_SCHEM	TYPE_NAME	COLUMN_- SIZE	DECIMAL_- DIGITS	IS_NULLABLE
DEPTNO	DSN8A10	CHAR	3	<null>	NO
DEPTNAME	DSN8A10	VARCHAR	36	<null>	NO
MGRNO	DSN8A10	CHAR	6	<null>	YES
ADMRDEPT	DSN8A10	CHAR	3	<null>	NO
LOCATION	DSN8A10	INTEGER	16	<null>	YES

Related reference:

Command line processor CHANGE MAXCOLUMNWIDTH command

 CREATE TABLE (DB2 SQL)

Command line processor DISCONNECT command

The command line processor DISCONNECT command closes the current connection to a DB2 server.

DISCONNECT succeeds only if no transactions are in progress.

```
▶▶—DISCONNECT—◀◀
```

Related reference:

Command line processor CONNECT command

Command line processor DISPLAY RESULT SETTINGS command

The DISPLAY RESULT SETTINGS command lists the current settings for the maximum column width and the maximum number of rows that are returned for queries that the command line processor executes.

The values that are displayed correspond to the values of command line processor properties MaxColumnWidth and MaxLinesFromSelect, or the overrides that you specify with the CHANGE MAXCOLUMNWIDTH and CHANGE MAXLINESFROMSELECT commands.

```
▶▶—DISPLAY RESULT SETTINGS—◀◀
```

Related reference:

Command line processor properties file

Command line processor CHANGE MAXLINESFROMSELECT command

Command line processor CHANGE MAXCOLUMNWIDTH command

Command line processor ECHO command

The ECHO command writes the contents of a text string to the output stream.

The ECHO command is useful for generating labels in an output file.

```
▶▶—ECHO—string—◀◀
```

Example: Use the ECHO command to identify a query in the output file

Suppose that you start the command line processor with the option -z output.txt. You connect to a DB2 server and execute these commands:

```
ECHO Find all tables that are not catalog tables:
ECHO SELECT NAME FROM SYSIBM.SYSTABLES WHERE NAME NOT LIKE 'SYS%'
SELECT NAME FROM SYSIBM.SYSTABLES WHERE NAME NOT LIKE 'SYS%'
```

output.txt contains output like this:

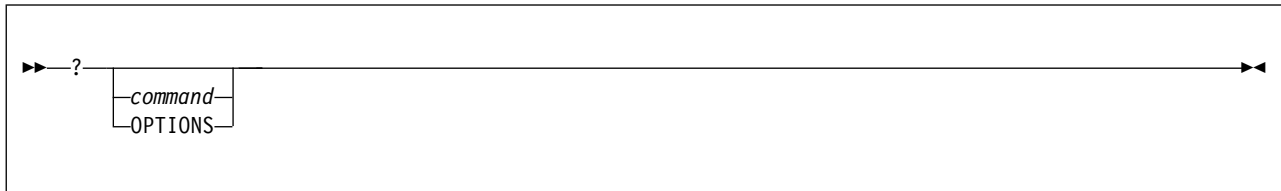
```
Find all tables that are not catalog tables:
SELECT NAME FROM SYSIBM.SYSTABLES WHERE NAME NOT LIKE 'SYS%'
NAME
...
ACT
AGEGROUP
AUX_BMP_PHOTO
AUX_EMP_RESUME
AUX_PSEG_PHOTO
...
```

Related reference:

Command line processor start command

Command line processor help command

The command line processor help command provides information about command line processor commands.



? Indicates that this is a request for help. If the question mark (?) is specified by itself, a list of command line processor commands is displayed.

command

Indicates that this is a request for help for a specific command line processor command. You can specify one or more keywords in the command name. If more than one command begins with the same keyword, the command line processor displays help for all commands that begin with that keyword.

OPTIONS

Indicates that this is a request for a list of command line processor start-up options and their default settings.

Examples

Example: Displaying a list of command line processor commands

To see a list of all command line processor commands, run this command:

```
?
```

Example: Displaying help for all LIST commands

To see help for all commands that begin with the keyword LIST, run this command:

```
? LIST
```

Example: Displaying help for a specific LIST command

To see help for the LIST TABLES command, run this command:

```
? LIST TABLES
```

Command line processor LIST COMMAND OPTIONS command

The LIST COMMAND OPTIONS command lists command line processor option flags, and indicates whether those option flags are set to ON or OFF.

```
▶—LIST COMMAND OPTIONS—▶
```

Related reference:

Command line processor start command

Command line processor LIST TABLES command

The LIST TABLES command lists tables, views, or aliases on a DB2 server.

```
▶—LIST TABLES—▶
```

```
└─FOR─┬─USER──┐
       │SYSTEM──┘
       │ALL──┐
       └─SCHEMA—schema-name──┘
```

FOR USER

Displays a list of tables, views, and aliases with a schema name that matches the user ID of the user that issued the CONNECT command to connect the command line processor to a DB2 server. The case of the user ID in the CONNECT command must match the case of the CREATOR value in SYSIBM.SYSTABLES for a table to be listed. FOR USER is the default.

FOR SYSTEM

Displays a list of catalog tables on the DB2 server.

FOR ALL

Displays a list of all tables, views, and aliases on the DB2 server.

FOR SCHEMA *schema-name*

Displays a list of all tables, views, and aliases with a schema name that matches *schema-name*.

Example: Listing tables, views, and aliases for the currently connected user

Suppose that you connected the command line processor to a DB2 server with user ID ADMF001. You want to display a list of tables, views, and aliases with a schema name of ADMF001. You do not want the output to wrap on your display, so you set the column widths on the display to at most 14 bytes. Use these commands to set the column width and display the list.

```
CHANGE MAXCOLUMNWIDTH TO 14
LIST TABLES
```

Information like this is displayed:

TABLE_SCHEM	TABLE_NAME	TABLE_TYPE
ADMF001	MYALIAS01	SYNONYM
ADMF001	MYTABLE01	TABLE
ADMF001	MYTABLE02	TABLE
ADMF001	MYVIEW01	VIEW

Example: Listing tables, views, and aliases for a specified schema name

Suppose that you want to display a list of tables, views, and aliases with a schema name of DSN8A10. You do not want the output to wrap on your display, and you do not want columns of up to 30 bytes to be truncated, so you set the column widths on the display to at most 30 bytes. Use these commands to set the column width and display the list.

```
CHANGE MAXCOLUMNWIDTH TO 30
LIST TABLES FOR SCHEMA DSN8A10
```

Information like this is displayed:

TABLE_SCHEM	TABLE_NAME	TABLE_TYPE
DSN8A10	AUX_BMP_PHOTO	AUXILIARY TABLE
DSN8A10	AUX_EMP_RESUME	AUXILIARY TABLE
DSN8A10	AUX_PSEG_PHOTO	AUXILIARY TABLE
DSN8A10	DSN_QUERY_AUX	AUXILIARY TABLE
DSN8A10	DSN_STATEMENT_CACHE_AUX	AUXILIARY TABLE
...		

Related reference:

Command line processor CHANGE MAXCOLUMNWIDTH command

Command line processor CONNECT command

 [SYSIBM.SYSTABLES table \(DB2 SQL\)](#)

Command line processor properties file

Use a command line processor properties file to set your own defaults for command line processor properties.

The name of the command line processor properties file is determined by the setting of the CLPPROPERTIESFILE environment variable. A sample command line processor properties file named clp.properties is in directory /usr/lpp/db2/db2a10/base/samples.

The command line properties file is a text file that contains properties in *keyword=value* format. When you invoke the command line processor, the command line processor loads the properties file. Any changes to the properties file while the command line processor session is active do not take effect during the session. However, you can temporarily change option values that have a value of OFF or ON during the session with the UPDATE COMMAND OPTIONS command.

The following table lists the keywords and values that you can put in the command line processor properties file.

The following table lists the options and their default values, if the defaults have not been set by a command line processor properties file. If you specify a blank value for a property, the command line processor uses the default value.

Table 1. Command line processor option flags

Keyword name	Option description	Possible values (first value is the default)
DisplaySQLCA	Display the contents of the SQLCA after SQL statement execution.	OFF ON blank
AutoCommit	Automatically commit SQL statements.	ON OFF blank
InputFilename	Read command input from the file specified by <i>file-name</i> .	<i>file-name</i> blank
DisplayOutput	Display data and messages on the standard output device.	ON OFF blank
StopOnError	Stop the command line processor and exit to the operating system if an error occurs during execution of a statement or command.	OFF ON blank
TerminationChar	The statement or command termination character: OFF or blank The command line processor handles each line as a complete statement unless the line ends with a space followed by a backslash(\). ON Use the semicolon (;) as the statement termination character. <i>single-character</i> Use that character as the termination character	OFF ON <i>single-character</i> blank
Echo	Echo the command text to the standard output device.	OFF ON blank
StripHeaders	Return data without any headers.	OFF ON blank
OutputFilename	Store output data and messages in the file specified by <i>file-name</i> .	OFF <i>file-name</i> blank

Table 1. Command line processor option flags (continued)

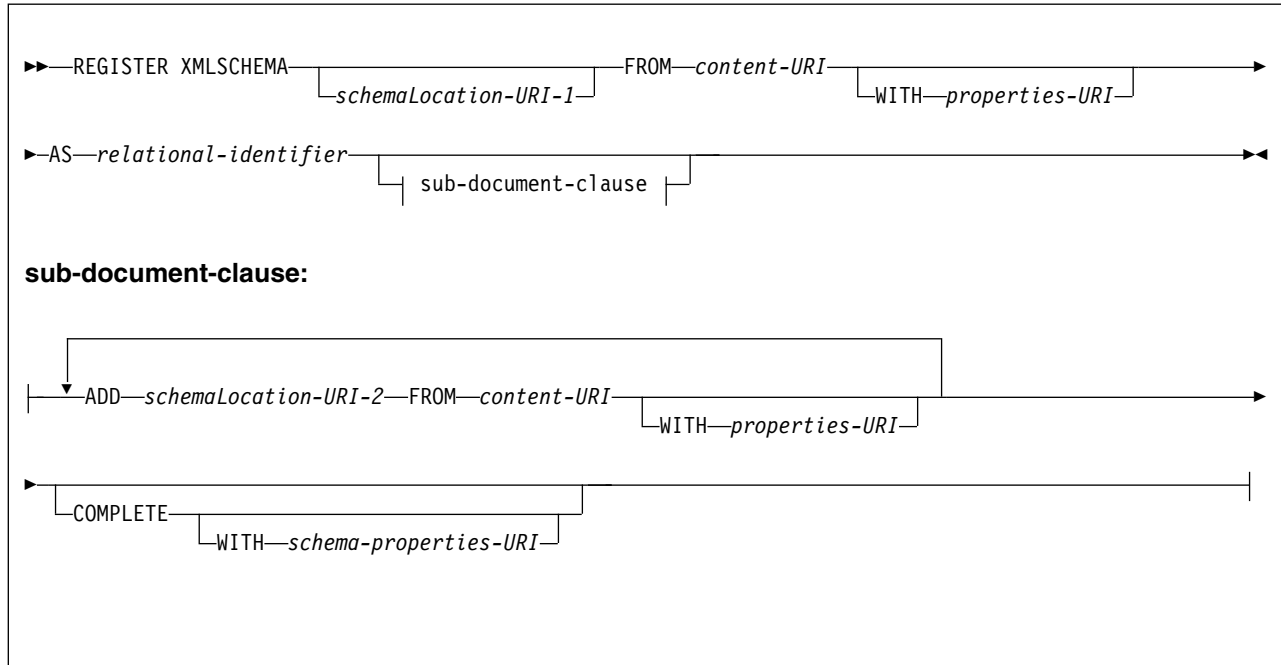
Keyword name	Option description	Possible values (first value is the default)
IsolationLevel	The SQL statement isolation level	CS RR RS UR CS RR RS UR blank
MaxLinesFromSelect	The maximum number of rows that are returned from an SQL SELECT statement	ALL <i>integer</i> blank
MaxColumnWidth	The maximum number of characters in any column that is displayed after execution of: <ul style="list-style-type: none"> • An SQL SELECT statement • An SQL CALL statement that returns a result set • A command line processor CATALOG command 	500 <i>integer</i> blank
<i>connection-alias</i>	A string that represents the information that is required in a CONNECT command. The string consists of a connection URL, user ID, and password, separated by commas. The connection URL, user ID and password are defined in "Command line processor CONNECT command" on page 32. Users can specify <i>connection-alias</i> in a CONNECT command to simplify the connection process.	None

Related reference:

Command line processor start command

Command line processor REGISTER XMLSCHEMA command

The command line processor REGISTER XMLSCHEMA command registers an XML schema with the DB2 database manager.



schemaLocation-URI-1

Specifies the schema location URI of the XML schema that is to be registered. The schema location is referenced from XML instance documents or the SQL XML schema validation function. If a single XML schema references multiple schema documents, each XML schema document other than the primary document must also be registered. Each document can be registered in either the sub-document clause in the REGISTER XMLSCHEMA command or in a separate ADD XMLSCHEMA DOCUMENT command.

FROM content-URI

Specifies the URI where the actual content of an XML schema document is located. The command line processor supports only file URIs, which begin with file://. The XML schema document must be in the Unicode encoding scheme.

WITH properties-URI

Specifies the URI of a properties document that are to be associated with a schema document. The command line processor supports only file URIs, which begin with file://.

AS relational-identifier

Specifies a name that can be used to refer to the XML schema in relational SQL or XML.

The schema of the relational identifier must be SYSXSR. If you specify a qualified name, the qualifier must be SYSXSR; for example, SYSXSR.MYSCHEMA. If you specify an unqualified name, you need to execute the SQL statement SET SCHEMA="SYSXSR" before you execute the command. A name that is enclosed in quotation marks, such as "SYSXSR.MYSCHEMA", is considered as an unqualified name. Qualifiers or

schema names that are enclosed in quotation marks are not folded to uppercase. Qualifiers or schema names that are not enclosed in quotation marks are folded to uppercase.

ADD *schemaLocation-URI-2*

Specifies the schema location URI of a related XML schema document, as it is referenced by this XML schema document.

COMPLETE

Indicates that there are no more XML schema documents to be associated with this XML schema. Validation of the schema is performed, and if there are no errors are found, the schema is to be marked as usable. Until XML schema registration is completed, the schema is not visible to or usable in other SQL or XML commands.

WITH *schema-properties-URI*

Specifies the URI of a properties document that is to be associated with this XML schema. The CLP supports only file URIs, which begin with file://. A schema properties document can be specified only when the XML schema is also being completed.

Example: Registering and completing an XML schema with a single XML schema document

Suppose that you want to register an XML schema named `http://mycompany.com/prod/p1_1.xsd`, and to make it available immediately. The content of the XML schema is a single document that has file URI `file:///u/temp/p1_1.xsd`. In SQL statements, you want to refer to the XML schema as `EXAMPLE.TEST1`. Use the following command to register and complete the XML schema.


```
REGISTER XMLSCHEMA http://mycompany.com/prod/p1_1.xsd
FROM file:///u/temp/p1_1.xsd AS EXAMPLE.TEST1 COMPLETE
```

Example: Registering and completing an XML schema with multiple XML schema documents

Suppose that you want to register an XML schema named `http://mycompany.com/prod/cust.xsd`, and to make it available immediately. The content of the XML schema consists of three documents. The contents of the three documents are locally stored as `file:///u/temp/cust1.xsd`, `file:///u/temp/cust2.xsd`, and `file:///u/temp/cust3.xsd`. The schema location URIs for the documents are `http://mycompany.com/prod/cust1.xsd`, `http://mycompany.com/prod/cust2.xsd`, and `http://mycompany.com/prod/cust3.xsd`. In SQL statements, you want to refer to the XML schema as `PRODSHEMA`. Additional properties for the XML schema are stored in a file with file URI `file:///u/temp/CUSTPROP.XML`. Use following to register and complete the XML schema.

```
REGISTER XMLSCHEMA http://mycompany.com/prod/cust.xsd
FROM file:///u/temp/cust.xsd'
AS PRODSHEMA WITH file:///u/temp/CUSTPROP.XML
ADD http://mycompany.com/prod/cust1.xsd
FROM file:///u/temp/cust1.xsd
ADD http://mycompany.com/prod/cust2.xsd
FROM file:///u/temp/cust2.xsd
ADD http://mycompany.com/prod/cust3.xsd
FROM file:///u/temp/cust3.xsd
COMPLETE
```

Related concepts:

 XML schema management with the XML schema repository (XSR) (DB2 Programming for XML)

Command line processor REMOVE XMLSCHEMA command

The command line processor REMOVE XMLSCHEMA command deletes a previously registered XML schema from the XML schema repository.

```
►► REMOVE XMLSCHEMA relational-identifier ◄◄
```

relational-identifier

Specifies a name that is used to refer to the XML schema in relational SQL or XML. This value is the relational identifier that you used when you registered the XML schema.


The relational identifier has the qualifier SYSXSR. If you specify REMOVE XMLSCHEMA with an unqualified relational identifier, you must have previously executed a SET SCHEMA SQL statement to set the schema name to SYSXSR.

Example: Removing an XML schema from the XML schema repository

Suppose that you want to remove an XML schema with relational identifier SYSXSR.PRODSHEMA from the XML schema repository. Run this command:

```
REMOVE XMLSCHEMA SYSXSR.PRODSHEMA
```

Related concepts:

 XML schema management with the XML schema repository (XSR) (DB2 Programming for XML)

Command line processor ROLLBACK command

The command line processor ROLLBACK command rolls back all SQL work since the previous commit or rollback operation, and releases any database locks that are currently held by the active connection.

The DB2 command line processor COMMIT command performs the same function as the SQL ROLLBACK statement.

```
►► ROLLBACK ◄◄
```

Notes

ROLLBACK and autocommit mode

If autocommit mode is enabled, the ROLLBACK command has no effect.

Related reference:

 COMMIT (DB2 SQL)

Command line processor start command

Command line processor SQL statements

Execute SQL statements in the command line processor after you issue the CONNECT command.

The SQL statements that can be executed in the command line processor have the same syntax as SQL statements that are executed in application programs.



Notes

Specifying output parameters in a CALL statement

To specify an output parameter in a CALL statement, use a question mark (?).

Examples

Example: Executing a query with the command line processor

Suppose that the sample tables are installed on a DB2 server with connection URL syszos1.abc.com:5021/ABCLOC1. Use the following commands to connect to that server, and issue a SELECT statement to display all the rows in the sample employee table.

```
CONNECT TO syszos1.abc.com:5021/ABCLOC1 USER myid01 USING mypw01
SELECT * FROM DSN8A10.EMP
```

Example: Calling a stored procedure with the command line processor

Suppose that you want to display information about all of the subsystem parameters, application programming defaults, and IRLM parameters for a DB2 server. The DB2-supplied stored procedure SYSPROC.ADMIN_INFO_SYSPARM provides that information. Use the following commands to connect to the server, and call the stored procedure.

```
CONNECT TO syszos1.abc.com:5021/ABCLOC1 USER myid01 USING mypw01
CALL SYSPROC.ADMIN_INFO_SYSPARM(NULL,?,?)
```

Related concepts:

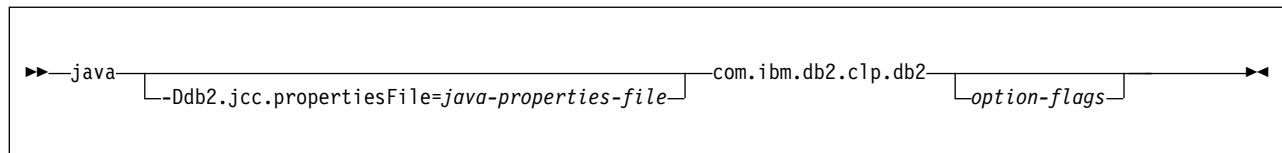
[Queries \(DB2 SQL\)](#)

Related reference:

[Statements \(DB2 SQL\)](#)

Command line processor start command

The command `java com.ibm.db2.clp.db2` starts the DB2 command line processor.



java com.ibm.db2.clp.db2

Starts the command line processor.

To simplify starting the command line processor, you can define an alias for this command, and then start the command line process by specifying the alias. For example:

```
alias db2="java com.ibm.db2.clp.db2"
db2
```

-Ddb2.jcc.propertiesFile=java-properties-file

The command line processor runs under the IBM Data Server Driver for JDBC and SQLJ. `-Ddb2.jcc.propertiesFile=java-properties-file` specifies the name of a file that contains configuration properties for the IBM Data Server Driver for JDBC and SQLJ. Specify this option only if your system administrator tells you to do so.

option-flags

Specifies one or more options for the command line processor. When you specify more than one option, separate the option flags with spaces. To turn an option on, precede the option flag with a minus (-) sign. To turn the option off, precede the option flag with a plus (+) sign.

The following table lists the options and their default values, if the defaults have not been set by a command line processor properties file.

Table 2. Command line processor option flags

Option flag	Option description	Default setting
a	Display the contents of the SQLCA after SQL statement execution.	OFF
c	Automatically commit SQL statements.	ON
f <i>file-name</i>	Read command input from the file specified by <i>file-name</i> .	OFF
o	Display data and messages on the standard output device.	ON
s	Stop the command line processor and exit to the operating system if an error occurs during execution of a statement or command.	OFF

Table 2. Command line processor option flags (continued)

Option flag	Option description	Default setting
t	Use the semicolon (;) as the statement termination character. If this option is not set, the command line processor handles each line as a complete statement unless the line ends with a space followed by a backslash(\).	OFF
tdx	Use x as the statement termination character.	OFF
v	Echo the command text to the standard output device.	OFF
x	Return data without any headers.	OFF
z <i>file-name</i>	Store output data and messages in the file specified by <i>file-name</i> .	OFF

The values of any options that you do not specify are determined by the values in the properties file that is specified by the CLPPROPERTIESFILE environment variable.

Examples

Example: Starting the DB2 command line processor with defaults

Suppose that no alias has been set for the DB2 command line processor name. Use this command to start the command line processor with the default values:

```
java com.ibm.db2.clp.db2
```

Example: Starting the DB2 command line processor with options

Suppose that the alias db2 has been set for the DB2 command line processor name. You want to start the command line processor with these options, which are not the default options:

- After each SQL statement is executed, the contents of the SQLCA are displayed.
- COMMIT is not executed automatically after each SQL statement.
- The statement termination character is that pound sign (#).

Use this command:

```
db2 -a +c -td#
```

Example: Starting the DB2 command line processor in batch mode

Suppose that the alias db2 has been set for the DB2 command line processor name. You want to run the command line processor with these options:

- The DB2 command line processor runs in batch mode. File db2commands.clp contains the commands.
- The commands are echoed to the display.

Use this command:

```
db2 -f db2commands.clp -v
```

Related concepts:

[Customization of IBM Data Server Driver for JDBC and SQLJ configuration properties \(DB2 Application Programming for Java\)](#)

Related reference:

[IBM Data Server Driver for JDBC and SQLJ configuration properties \(\)](#)

Command line processor TERMINATE command

The TERMINATE command stops the DB2 command line processor.

```
▶▶—TERMINATE—▶▶
```

Related reference:

Command line processor start command

Command line processor z/OS UNIX System Services command

A z/OS UNIX System Services command can be run in the command line processor can be run by preceding the command with an exclamation mark (!).

```
▶▶—!uss-command—▶▶
```

! Indicates that the text that follows is a this is a z/OS UNIX System Services command.

uss-command

Any z/OS UNIX System Services command.

Examples

Example: Listing files in the current directory

Suppose that you are in a command line processor session, and you need to run the z/OS ls command to obtain a list of files in the current directory. Run this command:

```
!ls
```

Command line processor UPDATE COMMAND OPTIONS command

The UPDATE COMMAND OPTIONS command changes property values for properties that have a value of ON or OFF.

```
►► UPDATE COMMAND OPTIONS—USING—  
| option-flag—ON—  
| option-flag—OFF—  
| property-keyword—ON—  
| property-keyword—OFF—  
◄◄
```

option-flag

One of the following option flags:

- a
- c
- o
- s
- v
- x
- z (OFF only)

See Command line processor start command for the meaning of each option flag.

property-keyword

One of the following option flags:

- DisplaySQLCA
- AutoCommit
- DisplayOutput
- StopOnError
- Echo
- StripHeaders

See Command line processor properties file for the meaning of each property keyword.

Related reference:

Command line processor start command

Command line processor properties file

Chapter 8. DB2 command text is recorded in the DB2 recovery log


All DB2 commands issued after DB2 restart and before DB2 shutdown are logged by DB2.

These commands are written in an IFCID 0090 trace record with a destination header that is mapped by macro DSNDQWIW. The log record type is 0010 (system event), and the subtype is 0041 (trace record).

Part 2. DB2 and related commands

The references topics for commands contain syntax diagrams, semantic descriptions, rules, and usage examples of commands. They are organized alphabetically by command name.

Related tasks:

 [Controlling DB2 operations by using commands \(DB2 Administration Guide\)](#)

Related information:

[About DB2 and related commands](#)

Chapter 9. ABEND (DSN)

The DSN subcommand ABEND causes the DSN session to terminate with abend completion code X'04E' and reason code of X'00C50101'. It is intended to be used at the direction of IBM Software Support, if a problem is suspected (such as incorrect output) with another DSN subcommand, and the problem does not cause an abend and a dump to be generated.

Important: The ABEND subcommand is used for diagnostic purposes only, and is intended to be used only under the direction of IBM Software Support. Use it only when diagnosing a problem with DSN or DB2.

Environment

You can use ABEND from DB2I, or from a DSN session under TSO that runs in either the foreground or background.

Data sharing scope: Member.

Authorization

None is required.

Syntax

▶▶ ABEND ◀◀


Usage notes

Sometimes, the information dumped is not the information that you want. Always use the ABEND subcommand as soon as possible after a problem is re-created to increase the chances of obtaining meaningful data. Do not press the ATTENTION key before issuing the ABEND subcommand; usually, the data is lost.

Related reference:

DSN (TSO)

Related information:

 [00C50101 \(DB2 Codes\)](#)

Chapter 10. -ACCESS DATABASE (DB2)

The DB2 command ACCESS DATABASE forces a physical open of a table space, index space, or partition, or removes the GBP-dependent status for a table space, index space, or partition. The MODE keyword specifies the action taken.

Abbreviation: -ACC

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS® terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- STARTDB privilege
- DBMAINT authority
- DBCTRL authority
- DBADM authority
- SYSCtrl authority
- SYSADM authority
- System DBADM authority

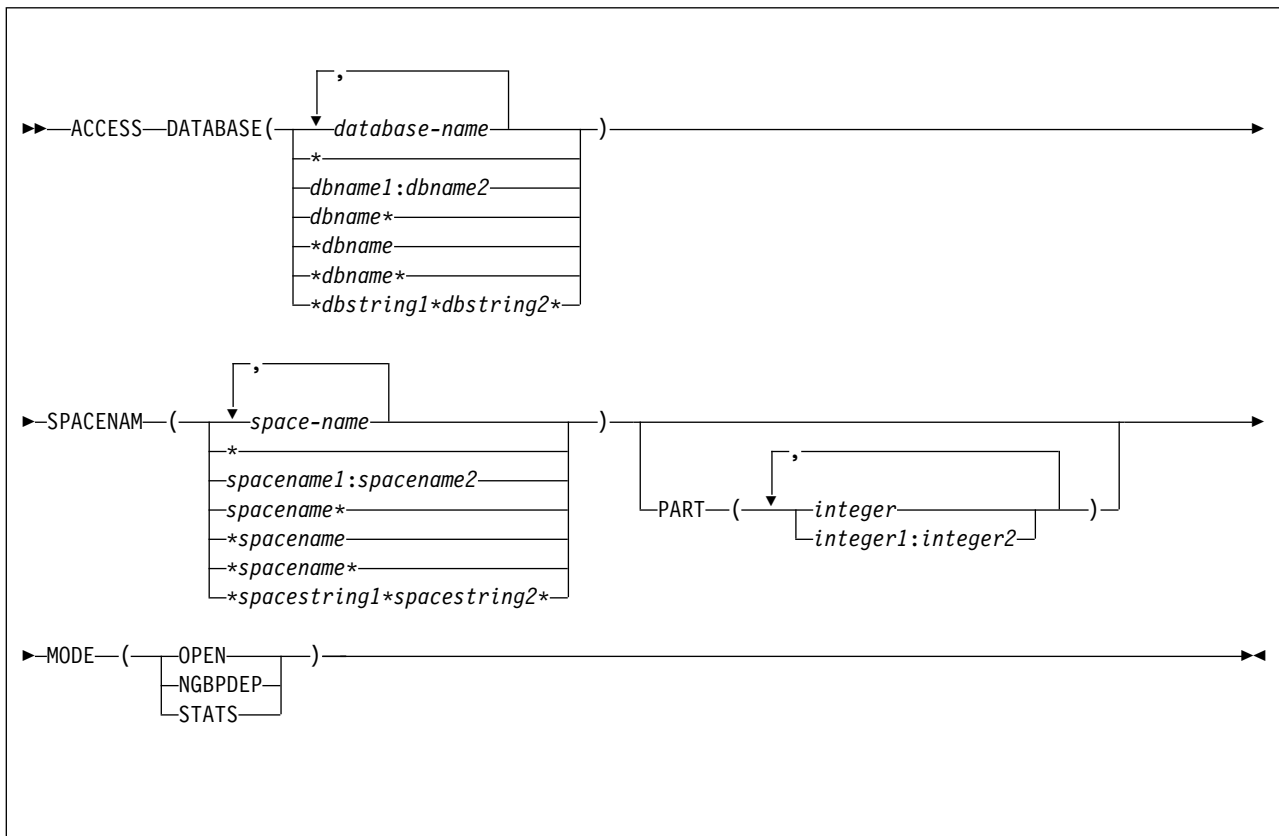
When you are using a privilege set that does not contain the STARTDB privilege for a specified database, DB2 issues an error message and the ACCESS command fails.

All specified databases with the STARTDB privilege included in the privilege set of the process are started.

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

When data definition control is active, installation SYSOPR or installation SYSADM authority is required to start a database, a table space, or an index space containing a registration table or index.

Syntax



Option descriptions

DATABASE (*database-name*, ...)

Specifies the names of the database, or database for the table spaces or index spaces to access.

Abbreviation: DB

database-name

The name of one or more database to access. To specify multiple database names, separate the values in the list by commas.

SPACENAM(*space-name*, ...)

Indicates names of table spaces or indexes within the specified database to access.

Abbreviation: SPACE, SP

space-name

The name of one or more table spaces or index spaces to access.

You can write *space-name* like *database-name* to designate the name of a single table space or index space.

PART(*integer*,)

Indicates the partition number of one or more partitions, within the specified table space or index, that are to be accessed.

The specified *integer* value must identify a valid partition number for the corresponding space name and database name. If you specify nonvalid

partition numbers, you receive an error message for each non-valid number, but all valid partitions that you specified are accessed.

integer can be written to designate one of the following specifications:

- A list of one or more partitions.
- A range of all partition numbers that collate greater than or equal to *integer-1* and less than or equal to *integer-2*.
- A combination of lists and ranges.

PART is valid with partitioned table spaces, partitioned indexes, and nonpartitioned type 2 indexes of partitioned table spaces. If you specify PART with a nonpartitioned table space or index on a nonpartitioned table space, you receive an error message, and the nonpartitioned space is not accessed. When a logical partition is accessed, the index is not closed. A nonpartitioning index must be accessed without the use of PART to close the index.

MODE(*mode-value*)

Specifies the action for the command, where *mode-value* is one of the following values:

OPEN

Forces the physical opening of the page set or partition on just the local member. This moves the overhead of the physical open from an SQL thread to the command thread. This improves the transaction rate for the first SQL thread to reference a given page set or partition.

When MODE(OPEN) is specified, DB2 does not process objects with the following characteristics:

- The objects are defined with DEFINE NO.
- The physical data sets for the objects have not been created.

NGBPDEP

Removes group buffer pool dependency from the specified page set or partition, without disrupting ongoing activity. Use this before running large batch processes against a particular page set or partition to improve performance in a data sharing environment. Issue this command only on the same member that runs the batch processes.

STATS

Externalizes the in-memory real-time statistics to the appropriate catalog tables. In data sharing environments, the in-memory statistics are externalized for all members. This mode does not physically open the page sets or change the states of the page sets.

When the MODE (STATS) option is specified, only certain combinations of *database-name* and *space-name* values are recommended.

Related information:

Updating real-time statistics immediately (DB2 Performance)

Usage notes

The following description contains additional information about how to use the ACCESS DATABASE command.

What to do if ACCESS DATABASE returns error message DSNI045I

Issuing ACCESS DATABASE MODE(OPEN) on several members of a data sharing group at the same time might result in error message DSNI045I

and reason code 00C90090, due to resource contention. If this error occurs, reissue the ACCESS DATABASE MODE(OPEN) command to open all of the objects that the command affects. To avoid the error, do not issue ACCESS DATABASE MODE(OPEN) on multiple data sharing group members at the same time.

Examples

Example: Physically opening partitions

This command physically opens partitions 1 and 3 of table space DSN9002 of database DSN9001.

```
-ACCESS DATABASE(DSN9001) SPACENAM(DSN9002) PART(1,3) MODE(OPEN)
```

Example: Physically closing a nonpartitioned table space

This command physically closes the entire nonpartitioned table space DSN9003 of database DSN9001 and makes it non-group bufferpool dependent.

```
-ACCESS DATABASE(DSN9001) SPACENAM(DSN9003) MODE(NGBPDEP)
```

Output similar to the following output indicates that the command completed successfully:


```
-DSNTDDIS 'ACCESS DATABASE' NORMAL COMPLETION
```

Example: Externalize all in memory statistics to the real-time statistics tables

The following command externalizes all in-memory statistics that are currently held in the system to the real-time statistics table.


```
-ACCESS DB(*) SP(*) MODE(STATS)
```

Related concepts:

 Physical open of a page set of partition (DB2 Data Sharing Planning and Administration)

 Inter-DB2 interest and GBP-dependency (DB2 Data Sharing Planning and Administration)

Related tasks:

 Improving batch processing performance in data sharing (DB2 Data Sharing Planning and Administration)

Related information:

Chapter 11. -ALTER BUFFERPOOL (DB2)

The DB2 command ALTER BUFFERPOOL alters attributes for active or inactive buffer pools. Altered values are used until altered again.

Abbreviation: -ALT BPOOL

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS, or CICS terminal, or a program using the instrumentation facility interface (IFI).

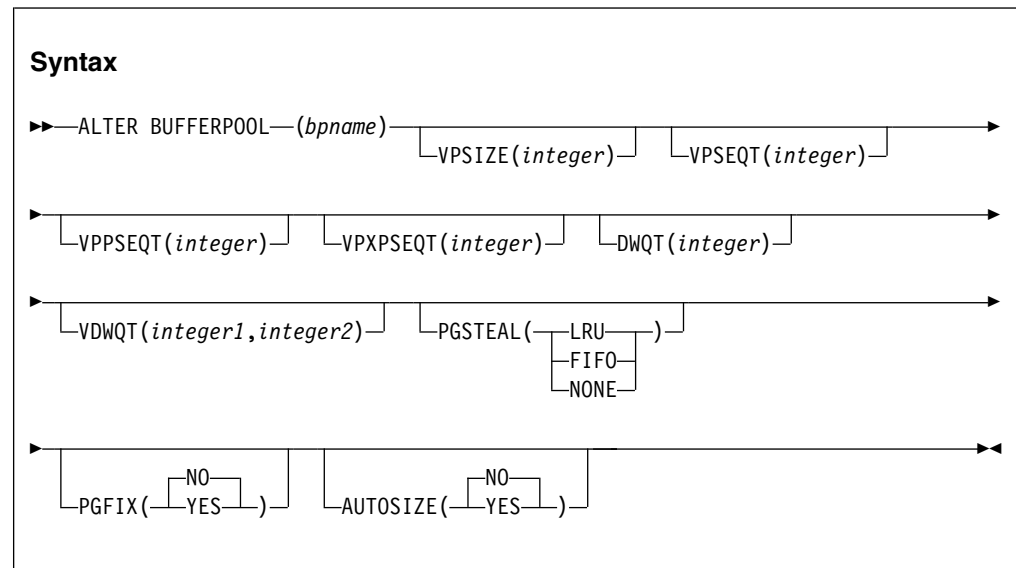
Data sharing scope: Member

Authorization

To issue this command, you must use a set of privileges for the process that includes one of the following authorities:

- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization that uses primary and secondary authorization IDs.



Option descriptions

(*bpname*)

Specifies the buffer pool to alter.

- 4 KB page buffer pools are named BP0 through BP49

- 8 KB page buffer pools are named BP8K0 through BP8K9
- 16 KB page buffer pools are named BP16K0 through BP16K9
- 32 KB page buffer pools are named BP32K through BP32K9

VPSIZE (*integer*)

Changes the buffer pool size.

The value of *integer* specifies the number of buffers to allocate to the active buffer pool.

The value of *integer* can range 0 - 250000000 for 4 KB page buffer pools other than BP0. For BP0, the minimum value is 2000. For 8 KB page buffer pools, the minimum value is 1000. For 16 KB page buffer pools, the minimum value is 500. For 32 KB page buffer pools, the minimum value is 250.

DB2 limits the total VPSIZE for all buffer pools to 1 TB. In addition, DB2 limits the amount of buffer pool storage to twice the available real storage for the z/OS image.

If you specify VPSIZE as 0 for an active buffer pool, DB2 quiesces all current database access and update activities for that buffer pool and then deletes the buffer pool. Subsequent attempts to use table spaces or indexes that are assigned to that buffer pool fail.

VPSEQT (*integer*)

Changes the sequential steal threshold for the buffer pool.

The value of *integer* specifies the sequential steal threshold for the buffer pool. This value is expressed as a percentage of the total buffer pool size, and valid values range 0 - 100. This threshold affects the allocation of buffers in the buffer pool to page read requests that are part of a sequential access pattern. This includes pages that are prefetched. If the number of buffers that contain sequentially accessed pages exceeds the threshold, a sequential request attempts to reuse one of those buffers rather than a buffer that contains a non-sequentially accessed page. The initial default value is 80.

When VPSEQT=0, sequentially accessed pages are not kept in the buffer pool after the accessing agent releases them. Also, prefetch is disabled.

When VPSEQT=100, DB2 does not prefer reusing sequential buffers over using non-sequential buffers.

VPPSEQT (*integer*)

Changes the parallel sequential threshold for the buffer pool. This threshold determines how much of the buffer pool is used for parallel processing operations.

The value of *integer* specifies the parallel sequential threshold for the buffer pool. This value is expressed as a percentage of the sequential steal threshold, and valid values range 0 - 100. The initial default value is 50.

When VPPSEQT=0, parallel processing operations and prefetch operations that are triggered by index I/O parallelism are disabled.

VPXPSEQT (*integer*)

Changes the assisting parallel sequential threshold for the buffer pool. This threshold determines the portion of the buffer pool that is used for processing queries that originate on other members of the data sharing group. This option is valid and effective only when DB2 is in data sharing mode; it is ignored when DB2 is not in data sharing mode.

The value of *integer* specifies the assisting parallel sequential threshold for the buffer pool. The value of *integer* is expressed as a percentage of the parallel sequential threshold (VPPSEQT). Altering the sequential steal threshold or the parallel sequential threshold directly affects the portion of buffer resources that are dedicated to assistant parallel operations. The valid values range 0 - 100. The initial default value is 0.

When VPXPSEQT=0, this buffer pool cannot be used to assist another DB2 with parallel processing.

DWQT (*integer*)

Changes the buffer pool's deferred write threshold.

The value of *integer* specifies the deferred write threshold for the buffer pool. This value is expressed as a percentage of the total buffer pool size, and valid values range 0 - 90. This threshold determines when deferred writes begin, based on the number of unavailable buffers. When the count of unavailable buffers exceeds the threshold, deferred writes begin. The initial default value is 30 percent.

VDWQT (*integer-1*,*integer-2*)

Changes the buffer pool's vertical deferred write threshold.

The value of *integer1* specifies the vertical deferred write threshold for the buffer pool. *integer1* is expressed as a percentage of the total buffer pool size, and valid values range 0 - 90.

This threshold determines when deferred writes begin, based on the number of updated pages for a particular data set. Deferred writes begin for that data set when the count of updated buffers for a data set exceeds the threshold. This threshold can be overridden for page sets accessed by DB2 utilities. It must be less than or equal to the value specified for the DWQT option.

The default value is 5 percent. A value of 0 indicates that the deferred write of 32 pages begins when the updated buffer count for the data set reaches 40.

The value of *integer-2* specifies the vertical deferred write threshold for the buffer pool. *integer-2* is expressed as an absolute number of buffers. You can use *integer2* when you want a relatively low threshold value for a large buffer pool, but *integer-1* cannot provide a fine enough granularity between *integer-1* values of 0 and 1. The value of *integer-2* applies only when the value of *integer-1* is 0. DB2 ignores a value that is specified for *integer-2* if the value specified for *integer-1* is non-zero. The value of *integer-2* can range 0 - 9999. The default value is 0.

If the value of *integer-1* is 0 and *integer-2* is a non-zero value, DB2 uses the value that is specified for *integer-2* to determine the threshold. If both values are 0, the *integer-1* value of 0 is used as the threshold.

PGSTEAL

Specifies the page-stealing algorithm that DB2 uses for the buffer pool.

The initial default is PGSTEAL(LRU).

(LRU)

Specifies that the buffer pool buffers are managed according to the rules of a least recently used (LRU) algorithm.

(FIFO)

Specifies that the buffer pool buffers are managed according to the rules of a first-in-first-out (FIFO) algorithm. This option reduces the cost of maintaining the information about which buffers are least-recently used.

However, the virtual storage might not be freed immediately. Determine the status of the buffer pool by issuing the DISPLAY BUFFERPOOL command.

Important: To avoid a major performance impact when you contract an active buffer pool, follow these guidelines:

- Do not lower the size of an active buffer pool by a large amount when there is a significant amount of buffer pool activity in the subsystem. Issue DISPLAY BUFFERPOOL to determine the amount of activity before you attempt to contract the buffer pool.
- If you need to do a very large buffer pool contraction, issue ALTER BUFFERPOOL several times to do multiple smaller contractions, instead of one large contraction.

Deleting an active buffer pool

If you use ALTER BUFFERPOOL to delete an active buffer pool (by specifying 0 for VPSIZE), DB2 issues a message to indicate that it is ready to explicitly delete this buffer pool. When DB2 accepts the delete buffer pool request, the buffer pool is marked as "delete pending". All current access to the buffer pool is quiesced, later access attempts fail with an error message, and all open page sets that refer to the buffer pool are closed.

Altering attributes that are stored in the BSDS

The buffer pool attributes that are stored in the BSDS cannot be changed offline.

Setting a buffer pool to be fixed in real storage

If you use the ALTER BUFFERPOOL command with the PGFIX option set to YES to fix a buffer pool in real storage, the change is pending and the buffer pool becomes fixed only at the next allocation.

In order to fix the buffer pool in real storage, issue the command ALTER BUFFERPOOL(*bpname*) PGFIX(YES). If the buffer pool that you specify for *bpname* is not currently allocated, the buffer pool becomes fixed in real storage when it is allocated. If the buffer pool that you specify for *bpname* is currently allocated, do one of the following procedures to fix the buffer pool in real storage:

- If the buffer pool that you specify for *bpname* is not one of the buffer pools that is used for the DB2 catalog and directory (BP0, BP8K0, BP16K0, or BP32K):
 1. Issue the ALTER BUFFERPOOL command with the VPSIZE option set to 0 to deallocate the buffer pool:

```
-ALTER BUFFERPOOL(bpname) VPSIZE(0)
```
 2. Issue the ALTER BUFFERPOOL command with the VPSIZE and PGFIX options to change the buffer pool size and to use long-term page-fixing at the next allocation:

```
-ALTER BUFFERPOOL(bpname) VPSIZE(vpsize) PGFIX(YES)
```
- If the buffer pool that you specify for *bpname* is one of the buffer pools that is used for the DB2 catalog and directory (BP0, BP8K0, BP16K0, or BP32K):
 1. Issue the ALTER BUFFERPOOL command with the PGFIX option to change the buffer pool to use long-term page fixing (the change is pending until the next allocation of the buffer pool):

```
-ALTER BUFFERPOOL(bpname) PGFIX(YES)
```

2. Issue the STOP DATABASE command or the STOP DB2 command to deallocate the buffer pool
3. Issue the START DATABASE command or the START DB2 command to reallocate the buffer pool (depending on which command you used to deallocate the buffer pool)

Relating VPPSEQT and VPXPSEQT

The following table explains how the two parallel sequential thresholds, VPPSEQT for parallel sequential and VPXPSEQT for assisting parallel sequential threshold, are related. VPXPSEQT is a percentage of VPPSEQT, which is itself a portion of VPSEQT. Multiply VPXPSEQT by VPPSEQT to obtain the total amount of the buffer pool that can be used to assist another DB2 subsystem with parallel processing. In addition, VPPSEQT is affected by changing VPSIZE and VPSEQT; therefore, VPXPSEQT is also affected by VPSIZE and VPSEQT.

Table 3. Relationship between VPPSEQT and VPXPSEQT

If VPPSEQT is set to:	and VPXPSEQT is set to:	The percentage of the buffer pool available to assist Sysplex query parallelism equals:
50	50	25
50	100	50
100	50	50
Any value.	0	0
0	Any value.	0

Examples

Example: Setting the buffer pool size

This command sets the size of buffer pool BP0 to 2000.

```
-ALTER BUFFERPOOL(BP0) VPSIZE(2000)
```

Example: Setting the sequential steal threshold of a buffer pool

This command sets the sequential steal threshold of buffer pool BP0 to 75% of the buffer pool size.

```
-ALTER BUFFERPOOL(BP0) VPSEQT(75)
```


Example: Deleting a buffer pool


This command deletes buffer pool BP1.

```
-ALTER BUFFERPOOL(BP1) VPSIZE(0)
```


Use this option carefully because specifying a 0 size for an active buffer pool causes DB2 to quiesce all current database access. All subsequent requests to open page sets fail.


Related concepts:

 Buffer pool threshold for parallelism assistants (DB2 Data Sharing Planning and Administration)

 Buffer pool thresholds (DB2 Performance)

Related tasks:

 Choosing a page-stealing algorithm (DB2 Performance)

 Fixing a buffer pool in real storage (DB2 Performance)

Chapter 12. -ALTER GROUPBUFFERPOOL (DB2)

The DB2 command ALTER GROUPBUFFERPOOL alters attributes of group buffer pools.

Abbreviation

-ALT GBPOOL

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group

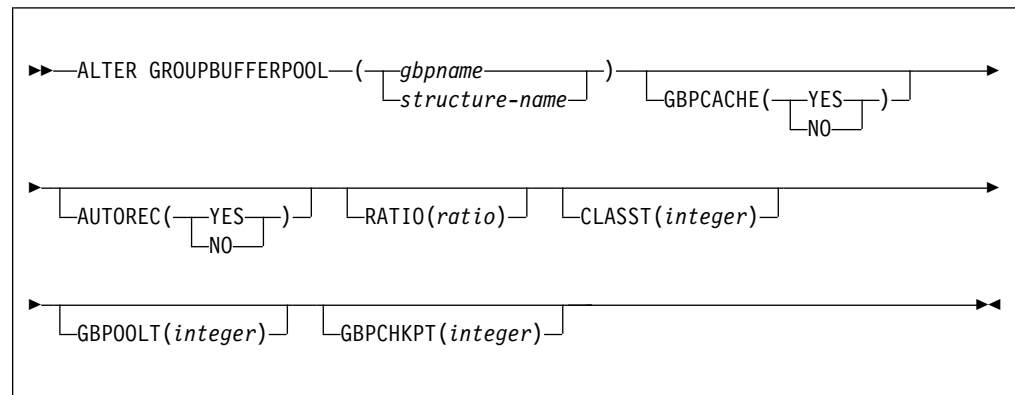
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SYSOPR authority
- SYSTRM authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

(*gbpname*)

Specifies the DB2 group buffer pool to alter.

- 4-KB group buffer pools are named GBP0 through GBP49
- 8-KB group buffer pools are named GBP8K0 through GBP8K9
- 16-KB group buffer pools are named GBP16K0 through GBP16K9

- 32-KB group buffer pools are named GBP32K through GBP32K9

(structure-name)

Specifies the coupling facility structure for the group buffer pool. The coupling facility structure name has the format, *groupname_gbpname*.

groupname is the DB2 data sharing group name and the underscore (_) separates *groupname* and *gbpname*.

GBPCACHE

Specifies whether *gbpname* is to be used for both caching data and cross-invalidation, or just for cross-invalidation.

(YES)

Indicates that *gbpname* is used for caching data and cross-invalidation.

Any no-data-caching attribute that is specified at either the page set or group buffer pool level takes precedence over a caching specification. The following table illustrates the precedence of a no-data-caching specification.

Table 4. Precedence of a no-data-caching specification

Group buffer pool specification	Page set specification	Attribute that takes precedence
GBPCACHE(NO)	GBPCACHE CHANGED GBPCACHE ALL	GBPCACHE(NO)
GBPCACHE(YES)	GBPCACHE NONE	GBPCACHE NONE

(NO)

Indicates that *gbpname* is used only for cross-invalidation. This group buffer pool contains no data entries. The GBPCACHE option of table spaces or index spaces that use this group buffer pool is ignored.

AUTOREC

Specifies whether automatic recovery by DB2 takes place when a structure failure occurs or when the connectivity of all members of the group to the group buffer pool is lost.

(YES)

Enables DB2 to automatically recover page sets and partitions that have a status of group buffer pool RECOVER pending (GRECP) and that have pages on the logical page list.

(NO)

Disables automatic recovery. Issue a START DATABASE command to recover page sets and partitions that have a status of GRECP or that have pages on the logical page list.

RATIO (ratio)

Changes the ratio of the number of directory entries to the number of data pages in the group buffer pool; that is, how many directory entries exist for each data page.

ratio can be a decimal number from 1.0 to 255, inclusive. Any digits after the first decimal place are ignored; for example, 5.67 is treated as 5.6. If *ratio* is greater than 25, any digits after the decimal point are ignored; for example, 25.98 is treated as 25. The default ratio is 5.

The actual number of directory entries and data pages that are allocated depends on the size of the coupling facility structure, which is specified in the coupling facility policy definitions (CFRM policy).

CLASST (*integer*)

Changes the threshold at which data in the class castout queue is cast out to disk.

integer is a percentage of the number of data entries and can be an integer between 0 and 90, inclusive. The default is 5.

For example, CLASST(10) starts class castout when the number of pages in that class equals 10% of the group buffer pool page capacity.

GBPOOLT (*integer*)

Changes the threshold at which data in the group buffer pool is cast out to disk.

integer is expressed as a percentage of the number of data entries and can range from 0 to 90. The default is 30.

For example, GBPOOLT(55) casts out data if the number of pages in the group buffer pool equals 55% of the group buffer pool page capacity.

GBPCHKPT (*integer*)

Changes the time interval, in minutes, between successive checkpoints of the group buffer pool. *integer* can range from 1 to 999999. Unless a value is explicitly specified for the GBPCHKPT option, the default value is 4 minutes.

The more frequently checkpoints are taken, the less time it takes to recover the group buffer pool if the coupling facility fails.

Usage notes

Defaults: Issuing the ALTER GROUPBUFFERPOOL command does not change any option that is not explicitly specified; the default is to leave the value unchanged. The following table lists the default values for the options when the command is first issued for a group buffer pool or a structure.

Table 5. Default option values when ALTER GROUPBUFFERPOOL is first issued

Option	Value
GBPCACHE	YES
RATIO	5
CLASST	5 (%)
GBPOOLT	30 (%)
GBPCHKPT	4 (minutes)

When new values take effect: When you issue the ALTER GROUPBUFFERPOOL command, some option specifications become effective only at the next allocation of the group buffer pool. The following table lists each option, when the new value takes effect, and if the option is applicable for a group buffer pool that is specified as GBPCACHE(NO).

Table 6. Changing group buffer pool attributes

Keyword	New value takes effect	Applicable if GBPCACHE(NO)?
GBPCACHE	at next allocation	N/A
AUTOREC	immediately	No
RATIO	at next allocation ²	No ³

Table 6. Changing group buffer pool attributes (continued)

Keyword	New value takes effect	Applicable if GBPCACHE(NO)?
CLASST	immediately	No ³
GBPOOLT	immediately	No ³
GBPCHKPT	immediately	No ³

Note:

1. You can use the z/OS command SETXCF START,REBUILD to have the change take effect if the group buffer pool is not duplexed. If the group buffer pool is duplexed and you want to change to GBPCACHE(NO), first go back to simplex mode and rebuild. GBPCACHE(NO) is not allowed for duplexed group buffer pools.
2. You can use the z/OS command SETXCF START,REBUILD to have the change take effect if the group buffer pool is not duplexed. If the group buffer pool is duplexed, first go back to simplex mode and rebuild; then optionally go back to duplex mode. If a group buffer pool is duplexed, both instances of that duplexed group buffer pool use the same RATIO value.
3. DB2 issues message DSNB761 when you specify this option for a GBPCACHE(NO) group buffer pool. These settings only take effect after the GBPCACHE attribute has been changed to YES.

Examples

Example: Changing the ratio of directory entries to data pages

For group buffer pool 0, the following command changes the ratio of directory entries to data pages to one directory entry for every data page. The RATIO specification becomes effective at the next allocation of the group buffer pool.

```
-DB1G ALTER GROUPBUFFERPOOL (GBP0) RATIO(1)
```

Example: Changing the class castout threshold

For group buffer pool 3, change the class castout threshold to 10 %. The new value takes effect immediately. Because the group name is DSNCAT, the coupling facility structure name is DSNCAT_GBP3. Also, when a structure fails, the AUTOREC(YES) option enables DB2 to automatically recover the page sets and partitions that are in a GRECP status or that have pages on the logical page list.

```
-DB1G ALTER GROUPBUFFERPOOL (DSNCAT_GBP3) CLASST(10) AUTOREC(YES)
```

Example: Changing the class castout threshold and the group buffer pool castout threshold

For group buffer pool 2, the following command changes the class castout threshold to 10% and the group buffer pool castout threshold to 50%. The new values take effect immediately.

```
-DB1G ALTER GROUPBUFFERPOOL (GBP2) CLASST(10) GBPOOLT(50)
```

Example: Changing the group buffer pool checkpoint frequency

For group buffer pool 32K, the following command changes the GBP checkpoint frequency to five minutes. The new value takes effect immediately. In this example, with AUTOREC(NO) specified, DB2 does not start automatic recovery when a structure fails. You might choose this option if you want to determine what page sets or partitions are in a GRECP status or have pages on the logical page list before you enter the START DATABASE command to recover the data with the options you specify.

```
-DB1G ALTER GROUPBUFFERPOOL (GBP32K) GBPCHKPT(5) AUTOREC(NO)
```

Chapter 13. -ALTER UTILITY (DB2)

The DB2 command ALTER UTILITY changes the values of certain parameters of an execution of the REORG utility that uses SHRLEVEL REFERENCE or CHANGE and the REBUILD utility that uses SHRLEVEL CHANGE.

Specifically, this command changes the values of DEADLINE, MAXRO, LONGLOG, and DELAY.

REBUILD and REORG can be altered only from the DB2 subsystem on which ALTER UTILITY is running.

ALTER UTILITY applies to a single utility control statement. It has no effect on other utility control statements that are later in the same job step. If a LISTDEF control statement is being processed, ALTER UTILITY applies only to the subset of list items that are currently being processed.

Abbreviation: -ALT UTIL

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or a CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope : Member

Authorization

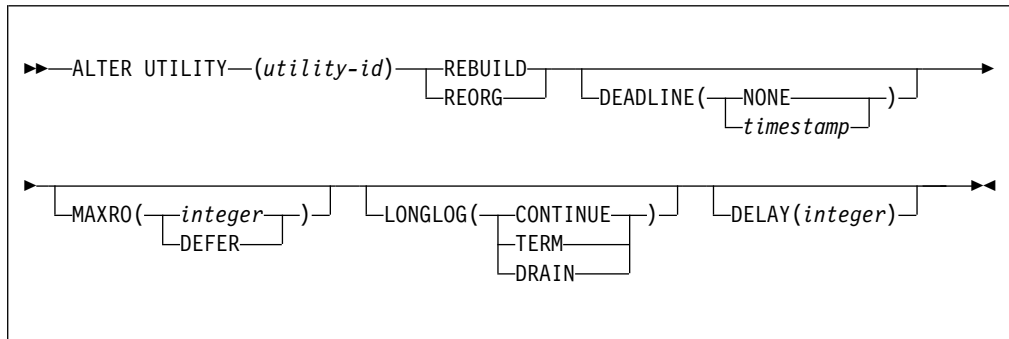
To execute this command, you must use the primary or some secondary authorization ID of the process that originally submitted the utility job. Alternatively, you must use a privilege set of the process that includes one of the following authorities:

- DATAACCESS authority
- DBCTRL authority
- DBADM authority
- System DBADM authority
- DATAACCESS authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

For users with DBMAINT, DBCTRL, or DBADM authority, the command takes effect only when a user has sufficient authority over each object that the utility job accesses.

Syntax



Option descriptions

(utility-id)

Is the utility identifier, or the UID parameter, used when creating the utility job step.

This job must execute REBUILD SHRLEVEL CHANGE, REORG with SHRLEVEL CHANGE, or SHRLEVEL REFERENCE.

If *utility-id* was created by the DSNU CLIST by default, it has the form *tso-userid.control-file-name*.

If *utility-id* was created by default by the EXEC statement that executed DSNUUTILB, it has the form *userid.jobname*.

If *utility-id* contains lowercase letters or special characters, it must be enclosed in single quotation marks (').

REBUILD

Specifies that a REBUILD SHRLEVEL CHANGE utility is being altered.

REORG

Specifies that a REORG SHRLEVEL REFERENCE or REORG SHRLEVEL CHANGE utility is being altered.

DEADLINE

Specifies the deadline by which the user wants the switch phase of reorganization to start.

Only valid when altering a REORG SHRLEVEL REFERENCE or REORG SHRLEVEL CHANGE utility.

If DB2 estimates that the switch phase will not start by the deadline, DB2 terminates reorganization. The default is the value of DEADLINE that is currently in effect.

The pre-switch processing might continue until after the deadline.

(NONE)

Specifies that there is no deadline for the read-only iteration of log processing.

(timestamp)

Specifies the deadline by which the user wants the switch phase to start processing. This deadline must not have been reached when ALTER UTILITY executes.

MAXRO

Specifies the maximum amount of time that is tolerated for the last iteration of log processing during reorganization. During that iteration, applications have read-only access.

The actual execution time of the last iteration can exceed the value specified for MAXRO.

The default is the value of MAXRO that is currently in effect.

(*integer*)

Specifies the number of seconds.

(DEFER)

Specifies that the log phase is deferred indefinitely.

LONGLOG

Specifies the action that DB2 performs (after sending the LONGLOG message to the console) if the number of log records that are processed during the next iteration is not sufficiently lower than the number of log records that were processed during the previous iterations. The default is the value of LONGLOG that is currently in effect.

(CONTINUE)

Specifies that DB2 continues the log phase.

(TERM)

Specifies that DB2 terminates the utility after the delay.

(DRAIN)

Specifies that DB2 drain the write claim class after the delay (if specified). The number of log records, and thus the *estimated* time, for a future iteration of log processing will be 0.

DELAY (*integer*)

Specifies a lower bound for the interval between the time when the utility sends the LONGLOG message to the console and the time when the utility performs the action specified by the LONGLOG parameter.

integer is the delay in seconds. The value must be nonnegative. The default is the value of DELAY that is currently in effect.

Examples

Example: Alter several options for an execution of the REORG utility

The following commands alters the execution of the REORG utility for the utility job step whose utility identifier is REORGEMP:

```
-ALTER UTILITY (REORGEMP) REORG MAXRO(240) LONGLOG(DRAIN)
```

The following list explains what each option does in the preceding example:

- MAXRO(240) changes the maximum tolerable time for the last iteration of log processing to 240 seconds (4 minutes).
- LONGLOG(DRAIN) specifies that DB2 drain the write claim class (if reading of the log during REORG is not catching up to the speed at which the application is writing the log).
- DELAY is not specified so this example does not change the existing delay between sending the LONGLOG message to the console and performing the action specified by LONGLOG.

- DEADLINE is not specified so this example does not change the deadline (if any) that was defined in the last iteration of log processing.

Chapter 14. -ARCHIVE LOG (DB2)

The DB2 command ARCHIVE LOG enables a site to close a current active log and open the next available log data set.

When issued without any options, the DB2 command ARCHIVE LOG performs the following functions:

- Truncates the current active log data sets
- Starts an asynchronous task to offload the data sets
- Archives previous active log data sets not yet archived
- Returns control to the user (immediately)

In a data sharing environment, you can truncate and archive the logs for an individual member or for all members in the group.

When specified with the option MODE(QUIESCE), the ARCHIVE LOG command attempts to quiesce (suspend) all DB2 user update activity on the DB2 active log prior to the offload process. When a system-wide point of consistency is reached (that is, when all currently active update users have reached a commit point), the active log is immediately truncated, and the offload process is initiated. The resulting point of consistency is captured in the current active log before it is offloaded. In a data sharing environment, you can create a system-wide point of consistency only for the entire group.

Abbreviation: -ARC LOG

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

The ARCHIVE LOG command can also be issued from the z/OS subsystem interface (SSI) to enable automated scheduling systems and other programs to execute the command via supervisor call instruction (SVC) 34.

Data sharing scope: Group or member, depending on whether you specify MODE(QUIESCE), or on which SCOPE option you choose

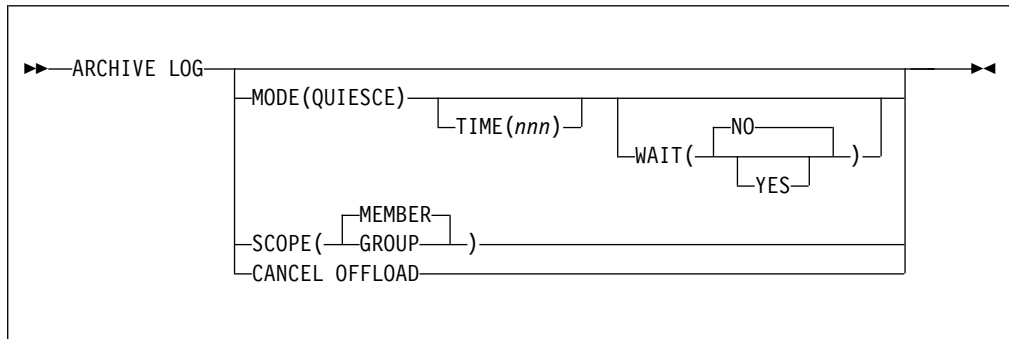
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- ARCHIVE privilege
- Installation SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

MODE (QUIESCE)

Halts all new update activity by the DB2 subsystem for a specified period of time and attempts to bring all existing users to a point of consistency after a commit or rollback. When a point of consistency is reached and captured in the current active log data set, the current active log data set is truncated, and another log data set in the inventory becomes current. Offload processing then begins with the oldest active log data set and ends with the active log data set that was truncated.

In a data sharing environment, before archiving logs of any member, this option quiesces all active members of a data sharing group. MODE (QUIESCE) also ensures that each inactive member had successfully quiesced its update activity and resolved any indoubt units of recovery (URs) before the inactive subsystem completed normal termination. If any DB2 subsystem is in a failed state, fails during quiesce processing, or is stopped with outstanding URs, the ARCHIVE LOG command fails, and the remaining active members allow update activity to proceed.

If no indoubt URs exist on all quiesced members, active or inactive, the archive operation can continue for active members in the group. Thus, you can archive logs of a data sharing group normally without forcing all members to be active. The current logs of inactive members are truncated and offloaded after they start.

If a system-wide point of consistency cannot be reached during the quiesce period, which is a length of time you can specify, execution of the ARCHIVE LOG command fails and an error message is issued. In a data sharing environment, the maximum time period applies for the whole group, and if any DB2 subsystem cannot quiesce within the time allowed, the command fails.

If there is no update activity on DB2 data when the command ARCHIVE LOG MODE (QUIESCE) is issued, the active log is truncated and offloaded immediately.

TIME (*nnn*)

Specifies the maximum length of time, in seconds, in which the DB2 subsystem is allowed to attempt a full system quiesce.

If you do not specify a time, the **default** is the length of time specified in the field QUIESCE PERIOD of installation panel DSNTIPA.

nnn can range from 001 to 999 seconds. You must allocate an appropriate time period for the quiesce processing or the following events can occur:

- The quiesce processing can expire before a full quiesce is accomplished.
- An unnecessary DB2 lock contention can be imposed.
- A timeout can occur.

This option is valid only when used in conjunction with the option MODE(QUIESCE).

WAIT

Specifies whether the DB2 subsystem should wait until the quiesce processing has completed before returning control to the invoking console or program, or should return control when the quiesce processing begins.

This option is valid only when used in conjunction with the option MODE(QUIESCE).

(NO)

Specifies that control must be returned to the invoking program when the quiesce processing begins.

If WAIT(NO) is used, quiesce processing is asynchronous to the user; that is, you can issue additional DB2 commands after the ARCHIVE LOG command returns control to you.

(YES)

Specifies that the quiesce processing must complete before returning control to the invoking console or program.

If WAIT(YES) is used, quiesce processing is synchronous to the user; that is, additional DB2 commands can be issued, but they are not processed by the DB2 command processor until the ARCHIVE LOG command is complete.

SCOPE

Specifies whether the command applies to the entire data sharing group or to a single member only. The SCOPE option is valid only in a data sharing environment; the option is ignored in a non-data-sharing environment. SCOPE cannot be specified if MODE(QUIESCE) is specified; the two keywords are mutually exclusive.

(MEMBER)

Initiates offload processing only for the member from which the command is issued. User update activity is not suspended. If that member, or the entire group, is already archiving, the command fails. This is the default, except when MODE(QUIESCE) is specified.

(GROUP)

Initiates offload processing for every member of the DB2 group. User update activity is not suspended. If any member of the group, or the entire group, is already archiving, the command fails.

CANCEL OFFLOAD

Cancels any off loading currently in progress and restarts the offload process, beginning with the oldest active log data set that has not been off loaded and proceeding through all active log data sets that need off loading. Any suspended offload operations are restarted.

Usage notes

Remote site recovery: The ARCHIVE LOG command is very useful when performing a DB2 backup in preparation for a remote site recovery. For example, the command allows the DB2 subsystem to quiesce all users after a commit point,

and capture the resulting point of consistency in the current active log *before* the archive is taken. Therefore, when the archive log is used with the most current image copy (during an offsite recovery), the number of data inconsistencies will be minimized.

Simultaneous executions: The ARCHIVE LOG command cannot be executed if another ARCHIVE LOG command is in progress. Instead, error message DSNJ318I is issued and the command fails. This is true in both data sharing and non-data-sharing environments. For example, in a data sharing environment, the command fails if the data sharing member, or group to which it belongs, is already archiving.

Available active log space: ARCHIVE LOG cannot be used when the current active log is the last available active log data set because of the following reasons:

- All available active log space would be used.
- The DB2 subsystem would halt processing until an offload is complete.

Executing ARCHIVE LOG while STOP DB2 is in progress: ARCHIVE LOG without the option MODE(QUIESCE) is permitted when STOP DB2 MODE(QUIESCE) is in progress. However, if an attempt is made to execute the ARCHIVE LOG command when a STOP DB2 MODE(FORCE) is in progress, error message DSNJ315I is issued and the ARCHIVE LOG command is not processed.

ARCHIVE LOG with the option MODE(QUIESCE) is not allowed when a STOP DB2 MODE(FORCE) or STOP DB2 MODE(QUIESCE) is in progress. If an attempt is made to run the ARCHIVE LOG command under these circumstances, error message DSNJ315I or DSNJ316I is issued.

If the system was not fully quiesced (as determined by the number of users which could not be quiesced), error message DSNJ317I is issued and ARCHIVE LOG command processing is terminated. The current active log data set is not truncated and switched to the next available active log data set, and the archive log is not created.

Canceling log offloads: It is possible for the offload of an active log to be suspended when something goes wrong with the offload process, such as a problem with allocation or tape mounting. Issuing ARCHIVE LOG CANCEL OFFLOAD interrupts the offload process and restarts the offload. The command causes an abnormal termination of the offload task, which can result in a dump. Use ARCHIVE LOG CANCEL OFFLOAD only if the offload task is no longer functioning, or if you want to restart a previous offload attempt that failed.

Demand on DB2 resources: Using the option MODE(QUIESCE) during times of peak activity or during periods in which time is critical causes a significant disruption in the availability of DB2 for all users of DB2 resources.

Interaction with DISPLAY THREAD: The command DISPLAY THREAD issues message DSNV400I, indicating that an ARCHIVE LOG MODE(QUIESCE) command is active.

Quiescing members of a data sharing group: It is not possible to quiesce a single member of a data sharing group. When MODE(QUIESCE) is specified in a data sharing group, the entire group is quiesced.

Executing ARCHIVE LOG while logging is suspended: While logging is suspended by SET LOG SUSPEND, do not use ARCHIVE LOG unless CANCEL OFFLOAD is specified. If logging is suspended, issue SET LOG RESUME to resume logging before issuing ARCHIVE LOG.

Examples

Example: Truncating the current active log data sets and switching to the next available active log

The following command truncates the current active log data sets and initiates an asynchronous job to offload the truncated data sets. No quiesce processing occurs.

```
-ARCHIVE LOG
```

Example: Initiating the default quiesce period before truncating the active log data sets and switching to the next available active log

The following command initiates a quiesce period. If all DB2 update activity is stopped within this period, truncate the current active log data set and switch to the next available active log data set. Let the value in the field QUIESCE PERIOD of installation panel DSNTIPA determine the length of the quiesce period. The MODE(QUIESCE) processing is asynchronous.

```
-ARCHIVE LOG MODE(QUIESCE)
```

If the DB2 subsystem can successfully block all update activity before the quiesce period ends, it proceeds to the next processing step. If the quiesce time period is insufficient to successfully quiesce the DB2 subsystem, the active log data sets are not truncated and the archive does not occur.

Example: Initiating a quiesce period of a specified length before truncating the active log data sets and switching to the next available active log

The following command initiates a quiesce period. If all DB2 update activity is stopped within this period, truncate the current active log data set and switch to the next available active log data set. The maximum length of the quiesce processing period is seven minutes (420 seconds) and the processing is synchronous for the entire seven minutes.

```
-ARCHIVE LOG MODE(QUIESCE) WAIT(YES) TIME(420)
```

If the DB2 subsystem can successfully block all update activity before the quiesce period ends, it proceeds to the next processing step. If the quiesce time period is insufficient to successfully quiesce the DB2 subsystem, the active log data sets are not truncated and the archive does not occur.

Example: Quiescing all members of a data sharing group before truncating the active log data sets and switching to the next available active log

The following command initiates a quiesce period for all members of the data sharing group. If all DB2 update activity is stopped within this period, truncate the current active log data set and switch to the next available active log data set. Specify a quiesce time period of 10 minutes (600 seconds) to override the value in the field QUIESCE PERIOD of installation panel DSNTIPA for member DB1G. If the update activity has not quiesced after the 10 minute quiesce period, the command fails and new update activity is allowed to proceed.

```
-DB1G ARCHIVE LOG MODE(QUIESCE) TIME(600)
```

Example: Truncating the active log data sets and initiating and offload for a single member of a data sharing group

In a data sharing environment the following command truncates the active

log data sets for group member DB2G and initiate an asynchronous job to offload the truncated data sets, without any quiesce processing. In this example, SCOPE(MEMBER) is used by default.

```
-DB2G ARCHIVE LOG
```

Example: Truncating the active log data sets and initiating and offload for all members of a data sharing group

The following command truncates the data sets for all members of the data sharing group and initiate an asynchronous job to offload the truncated data sets, without any quiesce processing.

```
-DB2G ARCHIVE LOG SCOPE(GROUP)
```

Chapter 15. BIND PACKAGE (DSN)

The DSN subcommand BIND PACKAGE builds an application package. DB2 records the description of the package in the catalog tables and saves the prepared package in the directory.

Environment

You can use BIND PACKAGE from DB2I, or from a DSN session under TSO that runs in either the foreground or background.

Data sharing scope: Group

Authorization

The package owner must have authorization to execute **all** statements embedded in the package for BIND PACKAGE to build a package without producing error messages. (The SYSADM authority includes this authorization.)

| For VALIDATE(BIND), DB2 verifies the authorization at bind time, with the
| exception of the LOCK TABLE statement, and some CREATE, ALTER, and DROP
| statements. For those SQL statements, DB2 verifies the authorization at run time.

For VALIDATE(RUN), DB2 verifies the authorization initially at bind time, but if the authorization check fails, DB2 rechecks it at run time.

The required authorization to add a new package or a new version of an existing package depends on the value of field BIND NEW PACKAGE on installation panel DSNTIPP. The default value is BINDADD.

The package owner must be a role to execute BIND PACKAGE in a trusted context with role ownership. If performing a bind in a trusted context that has a role-as-object owner, then the owner of the package will be a role. If OWNER is specified, then it is assumed to be a role. If it is not specified, then the role of the binder becomes the owner.

| To specify the option SQLERROR(CHECK), the binder must have the BIND,
| BINDAGENT, or EXPLAIN privilege.

| To specify the option EXPLAIN(ONLY), the binder must have the EXPLAIN
| privilege.

The following table summarizes the required authorization to run BIND PACKAGE, depending on the bind options that you specify, and in the case of the ADD option, the value of installation panel field BIND NEW PACKAGE.

Table 7. Summary of privileges needed for BIND PACKAGE options

Bind option	Installation panel field BIND NEW PACKAGE (BINDNV subsystem parameter)	Authorization required to run BIND PACKAGE
ADD, using the default owner or primary authorization ID	BINDADD	<p>The primary authorization ID (default owner) or role must have one of the following to add a new package or new version of an existing package to a collection:</p> <ul style="list-style-type: none"> • The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections • SYSADM, SYSCTRL, or system DBADM authority
ADD, using the default owner or primary authorization ID	BIND	<p>The primary authorization ID (default owner) or role must have one of the following to add a new package or a new version of an existing package to a collection:</p> <ul style="list-style-type: none"> • The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections • SYSADM, SYSCTRL, or system DBADM authority • PACKADM authority on the collection or on all collections • The BIND package privilege (can only add a new version of an existing package)

Table 7. Summary of privileges needed for BIND PACKAGE options (continued)

Bind option	Installation panel field BIND NEW PACKAGE (BINDNV subsystem parameter)	Authorization required to run BIND PACKAGE
ADD, specifying an OWNER other than the primary authorization ID ¹	BINDADD	<p data-bbox="1045 312 1464 716">If any of the authorization IDs or roles of the process has SYSADM authority, SYSCTRL authority, or system DBADM authority, OWNER <i>authorization-id</i> can be any value, when subsystem parameter SEPARATE_SECURITY is set to NO. If any of the authorization IDs has the BINDAGENT privilege granted from the owner, <i>authorization-id</i> can specify the grantor as OWNER. Otherwise, the OWNER <i>authorization-id</i> must be one of the primary or secondary authorization IDs of the binder.</p> <p data-bbox="1045 743 1464 856">If you specify OWNER <i>authorization-id</i>, DB2 first checks the OWNER and then the binder for the necessary bind privilege.</p> <p data-bbox="1045 884 1464 1083">If the binder does not have SYSADM, SYSCTRL, or system DBADM authority, the authorization ID or role of the OWNER must have one of the following to add a new package or new version of an existing package to a collection:</p> <ul data-bbox="1045 1094 1464 1312" style="list-style-type: none"> <li data-bbox="1045 1094 1464 1241">• The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections <li data-bbox="1045 1251 1464 1312">• SYSADM, SYSCTRL, or system DBADM authority

Table 7. Summary of privileges needed for BIND PACKAGE options (continued)

Bind option	Installation panel field BIND NEW PACKAGE (BINDNV subsystem parameter)	Authorization required to run BIND PACKAGE
ADD, specifying an OWNER other than the primary authorization ID ¹	BIND	<p>If any of the authorization IDs or roles of the process has SYSADM authority, SYSCTRL authority, or system DBADM authority, OWNER <i>authorization-id</i> can be any value, when subsystem parameter SEPARATE_SECURITY is set to NO. If any of the authorization IDs has the BINDAGENT privilege granted from the owner, <i>authorization-id</i> can specify the grantor as OWNER. Otherwise, the OWNER <i>authorization-id</i> must be one of the primary or secondary authorization IDs of the binder.</p> <p>If you specify OWNER <i>authorization-id</i>, DB2 first checks the OWNER and then the binder for the necessary bind privilege.</p> <p>If the binder does not have SYSADM, SYSCTRL, or system DBADM authority, the authorization ID or role of the OWNER must have one of the following to add a new package or new version of an existing package to a collection:</p> <ul style="list-style-type: none"> • The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections • SYSADM, SYSCTRL, or system DBADM authority • PACKADM authority on the collection or on all collections • The BIND package privilege (can only add a new version of an existing package)
REPLACE, using the default owner or primary authorization ID	BINDADD or BIND	<p>Primary authorization ID or role must have one of the following:</p> <ul style="list-style-type: none"> • Ownership of the package • BIND privilege on the package. If the package does not exist, see the authorization that is required for "ADD, using the default owner or primary authorization ID." • PACKADM authority on the collection or on all collections • SYSADM, SYSCTRL, or system DBADM authority

Table 7. Summary of privileges needed for BIND PACKAGE options (continued)

Bind option	Installation panel field BIND NEW PACKAGE (BINDNV subsystem parameter)	Authorization required to run BIND PACKAGE
REPLACE, specifying an OWNER other than the primary authorization ID ¹	BINDADD or BIND	<p>If any of the authorization IDs or roles of the process has SYSADM authority, SYSCTRL authority, or system DBADM authority, OWNER <i>authorization-id</i> can be any value, when subsystem parameter SEPARATE_SECURITY is set to NO. If any of the authorization IDs has the BINDAGENT privilege granted from the owner, <i>authorization-id</i> can specify the grantor as OWNER. Otherwise, the OWNER <i>authorization-id</i> must be one of the primary or secondary authorization IDs of the binder.</p> <p>If you specify OWNER <i>authorization-id</i>, DB2 first checks the OWNER and then the binder for the necessary bind privilege.</p> <p>If the binder does not have SYSADM or SYSCTRL or system DBADM authority, the authorization ID or role of the OWNER must have one of the following:</p> <ul style="list-style-type: none"> • BIND privilege on the package. If the package does not exist, the authorization that is required is the same as for ADD, when an OWNER other than the primary authorization ID is specified. • BINDAGENT privilege from the current owner of the package. • PACKADM authority on the collection or on all collections • SYSADM, SYSCTRL, or system DBADM authority
COPY	BINDADD or BIND	<p>The primary or secondary authorization ID or role of the binder or OWNER must have one of the following on the package being copied:</p> <ul style="list-style-type: none"> • Ownership of the package • COPY privilege on the package • BINDAGENT privilege from the owner of the package • PACKADM authority on the collection or on all collections • SYSADM, SYSCTRL, or system DBADM authority

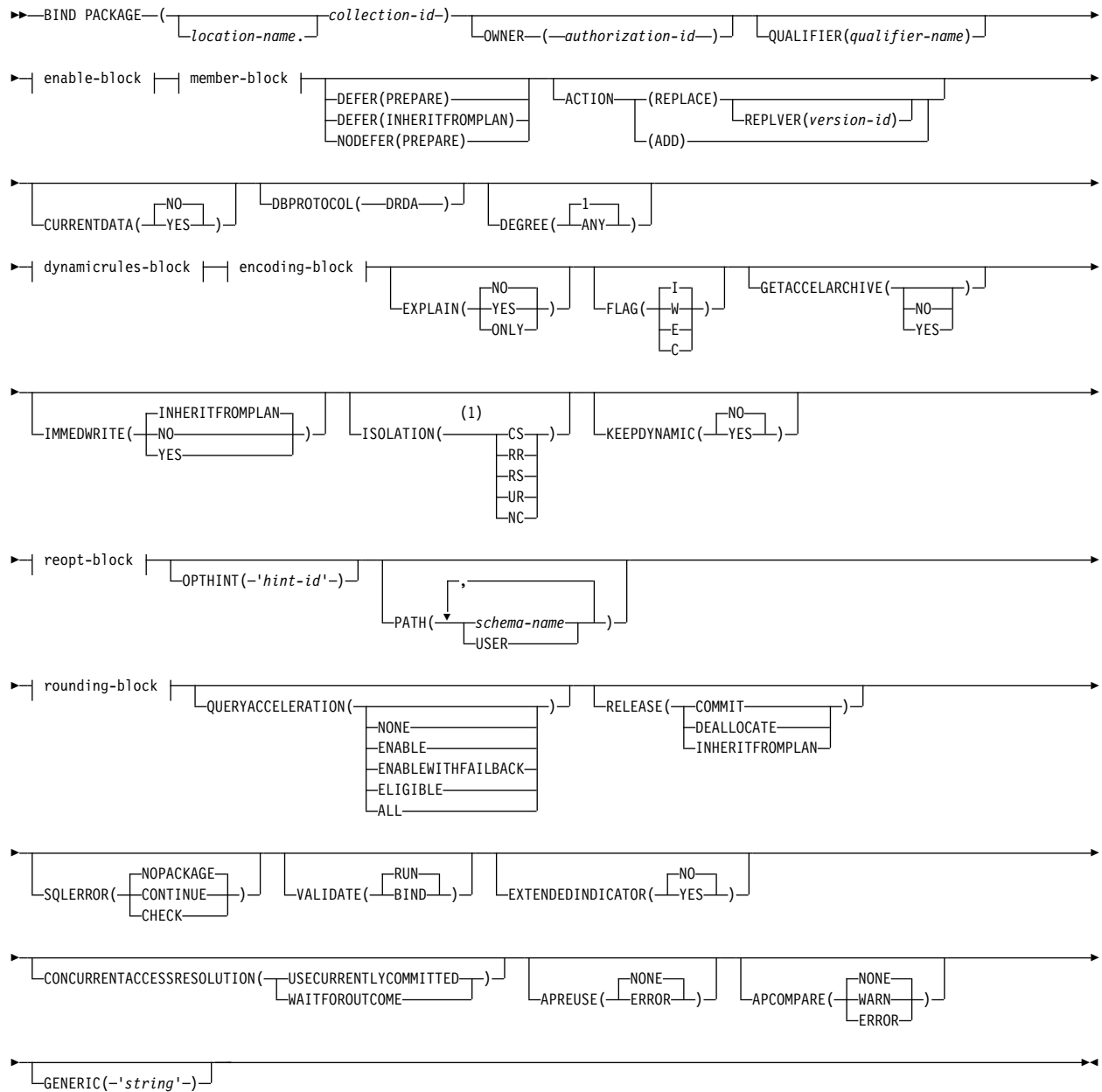
Table 7. Summary of privileges needed for BIND PACKAGE options (continued)

Bind option	Installation panel field BIND NEW PACKAGE (BINDNV subsystem parameter)	Authorization required to run BIND PACKAGE
-------------	--	--

Note:

- 1. If both the OWNER and the binder do not have the necessary bind privilege and the IFCID 140 trace is active, a trace record is written with details about the authorization failure.

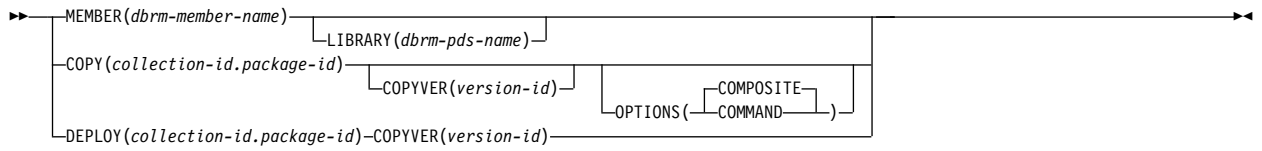
Syntax



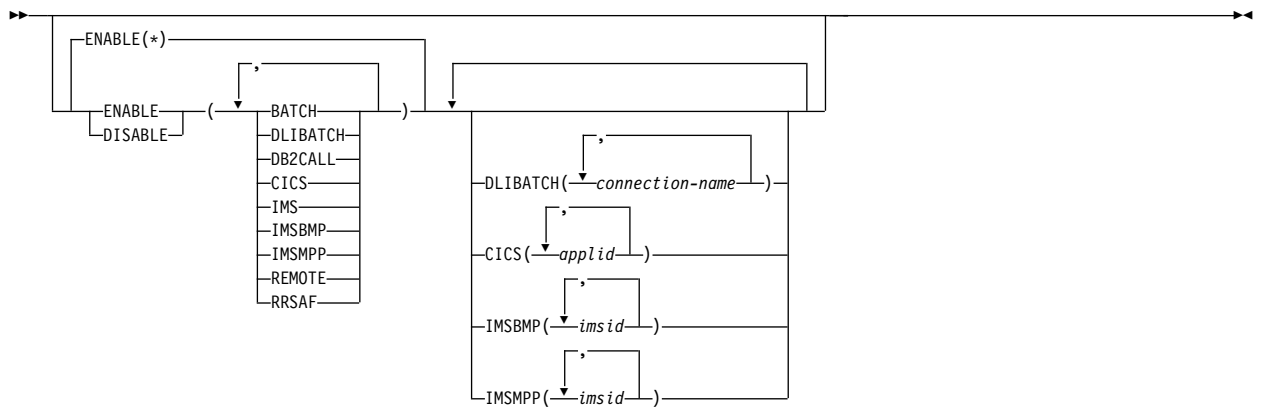
Notes:

- 1 The default for a local package is the plan value. The default for a remote package is CS.

member-block



enable-block



dynamicrules-block



encoding-block



reopt-block



Notes:

- 1 NOREOPT(VARS) can be specified as a synonym of REOPT(NONE)
- 2 REOPT(VARS) can be specified as a synonym of REOPT(ALWAYS)

rounding-block



Option descriptions

For descriptions of the options shown in the syntax diagram, see Chapter 18, "BIND and REBIND options for packages and plans," on page 101.

Examples

Example: Replacing a version of a package

The following command replaces version APRIL_VERSION of package TEST.DSN8BC10 at local location USIBMSTODB22 with another version of the package. The new version (or it could be the same) is in the DBRM DSN8BC10. If the DBRM contains no version ID, the version ID of the package defaults to the empty string. The package runs only from the TSO BATCH environment, and from the CICS environment if the connection ID is CON1. The name PRODUCTN qualifies all unqualified table, view, alias and index names.

```
BIND PACKAGE (USIBMSTODB22.TEST) -  
  MEMBER (DSN8BC10) -  
  ACTION (REPLACE) REPLVER (APRIL_VERSION) -  
  QUALIFIER (PRODUCTN) -  
  ENABLE (BATCH, CICS) CICS (CON1)
```

Example: Binding the SPUFI package with ISOLATION(UR)

UR isolation acquires almost no locks. It is fast and causes little contention, but it reads uncommitted data. Do not use ISOLATION(UR) unless you are sure that your applications and end users can accept the logically inconsistent data that can occur, such as in the case of this example.

Assume that a supervisor routinely executes SQL statements using SPUFI to check the status of parts as they go through the assembly process and to

update a table with the results of her inspection. She does not need to know the exact status of the parts; a small margin of error is acceptable.

The supervisor queries the status of the parts from a production table called ASSEMBLY-STATUS and makes the updates in a non-production table called REPORTS. She uses the SPUFI option AUTOCOMMIT NO and has the habit of leaving data on the screen while she performs other tasks.

If the supervisor executes a version of SPUFI that is bound with ISOLATION(UR), the query for the status of the parts executes without acquiring locks using UR isolation level and the update executes using CS isolation level. Thus, the query does not inadvertently hold locks in the production table, which interferes with the production jobs, and the supervisor has data good enough for her purposes.

The SPUFI application is bound as follows:

```
BIND PACKAGE(DSNESPUR) -  
  COPY(DSNESPCS.DSNESM68) -  
  ACTION(ADD) -  
  ISOLATION(UR)
```

Example: Binding a package for a native SQL procedure

The following command creates a native SQL procedure named CHICAGO.PRODUCTION.MYPROC from the current location procedure TEST.MYPROC. Both native SQL procedures have the same version ABC. The package for native SQL procedure CHICAGO.PRODUCTION.MYPROC.(ABC) has XYZ as QUALIFIER.

```
CREATE PROCEDURE TEST.MYPROC LANGUAGE SQL VERSION ABC ...
```

```
BEGIN  
...  
END
```

```
BIND PACKAGE(CHICAGO.PRODUCTION) DEPLOY(TEST.MYPROC) COPYVER(ABC)  
  ACTION(ADD) QUALIFIER(XYZ)
```

The following command then replaces the native SQL procedure CHICAGO.PRODUCTION.MYPROC version ABC, using the current location native SQL procedure TEST.MYPROC version ABC.

```
BIND PACKAGE(CHICAGO.PRODUCTION) DEPLOY(TEST.MYPROC) COPYVER(ABC)  
  ACTION(REPLACE) REPLVER(ABC)
```

Related concepts:

 The bind process (DB2 Application programming and SQL)

Related reference:

BIND and REBIND options for packages and plans

REBIND PACKAGE (DSN)

BIND PLAN (DSN)

Chapter 16. BIND PLAN (DSN)

The DSN subcommand BIND PLAN builds an application plan. All DB2 programs require an application plan to allocate DB2 resources and support SQL requests made at run time.

Environment

You can use BIND PLAN through DB2I, or from a DSN session under TSO that runs in either the foreground or background.

Data sharing scope: Group

Authorization

The plan owner must have authorization to execute **all** SQL statements embedded in the plan for BIND PLAN to build a plan without producing error messages. This excludes statements included in DBRMs that are bound to packages included in the package list of the plan. The SYSADM authority includes this authorization.

For VALIDATE(BIND), DB2 verifies the authorization at bind time, with the exception of the LOCK TABLE statement, and some CREATE, ALTER, and DROP statements. For those SQL statements, DB2 verifies the authorization at run time.

For VALIDATE(RUN), DB2 verifies the authorization initially at bind time, but if the authorization check fails, DB2 rechecks it at run time.

The plan owner must be a role to execute BIND PLAN in a trusted context with role ownership.

The following table explains the authorization required to run BIND PLAN, depending on the options specified.

Table 8. Summary of privileges needed for BIND PLAN options

Option	Authorization required to run BIND PLAN
ADD, using the default owner or primary authorization ID	Primary authorization ID (default owner) must have one of the following privileges: <ul style="list-style-type: none">• BINDADD privilege• SYSADM, SYSCTRL, or system DBADM authority
ADD, specifying an OWNER other than the primary authorization ID	If the binder does not have SYSADM or SYSCTRL or system DBADM authority, the authorization ID of the new OWNER must have one of the following privileges: <ul style="list-style-type: none">• BINDADD privilege• SYSADM, SYSCTRL, or system DBADM authority
REPLACE, using the default owner or primary authorization ID	Primary authorization ID of the process must have one of the following privileges: <ul style="list-style-type: none">• Ownership of the plan• BIND privilege on the plan. If the plan does not exist, the authorization is the same as for ADD, using the default owner or primary authorization ID.• SYSADM, SYSCTRL, or system DBADM authority

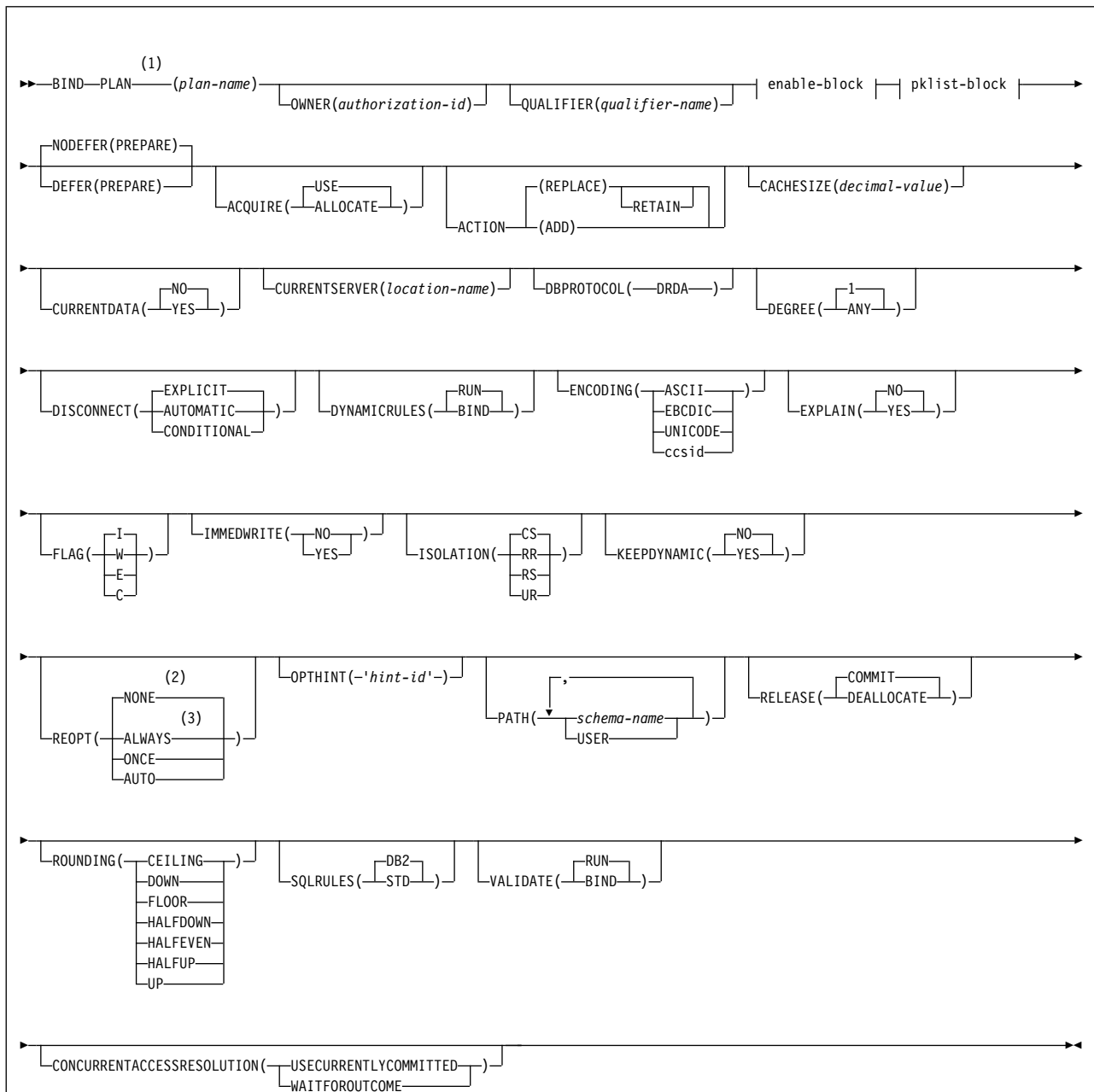
Table 8. Summary of privileges needed for BIND PLAN options (continued)

Option	Authorization required to run BIND PLAN
REPLACE, specifying an OWNER other than the primary authorization ID	<p>If the binder does not have SYSADM or SYSCTRL or system DBADM authority, the authorization ID of the OWNER must have one of the following privileges:</p> <ul style="list-style-type: none"> • Ownership of the plan • BIND privilege on the plan. If the plan does not exist, the authorization is the same as for ADD, specifying an OWNER other than the primary authorization ID. • BINDAGENT privilege from the current owner of the plan. • SYSADM, SYSCTRL, or system DBADM authority
PKLIST, specifying individual packages	<p>Authorization ID of the process must include one of the following privileges:</p> <ul style="list-style-type: none"> • EXECUTE authority on each package specified in the PKLIST • PACKADM authority on specific collections that contain the packages or on all collections • SYSADM or DATAACCESS authority
PKLIST, specifying (*), indicating all packages in the collection	<p>Authorization ID of the process must include one of the following privileges:</p> <ul style="list-style-type: none"> • EXECUTE authority on <i>collection-id</i> .* • PACKADM authority on specific collections that contain the packages or on all collections • SYSADM or DATAACCESS authority
<p>Note: If any of the authorization IDs of the process has SYSADM, SYSCTRL, or system DBADM authority, OWNER <i>authorization-id</i> can be any value, when the system parameter, SEPARATE SECURITY, is set to NO. If any of the authorization IDs has the BINDAGENT privilege granted from the owner, then <i>authorization-id</i> can specify the grantor as OWNER. Otherwise, the OWNER <i>authorization-id</i> must be one of the primary or secondary authorization IDs of the binder.</p>	

Specifying the OWNER for ADD and REPLACE: If any of the authorization IDs of the process has SYSADM authority or SYSCTRL authority, OWNER *authorization-id* can be any value. If any of the authorization IDs has the BINDAGENT privilege granted from the owner, *authorization-id* can specify the grantor as OWNER. Otherwise, OWNER *authorization-id* must be one of the primary or secondary authorization IDs of the binder.

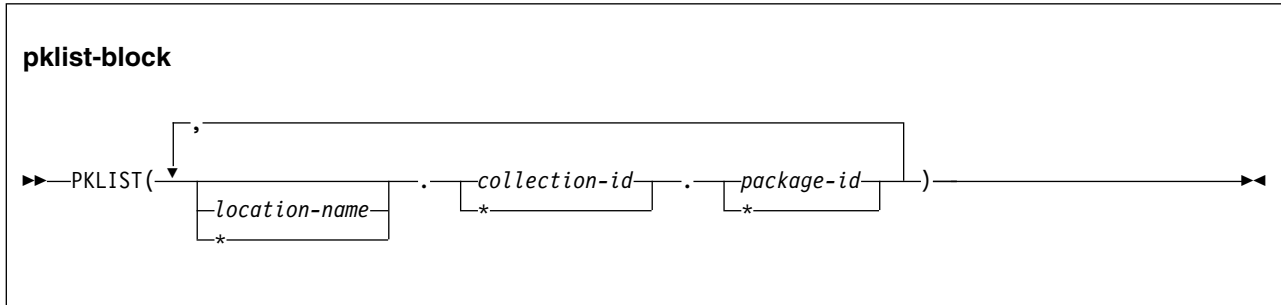
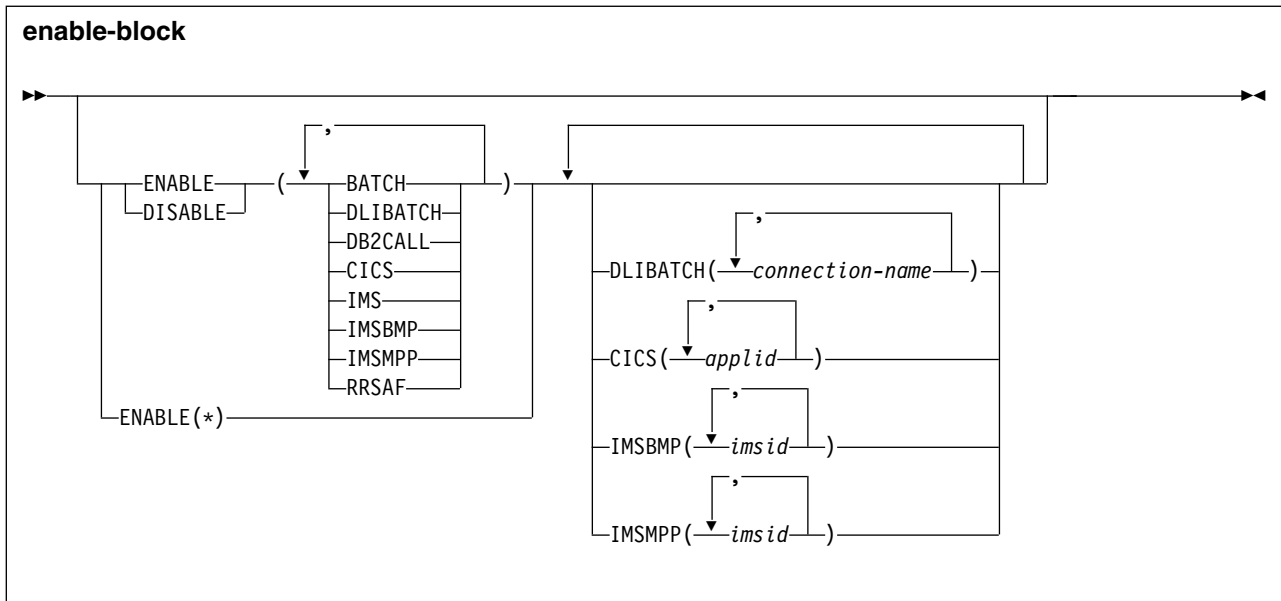
If you specify OWNER *authorization-id* , DB2 first checks the OWNER and then the binder for the necessary bind privilege. If both the OWNER and the binder do not have the necessary bind privilege and the IFCID 140 trace is active, the trace record is written.

Syntax



Notes:

- 1 If PLAN is not specified, all bind functions will be performed, including error diagnostics, without producing an application plan and without inserting rows into PLAN_TABLE for the option EXPLAIN.
- 2 NOREOPT(VARS) can be specified as a synonym of REOPT(NONE)
- 3 REOPT(VARS) can be specified as a synonym of REOPT(ALWAYS)



Option descriptions

For descriptions of the options shown in the syntax diagram, see BIND and REBIND options for packages and plans.

Usage notes

The MEMBER option is deprecated

Although the BIND PLAN command accepts the MEMBER option, the MEMBER option is deprecated. Use it only when you cannot run BIND PACKAGE to bind DBRMs into packages explicitly. When you specify MEMBER, DB2 binds the DBRMs into packages, and binds the packages to the specified plan.

Examples

Example: Binding a plan with ISOLATION(CS) for maximum concurrency

This subcommand creates a new plan called IMSONLY. The SQL statements for the plan are in the package DSN8IC10.

An ISOLATION level of cursor stability (CS) provides maximum concurrency when you run the plan, and protects database values only while the program uses them. DEPTM92 owns the plan, but PRODUCTN qualifies any unqualified table, view, index, and alias names that are referenced in the package.

A cache size of 0 indicates that users will not run the plan repeatedly. Caching the names of users authorized to run the plan helps only when the same user runs the plan repeatedly while it is in the EDM pool. Because this is not the case with this plan, there is no need to reserve space in the EDM pool for a cache that the plan does not use.

The option ENABLE(IMS) runs the plan only from an IMS environment (DLI Batch, BMP and MPP). If you attempt to run the plan from another environment, such as TSO Batch, the plan allocation fails.

```
BIND PLAN(IMSONLY) -  
  PKLIST(DSN8IC10.*) -  
  ACTION(ADD) -  
  ISOLATION(CS) -  
  OWNER(DEPTM92) -  
  QUALIFIER(PRODUCTN) -  
  CACHESIZE -  
  ENABLE(IMS)
```

Example: Binding a plan for a CICS environment only

The following subcommand creates a new plan called CICSONLY. The plan specifies an isolation level of cursor stability (CS). DEPTM12 owns the plan, but TESTSYS qualifies any unqualified table, view, index, and alias names referenced in the package.

The option ENABLE(CICS) CICS(CON1) runs the plan only from CICS VTAM[®] node CON1 which is specified in the APPLID parameter of the CICS SIT table. If you attempt to run the plan from another environment or from another CICS VTAM note, the run attempt fails. A cache size of 0 indicates that users will not run the plan repeatedly.

```
BIND PLAN(CICSONLY) -  
  PKLIST(DSN8IC10.*) -  
  ACTION(ADD) -  
  ISOLATION(CS) -  
  OWNER(DEPTM12) -  
  QUALIFIER(TESTSYS) -  
  CACHESIZE(0) -  
  ENABLE(CICS) CICS(CON1)
```

Related concepts:

 The bind process (DB2 Application programming and SQL)

Related reference:

BIND and REBIND options for packages and plans

BIND PACKAGE (DSN)

REBIND PLAN (DSN)

Chapter 17. BIND QUERY (DSN)

The DSN subcommand BIND QUERY reads the statement text, default schema, and a set of bind options from every row of DSN_USERQUERY_TABLE, and information from correlated EXPLAIN table rows. When LOOKUP(NO) is in effect, DB2 inserts the pertinent data into certain catalog tables.

The data inserted in the catalog tables creates one or more of the following methods for influencing access path selection for matching SQL statements:

- Specifying statement-level optimization parameters.
- Specifying statement level access paths.

Environment

You can use BIND QUERY from DB2I, or from a DSN session under TSO that runs in either the foreground or background. You can also use the SYSPROC.ADMIN_COMMAND_DSN to submit this subcommand from a remote requester.

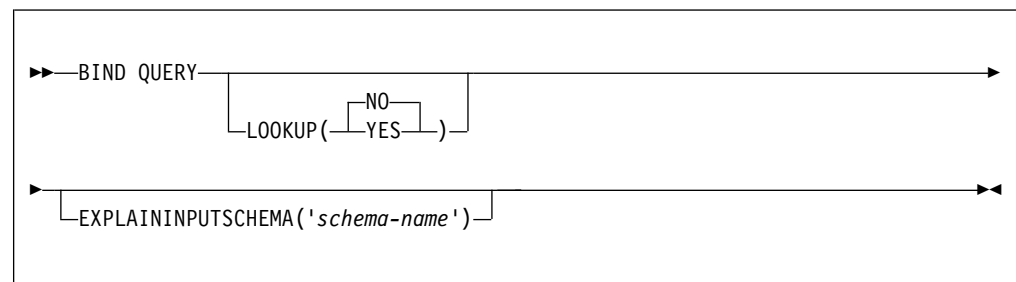
Data sharing scope: Group

Authorization

To issue this command, you must use a privilege set of the process that includes one of the following authorities:

- SQLADM authority
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

Syntax



Option description

LOOKUP

Specifies whether to check catalog tables for existing rows that match rows in DSN_USERQUERY_TABLE. LOOKUP(NO) is the default value. When LOOKUP(YES) is in effect, rows are not inserted or modified in the catalog tables. When matching rows are found in the SYSIBM.SYSQUERY and SYSIBM.SYSQUERYPLAN catalog tables, DB2 inserts the value of the SYSQUERYPLAN.QUERYID column into the DSN_USERQUERY_TABLE.QUERYID column of the matching row.

- A single DSNT290I message if some matching rows were found or a DSNT291I message if no matches were found.

EXPLAININPUTSCHEMA

Specifies the schema name of the EXPLAIN tables that are to be used for input during BIND QUERY processing. The schema name must be enclosed in single quotation marks (').

EXPLAININPUTSCHEMA enables you to create separate EXPLAIN tables to be used only as input to the BIND QUERY command. By creating separate input tables, you can eliminate the need to remove unneeded rows that might interfere with BIND QUERY process from the EXPLAIN output tables. When the EXPLAININPUTSCHEMA option is not specified, DB2 uses the tables that are qualified by the authorization ID of the user that issues the BIND QUERY command.

Usage notes

Eligible SQL statements: BIND QUERY only processes the following types of the SQL statements.

- SELECT
- INSERT with fullselect
- Searched UPDATE
- Searched DELETE
- MERGE
- TRUNCATE

If you enter any SQL statement types other than those on the list above, DB2 issues a message.

DECP options: The following options must be the same when the BIND QUERY command is issued as when packages were bound, for static SQL statements, or when the statements were prepared, for dynamic SQL statements:

- CCSID
- Decimal point
- String delimiter

Cached dynamic SQL statements: During BIND QUERY time, after DB2 processes a query successfully, the query will be removed from the dynamic statement cache when the following conditions are met.

- The query is a dynamic SQL statement
- The access path to be specified exists in the PLAN_TABLE
- The query was prepared and saved in the dynamic statement cache before the BIND QUERY time and value of the OPTHINTS subsystem parameter is set to 'YES'

Examples

Example: Specifying statement-level optimization parameters

Suppose that you created an instance of DSN_USERQUERY_TABLE under your schema, and populated it with rows that specify optimization parameters. Use the following subcommand to create corresponding rows in the SYSIBM.SYSQUERY and SYSIBM.SYSQUERYOPTS catalog tables.

```
BIND QUERY
```

| **Related tasks:**

| [↗](#) Influencing access path selection (DB2 Performance)

| **Related reference:**

| BIND and REBIND options for packages and plans

| [↗](#) SYSIBM.SYSQUERY table (DB2 SQL)

| [↗](#) SYSIBM.SYSQUERYPLAN table (DB2 SQL)

| **Related information:**

| [↗](#) DSNT280I (DB2 Messages)

| [↗](#) DSNT281I (DB2 Messages)

| [↗](#) DSNT290I (DB2 Messages)

| [↗](#) DSNT291I (DB2 Messages)

Chapter 18. BIND and REBIND options for packages and plans

There are several options you can use for binding or rebinding plans and packages. Some of the options are common for both bind and rebind and for both plans and packages.

Defaults: The *default* for an option is the value used if you omit the entire option.

A default of *plan value* for BIND PACKAGE means that the default is the same as the value determined during the bind or rebind of the plan to which the package is appended at run time.

A default of *existing value* for REBIND PLAN or REBIND PACKAGE means that the default is the value that was determined during the previous bind or rebind of the plan or package that you are rebinding.

Catalog records: The DB2 catalog records information about plans and packages, chiefly in the tables SYSIBM.SYSPLAN and SYSIBM.SYSPACKAGE. The descriptions of where the options record information omit the constant qualifier, SYSIBM, of those table names.

For all other cases, the option descriptions note the specific defaults, which DB2 assigns at bind time. If a specific default value exists, that value is underlined.

ACQUIRE bind option

The ACQUIRE bind option specifies that resources for the packages in the plan are to be acquired when the application first accesses them.

ACQUIRE	(<u>USE</u>)	On: BIND and REBIND PLAN
	(ALLOCATE) (deprecated)	

This behavior is the default, regardless of whether you specify ACQUIRE(USE), ACQUIRE(ALLOCATE), which has been deprecated, or neither option.

The ACQUIRE option does not affect page, row, LOB or XML locks.

(USE)

Acquires table space locks only when the application program first uses them.

(ALLOCATE)

This option has no effect. ALLOCATE is deprecated. If you specify ACQUIRE(ALLOCATE) DB2 issues a warning and uses ACQUIRE(USE).


Defaults:

Process	Default value
BIND PLAN	USE
BIND PACKAGE	N/A
REBIND PLAN	USE
REBIND PACKAGE	N/A

There is no ACQUIRE option for packages. A package always acquires resources when it first uses them, as if you specified ACQUIRE(USE).

Catalog record: Column ACQUIRE of table SYSPLAN.

Related concepts:

 Bind options for locks (DB2 Performance)

Related reference:

RELEASE bind option

ACTION bind option

The ACTION option determines whether the object (plan or package) replaces an existing object with the same name or is new.

ACTION	(REPLACE) (REPLACE) REPLVER (BIND PACKAGE only) (REPLACE) RETAIN (BIND PLAN only) (ADD)	On: BIND PLAN and PACKAGE
--------	--	---

(**REPLACE**)

The object replaces an existing one with the same identifier, and a new entry replaces the old one in the catalog table SYSPLAN or SYSPACKAGE. If no object with the given identifier already exists, the bind process creates the new object and a new entry.

The authorization ID designated explicitly or implicitly by the option OWNER becomes the owner of the new object. If that authorization ID is not the previous owner, all grants of privileges for the object that the previous owner issued change to name the new owner as the grantor.

If the bind fails, the old object and its entry remain.

For BIND PACKAGE: You cannot use REPLACE with a remote package bound with either of the options ENABLE or DISABLE. The attempt causes the bind to fail.

REPLVER (*version-id*) (For BIND PACKAGE only)

Replaces a specific version of the package, identified by *version-id*. If the package with the specified *version-id* does not exist, the bind fails.

The default for *version-id* comes from the DBRM if you use the MEMBER option on BIND, or from the COPYVER option if you use the COPY option.

RETAIN (For BIND PLAN only)

Preserves EXECUTE privileges when you replace the plan. If ownership of the plan changes, the new owner grants the privileges BIND and EXECUTE to the previous owner.

RETAIN is *not* the default. If you do not specify .RETAIN, everyone but the plan owner loses the EXECUTE privilege (but not the BIND privilege). If plan ownership changes, the new owner grants the BIND privilege to the previous owner.

(**ADD**)

Adds a new object, but does not replace an existing one. If the object name already exists in the catalog, the bind fails. If the bind fails for any reason, the bind process does not produce a new package or plan and makes no entry in the catalog.

Replacing a version of a package (REPLVER): ACTION(REPLACE) REPLVER can have different effects, depending on the situation. For example, assume that DBRM1 is the name of the member that contains the DBRM for the package, and that A and B are the names of two versions of the package. Suppose that you bind version A with the following command:

```
BIND PACKAGE(COLL1) MEMBER(DBRM1) ACTION(REPLACE) REPLVER(B)
```

- If neither DBRM1, version A, nor version B exist in the DB2 catalog, the command fails because version B is not in the catalog. No new package is added.
- If DBRM1 and version B, but not version A, exist in the DB2 catalog, then version A replaces version B. As a result, version A exists in the catalog, and version B no longer exists in the catalog.
- If DBRM1 and version A exist in the catalog, but not version B, the command fails because version B is not in the catalog. Version A continues to exist.
- If DBRM1 and both versions A and B exist in the catalog, the command fails because version A already exists.

Defaults:

Process	Default value
BIND PLAN	REPLACE
BIND PACKAGE	REPLACE
REBIND PLAN	REPLACE
REBIND PACKAGE	N/A

Catalog record: Tables SYSPLAN or SYSPACKAGE.

APCOMPARE bind option

The APCOMPARE option determines whether the new access paths are different from the older access paths.

APCOMPARE	(NO) (NONE) (WARN) (ERROR)	On: BIND PACKAGE, REBIND PACKAGE, and REBIND TRIGGER PACKAGE
-----------	-------------------------------------	--

(NO)
DB2 does not compare access paths.

(NONE)
DB2 does not compare access paths.

(WARN)
DB2 compares the old and new access paths for each matching statement. If the access paths are structurally dissimilar, DB2 sends message DSNT285I and continues processing the package.

(ERROR)
DB2 compares the old and new access paths for each matching statement. If the access paths are structurally dissimilar, DB2 sends message DSNT285I and terminates package processing.


The APCOMPARE option is ignored for packages and package copies that were bound prior to DB2 9. You can identify the version on which a package or package

copy was bound by checking the RELBOUND column in the SYSIBM.SYSPACKAGE or SYSIBM.SYSPACKCOPY catalog table.


Defaults:


Process	Default value
BIND PLAN	N/A
BIND PACKAGE	NONE
REBIND PLAN	NONE
REBIND PACKAGE	NONE

Related concepts:


 Temporal tables and data versioning (DB2 Administration Guide)


Related tasks:

 Analyzing access path changes at bind or rebind (DB2 Performance)

 Reusing and comparing access paths at bind and rebind (DB2 Performance)

Related reference:

 DB2 catalog tables (DB2 SQL)

 PLAN_TABLE (DB2 Performance)

APRETAINDUP bind option

The APRETAINDUP option determines whether or not DB2 retains an old package copy when access paths of the old copy are identical to the incoming copy.

APRETAINDUP	(<u>YES</u>)	On: REBIND PACKAGE, and
	(NO)	REBIND TRIGGER PACKAGE

This option only applies when PLANMGMT(BASIC) or PLANMGMT(EXTENDED) are in effect.

(YES)

DB2 retains an old package copy

(NO)

DB2 does not retain an old package copy

Defaults:

Process	Default value
BIND PLAN	N/A
BIND PACKAGE	N/A
REBIND PLAN	N/A
REBIND PACKAGE	YES
REBIND TRIGGER PACKAGE	YES

Related reference:

PLANMGMT bind option

APREUSE bind option

The APREUSE option specifies whether DB2 tries to reuse previous access paths for SQL statements in a package. DB2 uses information about the previous access paths from the directory to create a hint.

APREUSE	(NONE) (NO) (ERROR)	On: BIND PACKAGE, REBIND PACKAGE, and REBIND TRIGGER PACKAGE
---------	---------------------------	--

The access path hint is not guaranteed to succeed in all cases. For example, a hint for an access path that relies on objects (such as indexes) that no longer exist cannot be reused. Version incompatibilities might also prevent access paths from being reused. Some access paths cannot be reused because of ambiguity in underlying hint. For example, a hint to use a merge join indicates the type of join to use and the number of matching columns, but the names of the matching columns are not available.

When APREUSE(ERROR) is specified for a BIND PACKAGE command, DB2 tries to locate the access path information from a package that has a matching identity based on the following criteria:

- Location
- Collection ID
- Name
- Version

If no such a package exists, DB2 tries to locate the most recently created version of a package that otherwise matches. When a prior version of a matching package is reused, DB2 issues a DSNT294I message. Even if a prior version exists, the set of static SQL statements in a previous version might not be identical to the set of statements in the new package. In such a situation, the APREUSE option only applies to statements that are identical in both versions.

DB2 issues a DSNT292I warning message when it cannot locate any previous package, and ignores the APREUSE option for that package.

DB2 reports the number of statements that were processed, the number of statements that reused the previous access path and the number of statements that could not reuse the previous access path in a DSNT286I message.

(NONE)

DB2 does not try to reuse previous access paths for statements in the package.

(NO)

An accepted synonym for NONE.

(ERROR)

DB2 tries to reuse the previous access paths for SQL statements in the package. If statements in the package cannot reuse the previous access path, the bind operation for the package that contains the statement ends, and processing continues for the next package. DB2 indicates the number of statements that cannot be reused in any package in a message. New and changed statements in a package never prevent the completion of the bind or rebind operation, even though no previous access path is available for reuse.

The APREUSE option is ignored for package copies that were bound prior to DB2 9. You can identify the version on which a package or package copy was bound by checking the RELBOUND column in the SYSIBM.SYSPACKAGE or SYSIBM.SYSPACKCOPY catalog table.

Defaults:

Process	Default value
BIND PLAN	N/A
BIND PACKAGE	NO
REBIND PLAN	N/A
REBIND PACKAGE	NO
REBIND TRIGGER PACKAGE	NO
Automatic rebind	NO

Related concepts:

[Temporal tables and data versioning \(DB2 Administration Guide\)](#)

Related tasks:

[Reusing and comparing access paths at bind and rebind \(DB2 Performance\)](#)

Related reference:

[DB2 catalog tables \(DB2 SQL\)](#)

[PLAN_TABLE \(DB2 Performance\)](#)

CACHESIZE bind option

The CACHESIZE option determines the size (in bytes) of the authorization cache acquired in the EDM pool for the plan.

CACHESIZE	(<i>decimal-value</i>)	On: BIND and REBIND PLAN
-----------	--------------------------	--------------------------

At run time, the authorization cache stores user IDs authorized to run. Consulting the cache can avoid a catalog lookup for checking authorization to run the plan.

decimal-value

The size of the cache can range from 0 to 4096. Nonzero values that are not multiples of 256 round to the next highest multiple of 256. CACHESIZE(0) specifies creating no cache when the plan runs.

Defaults:

Process	Default value
BIND PLAN	Value of AUTHCACH subsystem parameter
BIND PACKAGE	N/A
REBIND PLAN	Existing value
REBIND PACKAGE	N/A

Catalog record: Column CACHESIZE of table SYSPLAN.

COLLID bind option

The COLLID option specifies that any DBRMs in the plan are to be bound to packages.

COLLID	<i>(collection-id)</i> (*)	On: REBIND PLAN
---------------	--------------------------------	------------------------

The resulting packages are then bound to the specified plan. The resulting plan includes only packages. It does not include any DBRMs.

The COLLID option applies to only those plans that were created prior to DB2 10 and have DBRMs bound directly into the plan.

collection-id

Specifies the collection identifier for the new packages.

- * Specifies that the default collection identifier, `DSN_DEFAULT_COLLID_plan-name`, is to be used for the new packages.

When COLLID(*) is specified, subsystem parameter `DISALLOW_DEFAULT_COLLID` must be set to NO for REBIND to be successful.

If you do not specify COLLID in REBIND PLAN, and the plan was created prior to DB2 10 and has DBRMs bound directly into it, COLLID(*) is in effect.

Catalog record: Tables SYSPACKAGE, SYSPACKAUTH, SYSPACKDEP, SYSPACKSTMT, and SYSPACKSYSTEM.

CONCURRENTACCESSRESOLUTION bind option

The CONCURRENTACCESSRESOLUTION option specifies which concurrent access resolution option to use for statements in a package.

CONCURRENTACCESSRESOLUTION	(WAITFOROUTCOME) (USECURRENTLYCOMMITTED)	On: BIND and REBIND PLAN and PACKAGE, REBIND TRIGGER PACKAGE
		Not valid for REBIND of a native SQL procedure package.

(USECURRENTLYCOMMITTED)

Specifies that when a read transaction requires access to a row that is locked by an INSERT or DELETE operation, the database manager can access the currently committed row, if one exists, and continue to the next row. The read transaction does not need to wait for the INSERT or DELETE operation to commit. This clause applies when the isolation level in effect is cursor stability or read stability.

Under the following circumstances, if USECURRENTLYCOMMITTED is specified for a package, WAITFOROUTCOME behavior is used instead:

- The table space on which the package operates is not a universal table space.

- XML data is being selected, and the data does not support multiple XML versions. In this case, the DB2 database manager cannot determine whether the data has been committed.

(WAITFOROUTCOME)

Specifies that when a read transaction requires access to a row that is locked by an INSERT or DELETE operation, the read transaction must wait for a COMMIT or ROLLBACK operation to complete.

Defaults:

Process	Default value
BIND PLAN	None
BIND PACKAGE	None
REBIND PLAN	None
REBIND PACKAGE	None
REBIND TRIGGER PACKAGE	None

CONCURRENTACCESSRESOLUTION has no default. The following table shows how DB2 handles uncommitted INSERTs for all combinations of CONCURRENTACCESSRESOLUTION settings and subsystem parameter SKIPUNCI settings.

Table 9. DB2 behavior for combinations of settings for subsystem parameter SKIPUNCI and bind option CONCURRENTACCESSRESOLUTION

SKIPUNCI value	CONCURRENTACCESSRESOLUTION value	Skip uncommitted INSERTs or wait for COMMIT or ROLLBACK
YES	USECURRENTLYCOMMITTED	Skip uncommitted INSERTs
YES	WAITFOROUTCOME	Wait for COMMIT or ROLLBACK
YES	Not specified	Skip uncommitted INSERTs
NO	USECURRENTLYCOMMITTED	Skip uncommitted INSERTs
NO	WAITFOROUTCOME	Wait for COMMIT or ROLLBACK
NO	Not specified	Wait for COMMIT or ROLLBACK

COPY bind option

The COPY option copies an existing package, and names that package.

COPY	(<i>collection-id.package-id</i>) (<i>collection-id.package-id</i>) COPYVER	On: BIND PACKAGE
-------------	---	-------------------------

Copying the package recalculates the access paths in the copy.

To create a remote copy, this option copies SQL statements from a package at your local server. Therefore, you must hold the COPY privilege or its equivalent at the **local** server.

collection-id

The name of the collection that contains the package to copy, as listed in column COLLID of catalog table SYSPACKAGE.

package-id

The name of the package to copy, as listed in column NAME of catalog table SYSPACKAGE.

COPYVER(*version-id*)

Determines the version of the package to copy. The default for *version-id* is the empty string.

Restrictions:

- *collection-id.package-id* must identify a package on the local server.
- You cannot copy to a package in the same collection. If you make the copy on the local server, *collection-id.* on the COPY option must not name the collection used on the PACKAGE option.
- DB2 uses the ISO format for output values unless the SQL statement explicitly specifies a different format. Input values can be specified in one of the standard formats, or in a format that is recognized by the server's local date/time exit.

Defaults:

Process	Default value
BIND PLAN	N/A
BIND PACKAGE	None
REBIND PLAN	N/A
REBIND PACKAGE	N/A

COPY has **no default**. If you do not use COPY, you must use MEMBER. You cannot use both options.

The option values of the package copied (**except** the values of ENABLE, DISABLE, OWNER, and QUALIFIER) become the defaults for binding the new package. You can override a default by choosing a new value for an option on the BIND PACKAGE command.

Copy packages to remote servers: To copy and bind packages from DB2 10 for z/OS to some other server that does not support all the new BIND options in DB2 10, use the OPTIONS(COMMAND) option on BIND PACKAGE COPY. Any options you do not explicitly specify on the BIND PACKAGE subcommand are set to the server's defaults. Using this option can prevent bind errors when you bind and copy packages to servers other than DB2 10 for z/OS.

BIND PACKAGE for remote SQL: To execute SQL statements in the native SQL procedure after a CONNECT SQL statement or SQL statements that contain a three-part name from a remote server, you need a package at the target server. Use the BIND PACKAGE COPY command to specify the following:

- Target location as the target site on the CONNECT/implicit DRDA SQL
- Collection ID as the native SQL procedure's schema
- Package ID as the native SQL procedure's name
- COPYVER as the native SQL procedure's version

Example 1: To copy a version to a remote server using the BIND PACKAGE command, issue::

```
CREATE PROCEDURE TEST.MYPROC LANGUAGE SQL VERSION ABC ...
BEGIN
...
```

```
CONNECT TO SAN_JOSE
...
END
BIND PACKAGE(SAN_JOSE.TEST) COPY(TEST.MYPROC) COPYVER(ABC) ACTION(ADD)
```

BIND PACKAGE with SET CURRENT PACKAGESET and SET CURRENT PACKAGE PATH: To use the SQL statements SET CURRENT PACKAGESET and SET CURRENT PACKAGE PATH, you must use the BIND PACKAGE COPY command to specify the following:

- Target collection ID as the target of the SQL statements
- Source collection ID as the native SQL procedure's schema
- Package ID as the native SQL procedure's name
- COPYVER as the native SQL procedure's version

Example 2: To use the SQL statement SET CURRENT PACKAGESET with the BIND PACKAGE command , issue:

```
CREATE PROCEDURE TEST.MYPROC LANGUAGE SQL VERSION ABC ...
BEGIN
...
SET CURRENT PACKAGESET = 'COLL2'
...
END
BIND PACKAGE(COLL2) COPY(TEST.MYPROC) COPYVER(ABC)
ACTION(ADD) QUALIFIER(XYZ)
```

If you need to create a new copy because the native SQL procedure has changed and requires a regeneration, use the BIND COPY ACTION(REPLACE) command.

Catalog record: Column COPY of table SYSPACKAGE.

CURRENTDATA bind option

The CURRENTDATA option determines whether to require data currency for read-only and ambiguous cursors when the isolation level of cursor stability is in effect. It also determines whether block fetching can be used for distributed, ambiguous cursors.

CURRENTDATA	(NO) (YES)	On: BIND and REBIND PLAN and PACKAGE, REBIND TRIGGER PACKAGE
		Not valid for REBIND of a native SQL procedure package.

(NO) Specifies that currency is not required for read-only and ambiguous cursors. Block fetching for distributed, ambiguous cursors is allowed.

If your application attempts to dynamically prepare and execute a DELETE WHERE CURRENT OF statement against an ambiguous cursor, after that cursor is opened, use of CURRENTDATA(NO) is not recommended. You receive a negative SQLCODE if your application attempts a DELETE WHERE CURRENT OF statement for any of the following cursors:

- A cursor that is using block fetching
- A cursor that is using query parallelism
- A cursor that is positioned on a row that is modified by this or another application process

(YES)

Specifies that currency is required for read-only and ambiguous cursors. DB2 acquires page or row locks to ensure data currency. Block fetching for distributed, ambiguous cursors is inhibited.

Restriction for remote rebinds: You cannot use CURRENTDATA when rebinding a package at a remote server. To change the value of CURRENTDATA, you can:

- Issue BIND REPLACE, remotely or locally.
- Free the package and issue BIND ADD, remotely or locally.
- Rebind the package locally at the location where the package resides.

Defaults:

Process	Default value
BIND PLAN	NO
BIND PACKAGE	NO
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column DEFERPREP of table SYSPACKAGE and column EXPREDICATE of table SYSPLAN.

CURRENTSERVER bind option

The CURRENTSERVER option determines the location to connect to before running a plan.

CURRENTSERVER	(<i>location-name</i>)	On: BIND and REBIND PLAN
---------------	--------------------------	--------------------------

The column CURRENTSERVER in catalog table SYSPLAN records the value of *location-name*. The special register CURRENT SERVER also receives that value at the server when the plan is allocated.

You can use CURRENTSERVER to cause a local application to use data from a remote server without changing the application; however, using CURRENTSERVER causes poor performance and should be avoided where possible. Avoid using CURRENTSERVER with applications that contain explicit CONNECT statements.

location-name

The name of the location to connect to. The catalog table SYSIBM.LOCATIONS must contain this name. If the table does not exist, if the table does not contain the DBMS, or if there are no packages at that location, warning messages occur.

SQL return codes: CURRENTSERVER causes DB2 to execute a CONNECT statement. DB2 does not display or report to the application program any warnings that this CONNECT returns. To display the warnings, use explicit CONNECT statements rather than the CURRENTSERVER bind option.

Defaults:

Process	Default value
BIND PLAN	Local database system (regardless of the name of the local location)
BIND PACKAGE	N/A

Process	Default value
REBIND PLAN	Existing value
REBIND PACKAGE	N/A

Catalog record: Column CURRENTSERVER of table SYSPLAN.

DBPROTOCOL bind option

The DBPROTOCOL option specifies the protocol to be used when connecting to a remote site that is identified by a three-part name statement. The only value allowed is DRDA.

DBPROTOCOL	(<u>DRDA</u>)	On: BIND and REBIND PLAN and PACKAGE
		Not valid for REBIND of a native SQL procedure package.

A copy of the package must be bound to each remote site that is referenced by a three-part name statement or a CONNECT statement.

If you specify either OPTIONS(COMPOSITE) or OPTIONS(COMMAND) on the BIND PACKAGE COPY command, DB2 always uses DBPROTOCOL(DRDA) to create the target package, regardless of the DBPROTOCOL option that was used to create the source package that you specified in the COPY parameter.

If you specify an option on the BIND PACKAGE command, DB2 uses that remote access method for the package statements, regardless of the BIND PLAN option. For remote bind, the default is the system default at the remote site.

If you specify an option on the BIND PLAN statement, that information is stored in table SYSPLAN.

(DRDA)

DBPROTOCOL(DRDA) is passed on BIND PACKAGE, BIND PLAN, REBIND PACKAGE, or REBIND PLAN invocation.

Defaults:

Process	Default value
BIND PLAN	DRDA
BIND PACKAGE	DRDA
REBIND PLAN	DRDA
REBIND PACKAGE	DRDA

Catalog record: Column DBPROTOCOL of tables SYSPACKAGE and SYSPLAN.

DEFER and NODEFER bind options

The DEFER and NODEFER options determine whether to defer preparation for dynamic SQL statements that refer to remote objects, or to prepare them immediately.

NODEFER	NODEFER(PREPARE)	On: BIND and REBIND PLAN and PACKAGE
DEFER	DEFER(PREPARE)	
	DEFER(INHERITFROMPLAN)	Not valid for REBIND of a native SQL procedure package.

If you defer preparation, the dynamic statement is prepared when DB2 first encounters a statement of the type EXECUTE, OPEN, or DESCRIBE that refers to the dynamic statement.

For BIND and REBIND PACKAGE, if neither option is specified, and REOPT(NONE) applies:

- For local binds, the package inherits the plan's option at run time.
- For remote bind the default is NODEFER(PREPARE) at the remote DB2 server.

If you specify the bind option REOPT(ALWAYS), REOPT(AUTO), or REOPT(ONCE), DB2 sets the bind option DEFER(PREPARE) automatically.

You cannot use both DEFER and NODEFER.

NODEFER(PREPARE)

Does not defer preparation.

DEFER(PREPARE)

Defers preparation.

DEFER(INHERITFROMPLAN)

Enables a local package to inherit the value of the DEFER option from the plan, regardless of whether the package was bound remotely or locally.

If you bind a package remotely with the DEFER(INHERITFROMPLAN) option and the remote server does not understand the INHERITFROMPLAN value, the server might return an error.

The DEFER(INHERITFROMPLAN) value is not applied in the following situations, because no associated plan exists:

- If you bind the application locally and then copy the package to a remote server.
- If you bind an application that uses RRSAP.
- For any packages that are created for utilities

In these cases, NODEFER(PREPARE) is in effect for the package.

DEFER(PREPARE) and distributed processing: Specify the bind option DEFER(PREPARE) to improve performance, instead of NODEFER(PREPARE), and when binding dynamic SQL for DRDA access. DB2 does not prepare the dynamic SQL statement until that statement executes. (The exception to this situation is dynamic SELECT, which combines PREPARE and DESCRIBE, regardless of whether the DEFER(PREPARE) option is in effect.) When a dynamic SQL statement accesses remote data, the PREPARE and EXECUTE statements can be transmitted over the network together and processed at the remote location. Responses to both statements can be sent back to the local subsystem together. This reduces network traffic, which improves the performance of the dynamic SQL statement.

PREPARE statements that contain INTO clauses are not deferred.

To defer the preparation of an SQL statement in an application, bind or rebind the application with the option DEFER(PREPARE). This defers PREPARE messages for SQL statements that refer to a remote object until either:

- The statement executes
- The application requests a description of the results of the statement

If you choose to defer PREPARE statements, after the EXECUTE or DESCRIBE statement, you should code your application to handle any SQL error codes or SQLSTATES that the PREPARE statement might return. You can defer PREPARE statements only if you specify the bind option DEFER(PREPARE).

Defaults:

Process	Default value
BIND PLAN	NODEFER
BIND PACKAGE	Plan value
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column DEFERPREP of table SYSPLAN and column DEFERPREPARE of table SYSPACKAGE.

DEGREE bind option

The DEGREE option determines whether to attempt to run a query using parallel processing to maximize performance.

DEGREE	(1) (ANY)	On: BIND and REBIND PLAN and PACKAGE
		Not valid for REBIND of a native SQL procedure package.

For plans, DEGREE has no effect.

Limitations: If you bind plans or packages using DEGREE(ANY), the space required in the EDM pool could increase by 50% to 70%.

Defaults:

Process	Default value
BIND PLAN	1
BIND PACKAGE	1
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column DEGREE of tables SYSPACKAGE and SYSPLAN.

DEPLOY bind option

The DEPLOY option deploys a compiled SQL function or a native SQL procedure.

DEPLOY	(<i>collection-id.package-id</i>) COPYVER(<i>version-id</i>)	On: BIND PACKAGE
--------	---	------------------

collection-id

The name of the collection that contains the package to deploy, as listed in column COLLID of catalog table SYSPACKAGE.

package-id

The name of the package to deploy, as listed in column NAME of catalog table SYSPACKAGE.

COPYVER (*version-id*)

Determines the version of the package to deploy. The default for *version-id* is the empty string.

Specify BIND PACKAGE DEPLOY only when the target DB2 is a DB2 for z/OS server. You can use the ADMIN_COMMAND_DSN or DSNACCTS procedure to submit this command from a remote requester environment.

If you specify DEPLOY, the only other options that you can specify are PACKAGE, QUALIFIER, ACTION, OWNER, QUERYACCELERATION, and GETACCELARCHIVE. If you do not specify QUALIFIER or OWNER, their default values are the authorization ID that issues the BIND PACKAGE DEPLOY command.

Using DEPLOY with the ACTION option: If you specify ACTION(ADD) for a version that does not exist at the target location, DB2 creates or adds a new version of the SQL function or a native SQL procedure and its associated package while keeping the logic from the source object. DB2 adds a new version if an SQL function or a native SQL procedure with the same target name already exists.

If you specify ACTION(REPLACE), DB2 replaces the version specified in COPYVER. If you specify REPLVER, the version ID must be the same as the COPYVER version ID or DB2 returns TSO error message, DSNE977E.

Authorization:

To issue a BIND DEPLOY ACTION(ADD) command, the following authorization is necessary:

To issue a BIND DEPLOY ACTION(REPLACE) command, the following authorization is necessary:

Defaults:

Process	Default value
BIND PLAN	N/A
BIND PACKAGE	None
REBIND PLAN	N/A
REBIND PACKAGE	N/A

DEPLOY has **no default**. If you do not use DEPLOY, you must use MEMBER or COPY.

Catalog record: Column TYPE of table SYSPACKAGE

DISCONNECT bind option

The DISCONNECT option determines which remote connections to destroy during commit operations.

DISCONNECT	(EXPLICIT) (AUTOMATIC) (CONDITIONAL)	On: BIND and REBIND PLAN
------------	--	--------------------------

The option applies to any application process that uses the plan and has remote connections of any type. Regardless of the value of this option, a commit operation destroys all connections in the release pending state. You can put a connection in the release pending state using the SQL statement RELEASE.

(EXPLICIT)

Destroy only connections in the release pending state. This value allows you maximum flexibility for controlling remote connections.

(AUTOMATIC)

Destroy all remote connections.

(CONDITIONAL)

Destroy all remote connections unless an open cursor defined as WITH HOLD is associated with the connection.

Defaults:

Process	Default value
BIND PLAN	EXPLICIT
BIND PACKAGE	N/A
REBIND PLAN	Existing value
REBIND PACKAGE	N/A

Catalog record: Column DISCONNECT of table SYSPLAN.

DYNAMICRULES bind option

The DYNAMICRULES option determines the rules that apply at run time for certain dynamic SQL attributes.

DYNAMICRULES	(RUN) (BIND) (DEFINEBIND) (DEFINERUN) (INVOKEBIND) (INVOKERUN)	On: BIND and REBIND PLAN and PACKAGE Not valid for REBIND of a native SQL procedure package.
--------------	---	---

If DYNAMICRULES is specified for BIND PLAN, the DYNAMICRULES value has an effect only if any of the following conditions are true:

- The MEMBER option is specified. In that case, the generated package inherits the DYNAMICRULES value from the BIND PLAN command. If DYNAMICRULES is specified for REBIND PLAN, the DYNAMICRULES value has no effect.
- An existing package has no DYNAMICRULES value. If BIND PLAN with a DYNAMICRULES value is issued, and the package list includes a package without a DYNAMICRULES value, the package inherits the DYNAMICRULES value from the plan when the package runs.

If DYNAMICRULES is specified for REBIND PLAN, the DYNAMICRULES value has an effect only if an existing package has no DYNAMICRULES value. If REBIND PLAN with a DYNAMICRULES value is issued, and the package list includes the package without a DYNAMICRULES value, the package inherits the DYNAMICRULES value from the plan when the package runs.

The dynamic SQL attributes that DYNAMICRULES affects are:

- The authorization ID that is used to check authorization
- The qualifier that is used for unqualified objects
- The source for application programming options that DB2 uses to parse and semantically verify dynamic SQL statements
- Whether dynamic SQL statements can include GRANT, REVOKE, ALTER, CREATE, DROP, and RENAME statements

In addition to the DYNAMICRULES value, the run time environment of a package controls how dynamic SQL statements behave at run time. The two possible run time environments are:

- The package runs as part of a stand-alone program
- The package runs as a stored procedure or user-defined function package, or runs under a stored procedure or user-defined function

The combination of the DYNAMICRULES value and the run time environment determine the values for the dynamic SQL attributes. That set of attribute values is called the dynamic SQL statement *behavior*. The four behaviors are:

- Run behavior
- Bind behavior
- Define behavior
- Invoke behavior

The following DYNAMICRULES option descriptions include a description of the dynamic SQL statement behavior for each run time environment. This information is summarized in Table 10 on page 121.

- (RUN)** Processes dynamic SQL statements using the standard attribute values for dynamic SQL statements, which are collectively called *run behavior*:
- DB2 uses the authorization ID of the application process and the SQL authorization ID (the value of the CURRENT SQLID special register) for authorization checking of dynamic SQL statements.
 - DB2 uses the value of the CURRENT SCHEMA special register as the default schema of table, view, index, and alias names.
 - Dynamic SQL statements use the values of application programming options that were specified during installation. The installation option USE FOR DYNAMICRULES has no effect.

- GRANT, REVOKE, CREATE, ALTER, DROP, and RENAME statements can be executed dynamically.

(BIND)

Processes dynamic SQL statements using the following attribute values, which are collectively called *bind behavior*:

- DB2 uses the authorization ID of the package for authorization checking of dynamic SQL statements.
- Unqualified table, view, index, and alias names in dynamic SQL statements are implicitly qualified with value of the bind option QUALIFIER; if you do not specify QUALIFIER, DB2 uses the authorization ID of the package owner as the default schema.
- The attribute values that are described in Common attribute values for bind, define, and invoke behaviors.
- If you are running a trusted context with a role, or when the owner is a role, DB2 uses the authorization ID of the owner of the package for authorization checking of dynamic SQL statements. The same rules that are used to determine the authorization ID for static (embedded) statements are used for dynamic statements. DB2 uses the authorization ID of the owner of the package for authorization checking of dynamic SQL statements. The privilege set is the privileges that are held by the authorization ID of the owner of the package. The owner can also be a role. The identifier specified in the QUALIFIER option of the bind command that is used to bind the SQL statements is the implicit qualifier for all unqualified tables, views, aliases, indexes, and sequences. If this bind option was not used when the package was created or last rebound, the implicit qualifier is the authorization ID of the owner of the package. The owner can also be a role.

(DEFINEBIND)

Processes dynamic SQL statements using one of two behaviors, *define behavior* or *bind behavior*.

When the package is run as or runs under a stored procedure or user-defined function package, DB2 processes dynamic SQL statements using define behavior, which consists of the following attribute values:

- DB2 uses the authorization ID of the user-defined function or stored procedure owner for authorization checking of dynamic SQL statements in the application package.
- The default qualifier for unqualified objects is the user-defined function or stored procedure owner.
- The attribute values that are described in Common attribute values for bind, define, and invoke behaviors.
- If you are running a trusted context with a role, or when the owner is a role, then define behavior applies only if the dynamic SQL statement is in a package that is run as a stored procedure or user-defined function (or runs under a stored procedure or user-defined function package), and the package is bound with DYNAMICRULES (DEFINEBIND). DB2 uses the authorization ID of the stored procedure or user-defined function owner for authorization checking of dynamic SQL statements in the application package. This can be a primary or secondary authorization ID or a role. In this case, the privilege set is the privileges that are held by the authorization ID of the stored procedure or user-defined function owner. This owner can be a primary or secondary authorization ID or a role. The authorization ID of the stored procedure

or user-defined function owner is also the implicit qualifier for unqualified table, view, alias, index, and sequence names. The owner can also be a role.

When the package is run as a stand-alone program, DB2 processes dynamic SQL statements using bind behavior, which is described in BIND keyword.

(DEFINERUN)

Processes dynamic SQL statements using one of two behaviors, *define behavior* or *run behavior*.

When the package is run as or runs under a stored procedure or user-defined function package, dynamic SQL statements have define behavior, which is described in DEFINEBIND keyword.

When the package is run as a stand-alone program, DB2 processes dynamic SQL statements using run behavior, which is described in RUN keyword.

If you are running a trusted context with a role, or when the owner is a role, then define behavior applies only if the dynamic SQL statement is in a package that is run as a stored procedure or user-defined function (or runs under a stored procedure or user-defined function package), and the package is bound with DYNAMICRULES (DEFINERUN). DB2 uses the authorization ID of the stored procedure or user-defined function owner for authorization checking of dynamic SQL statements in the application package. This can be a primary or secondary authorization ID or a role. In this case, the privilege set is the privileges that are held by the authorization ID of the stored procedure or user-defined function owner. This owner can be a primary or secondary authorization ID or a role. The authorization ID of the stored procedure or user-defined function owner is also the implicit qualifier for unqualified table, view, alias, index, and sequence names. The owner can also be a role.

(INVOKEBIND)

Processes dynamic SQL statements using one of two behaviors, *invoke behavior* or *bind behavior*.

When the package is run as or runs under a stored procedure or user-defined function package, DB2 processes dynamic SQL statements using invoke behavior, which consists of the following attribute values:

- DB2 uses the authorization ID of the user-defined function or stored procedure invoker for authorization checking of dynamic SQL statements in the application package.

If the invoker is the primary authorization ID of the process or the CURRENT SQLID value, secondary authorization IDs are also checked if they are needed for the required authorization. Otherwise, only one ID, the ID of the invoker, is checked for the required authorization.

- The default qualifier for unqualified objects is the user-defined function or stored procedure invoker.
- The attribute values that are described in Common attribute values for bind, define, and invoke behaviors.
- If you are running a trusted context with a role, or when the owner is a role, then invoke behavior applies only if the dynamic SQL statement is in a package that is run as a stored procedure or user-defined function (or runs under a stored procedure or user-defined function package), and the package was bound with DYNAMICRULES (INVOKEBIND).

DB2 uses the authorization ID of the stored procedure or user-defined function invoker for authorization checking of dynamic SQL statements in the application package. The invoker can also be a role. The privilege set is the privileges that are held by the authorization ID of the stored procedure or user-defined function invoker. However, if the invoker is the primary authorization ID of the process or the CURRENT SQLID value, secondary authorization IDs are also checked along with the primary authorization ID's role. Therefore, the privilege set is the union of the set of privileges held by each authorization ID of the process and the primary authorization ID's role. The authorization ID of the stored procedure or user-defined function invoker is also the implicit qualifier for unqualified table, view, alias, index, and sequence names. The invoker can also be a role.

When the package is run as a stand-alone program, DB2 processes dynamic SQL statements using bind behavior, which is described in BIND keyword.

(INVOKERUN)

Processes dynamic SQL statements using one of two behaviors, *invoke behavior* or *run behavior*.

When the package is run as or runs under a stored procedure or user-defined function package, DB2 processes dynamic SQL statements using invoke behavior, which is described in INVOKEBIND keyword.

If you are running a trusted context with a role, or when the owner is a role, then invoke behavior applies only if the dynamic SQL statement is in a package that is run as a stored procedure or user-defined function (or runs under a stored procedure or user-defined function package), and the package was bound with DYNAMICRULES(INVOKERUN). DB2 uses the authorization ID of the stored procedure or user-defined function invoker for authorization checking of dynamic SQL statements in the application package. The invoker can also be a role. The privilege set is the privileges that are held by the authorization ID of the stored procedure or user-defined function invoker. However, if the invoker is the primary authorization ID of the process or the CURRENT SQLID value, secondary authorization IDs are also checked along with the primary authorization ID's role. Therefore, the privilege set is the union of the set of privileges held by each authorization ID of the process and the primary authorization ID's role. The authorization ID of the stored procedure or user-defined function invoker is also the implicit qualifier for unqualified table, view, alias, index, and sequence names. The invoker can also be a role.

When the package is run as a stand-alone program, DB2 processes dynamic SQL statements using run behavior, which is described in RUN keyword.

Common attribute values for bind, define, and invoke behavior: The following attribute values apply to dynamic SQL statements in packages that have bind, define, or invoke behavior:

- You can execute the statement SET CURRENT SQLID in a package that is bound with any DYNAMICRULES value. However, DB2 does not use the value of CURRENT SQLID as the authorization ID for dynamic SQL statements.

DB2 always uses the value of CURRENT SQLID as the qualifier for the EXPLAIN output and optimizer hints input PLAN_TABLE. (If the value of CURRENT SQLID has an alias on PLAN_TABLE and has the appropriate privileges, that PLAN_TABLE is populated.)

- If the value of installation option USE FOR DYNAMICRULES is YES, DB2 uses the application programming default values that were specified during installation to parse and semantically verify dynamic SQL statements. If the value of USE for DYNAMICRULES is NO, DB2 uses the precompiler options to parse and semantically verify dynamic SQL statements.
- GRANT, REVOKE, CREATE, ALTER, DROP, and RENAME statements cannot be executed dynamically.

Remote DB2 servers: For a package that uses DRDA access, DB2 sends the DYNAMICRULES option to the DB2 server at bind time.

The following table summarizes the dynamic SQL statement attribute values for each behavior.

Table 10. Definitions of dynamic SQL statement behaviors

Dynamic SQL attribute	Value for bind behavior	Value for run behavior	Value for define behavior	Value for invoke behavior
Authorization ID	Package OWNER	Current SQLID	User-defined function or stored procedure owner	Authorization ID of invoker
Default qualifier for unqualified objects	Bind OWNER or QUALIFIER value	Current SQLID	User-defined function or stored procedure owner	Authorization ID of invoker
CURRENT SQLID	Initialized to primary authid. SET SQLID is allowed.	Initialized to primary authid. SET SQLID is allowed.	Initialized to primary authid. SET SQLID is allowed.	Initialized to primary authid. SET SQLID is allowed.
Source for application programming options	As determined by the application defaults parameter DYNRULS	Application programming defaults installation panels	As determined by the application defaults parameter DYNRULS	As determined by the application defaults parameter DYNRULS
Can execute GRANT, REVOKE, CREATE, ALTER, DROP, RENAME?	No	Yes	No	No

Defaults:

Process	Default value
BIND PLAN	RUN
BIND PACKAGE	blank, or RUN for packages that run on a remote DB2 for z/OS server. ^{1, 2}
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Note:

1. When the package value is blank, the value that is used when the package is run is inherited from the plan value.
2. The default for a package on a remote DB2 for z/OS server is RUN. The default is set by the server.

Catalog record: Column DYNAMICRULES of tables SYSPACKAGE and SYSPLAN.

Related concepts:

 DYNAMICRULES bind option (DB2 Application programming and SQL)

ENABLE and DISABLE bind options

The ENABLE and DISABLE options determines which connections can use a plan or package.

<u>ENABLE</u>	(*)	On: BIND and REBIND
<u>DISABLE</u>	(BATCH)	PLAN and PACKAGE
	(CICS)	
	(CICS) CICS (<i>applid</i> , ...)	Not valid for REBIND of a
	(DB2CALL)	native SQL procedure
	(DLIBATCH)	package.
	(DLIBATCH) DLIBATCH (<i>connection-name</i> , ...)	
	(IMS)	
	(IMSBMP)	
	(IMSBMP) IMSBMP (<i>imsid</i> , ...)	
	(IMSMPP)	
	(IMSMPP) IMSMPP (<i>imsid</i> , ...)	
	(REMOTE) (BIND and REBIND PACKAGE only)	
	(RRSAF)	

You cannot use both DISABLE and ENABLE. For packages, DISABLE and ENABLE are valid only for local bind operations.

ENABLE

Lists the system connection types that can use the plan or package. Connection types not listed cannot use it.

DISABLE

Lists the system connection types that cannot use the plan or package. Connection types not listed can use it.

With some connection types you can list connection IDs to identify specific connections of the type to disable or enable.

If you list connection IDs as disabled, any connections not listed for the same connection type are enabled.

If you list connection IDs as enabled, any connections not listed for the same connection type are disabled.

A connection ID is valid only after the keyword that names its corresponding connection type.

Connection types:

(*)

Specifies all valid connection types. Use only with ENABLE.

(BATCH)

Indicates that all TSO connections are either enabled or disabled for the plan or package.

(CICS)

Identifies the CICS Connection. All CICS VTAM node names specified in the CICS SIT table are either enabled or disabled for the plan or package.

(**CICS**) **CICS** (*applid* , ...)

Identifies the CICS VTAM node name specified in the APPLID parameter of the CICS SIT table. The CICS VTAM node identified by *applid* is either enabled or disabled for the plan or package.

(**DB2CALL**)

Indicates that the call attachment facility (CAF) connection is either enabled or disabled for the plan or package.

(**DLIBATCH**)

Identifies the Data Language I (DL/I) Batch Support Facility connection. All connection identifiers from the DDITV02 data set or the job name in the JCL that the DL/I batch support system needs to have are either enabled or disabled for the plan or package.

(**DLIBATCH**) **DLIBATCH** (*connection-name* , ...)

Specifies the connection identifier as from the DDITV02 data set or the job name in the JCL that the DL/I batch support system needs to have. The DL/I batch connection identified by *connection-name* is either enabled or disabled for the plan or package.

(**IMS**)

Specifies that all Information Management System (IMS) connections, DLIBATCH, IMSBMP, and IMSMPP are either enabled or disabled for the plan or package.

(**IMSBMP**)

Specifies the IMS connection for the Batch Message Program (BMP) region. All IMS BMP connections identified by the value of IMSID on the CTL parameter EXEC are either enabled or disabled for the plan or package.

(**IMSBMP**) **IMSBMP** (*imsid* , ...)

Specifies the value of IMSID on the CTL parameter EXEC. The IMS BMP connection identified by *imsid* is either enabled or disabled for the plan or package.

(**IMSMPP**)

Specifies the IMS connection for the Message Processing Program (MPP) and IMS Fast Path (IFP) regions. All IMS MPP connections identified by the value of the IMSID on the CTL parameter EXEC. are either enabled or disabled for the plan or package.

(**IMSMPP**) **IMSMPP** (*imsid* , ...)

Specifies the value of IMSID on the CTL parameter EXEC. The IMS MPP connection identified by *imsid* is either enabled or disabled for the plan or package.

(**REMOTE**)

Indicates that all remote connections are either enabled or disabled for the package.

(**RRSAF**)

Indicates that the RRS attachment facility connection is either enabled or disabled for the plan or package.

Plans that disable a system: If a plan disables a system, then no packages appended to that plan can run from that system, regardless of the ENABLE/DISABLE options. However, if the same packages are appended to other plans that enable the system, those packages can run from that system under those plans.

Interactions with the PLANMGMT option: If you plan to change either of these options and the PLANMGMT option in a REBIND command, see “PLANMGMT bind option” on page 144 for information about the implications.

Defaults:

Process	Default value
BIND PLAN	ENABLE(*)
BIND PACKAGE	ENABLE(*)
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Table SYSPKSYSTEM for packages and table SYSPLSYSTEM for plans.

ENCODING bind option

The ENCODING option specifies the application encoding for all host variables in static statements in a plan or package.

ENCODING	(ASCII)(EBCDIC)(UNICODE)(ccsid)	On: BIND and REBIND PLAN and PACKAGE
		Not valid for REBIND of a native SQL procedure package.

EBCDIC is the only valid option for a plan or package that was precompiled prior to DB2 Version 7. If you specify *ccsid* on any plan or package precompiled prior to DB2 Version 7, the value of *ccsid* must match the EBCDIC CCSID specified on the installation panel DSNTIPF (the SYSTEM EBCDIC CCSID). You can specify ASCII, UNICODE, or *ccsid*, where *ccsid* is a value other than the SYSTEM EBCDIC CCSID for any plan or package precompiled on DB2 Version 7 or later. You might select this option when a data source, such as a terminal emulator, uses a CCSID that is not the same as the SYSTEM EBCDIC CCSID. For example, a user has a terminal emulator with a CCSID of 1047, but the SYSTEM EBCDIC CCSID is 37. In this case, the plan or package being used by that user should be bound with ENCODING (1047).

ENCODING also affects the content of the data that is returned by the SQL statement DESCRIBE. DB2 will return column names, label names, or both (if requested) in the specified application encoding scheme.

Defaults: The default package application encoding scheme is not inherited from the plan application encoding option. The default for a package that is bound on a remote DB2 for z/OS system is the remote server's default application encoding scheme. Similarly, when a plan or package is run on a remote DB2 for z/OS server, the specified ENCODING option is ignored. Instead, the remote server's encoding scheme is used.

The following statements set the value of the host variable and do not require the package to be bound into the plan:

```
SET CURRENT PACKAGE SET = :HV ,
SET :HV = CURRENT PACKAGE SET ,
SET :HV = CURRENT PACKAGE PATH ,
SET CURRENT PACKAGE PATH = :HV
```

The host variable uses the system default application encoding scheme, even when the application is bound with the ENCODING (EBCDIC/UNICODE) bind option.

Defaults:

Process	Default value
BIND PLAN	The system default application encoding scheme that was specified at installation time
BIND PACKAGE	The system default application encoding scheme that was specified at installation time
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column ENCODING_CCSID of table SYSPLAN or SYSPACKAGE. The value is set as follows:

- For a MIXED=NO subsystem, if ENCODING(ASCII) or ENCODING(EBCDIC) is specified, the SBCS CCSID of the encoding scheme is stored in the catalog.
- For a MIXED=YES subsystem, if ENCODING(ASCII) or ENCODING(EBCDIC) is specified, the mixed CCSID of the encoding scheme is stored in the catalog.
- If ENCODING(UNICODE) is specified, the mixed CCSID (1208) is stored in the catalog, regardless of the MIXED setting.

EXPLAIN bind option

The EXPLAIN option causes the bind process to obtain information about how SQL statements in the package or packages are to execute.



EXPLAIN	(<u>NO</u>) (YES) (ONLY)	On BIND and REBIND PLAN and PACKAGE, REBIND TRIGGER PACKAGE
---------	--------------------------------	---

The bind process inserts that information into the table *owner*.PLAN_TABLE. *Owner* can be the authorization ID of the owner of the plan or package. Alternatively, the authorization ID of the owner of the plan or package can have an alias as *owner*.PLAN_TABLE that points to the base table, PLAN_TABLE. *Owner* must also have the appropriate SELECT and INSERT privileges on that table. This option does not obtain information for statements that access remote objects.

PLAN_TABLE must have a base table and can have multiple aliases with the same table name, PLAN_TABLE, but using different authorization IDs; it cannot be a view or a synonym. It should exist before the bind process begins.

The EXPLAIN option also populates all of the explain tables, except for the following:

- DSN_QUERY_TABLE
- DSN_STATEMENT_CACHE_TABLE

You can get EXPLAIN output for a statement that is embedded in a program that is bound with EXPLAIN(NO) by embedding the SQL statement EXPLAIN in the

program. Otherwise, the value of the EXPLAIN option applies to all explainable SQL statements in the program, and to the fullselect portion of any DECLARE CURSOR statements.

In all inserts to *owner* .PLAN_TABLE, the value of QUERYNO is the statement number that the precompiler assigned and placed in the DBRM.

For automatic rebind: EXPLAIN(YES) is in effect if you bind the plan or package with EXPLAIN(YES) and if the value of field EXPLAIN PROCESSING on installation panel DSNTIPO is YES. If EXPLAIN(YES) and VALIDATE(BIND) are in effect and PLAN_TABLE is not correct, the automatic rebind fails.

(**NO**) Provides no EXPLAIN information.

(**YES**)

Inserts information in the tables populated by EXPLAIN. No secondary authorization checks for the owner authorization ID are performed during the bind operation. You must grant all privileges and authorities directly to the owner authorization ID. If *owner* .PLAN_TABLE does not exist at bind time, the value of the option VALIDATE determines the success of the bind operation.

- If the value is BIND, the bind fails.
- If the value is RUN, DB2 checks to see if the table exists again at run time. If it still does not exist, the plan or package cannot run. If it does exist, DB2 inserts information in PLAN_TABLE before the plan or package runs.

If neither or both of the optional tables DSN_FUNCTION_TABLE or DSN_STATEMNT_TABLE exist, or if they are defined incorrectly, the bind does not fail.

(**ONLY**)

Populates EXPLAIN tables but does not create or rebind a package. This option allows EXPLAIN to be run when the authorization ID of the bind or rebind process does not have the privilege to execute statements in the package. If the command that specifies EXPLAIN(ONLY) refers to a package and version that already exists, the existing package is not dropped or replaced, even if ACTION(REPLACE) is specified. If an error is encountered during population of the EXPLAIN tables, information is not added to the EXPLAIN tables for the statement where the error occurred, and for any subsequent statements.

The EXPLAIN(ONLY) option is also useful for testing whether access paths can be successfully reused before you bind or rebind a package with the APREUSE(ERROR) or APCOMPARE(ERROR) bind options. When EXPLAIN(ONLY) is specified with those options, the PLAN_TABLE is populated with information about all statements in a package, even when reuse or comparison fails for the access path for one or more statements. You can examine the HINT_USED and REMARKS columns to discover which statements in the package that cannot be reused.

Important: Do not rely upon the PLAN_TABLE data for a statement when access path reuse fails. The access path described by the data does not match the previous access path for the statement or the access path that DB2 selects when reuse is not specified. The data is useful only for identifying the statements that cannot reuse the previous access path.

A 'Y' value in the BIND_EXPLAIN_ONLY column indicates PLAN_TABLE rows that are created under this option.

Invalidation resulting from an unsuccessful rebind: An unsuccessful rebind generating a return code of greater than 4 invalidates the rebind object and rolls back all changes to the object, leaving it as it was before the rebind attempt. However, if the rebind fails because of either the REBIND option EXPLAIN or the SQL statement EXPLAIN (that is, the PLAN_TABLE does not exist or was created incorrectly), DB2 rolls back all changes to the object but does not invalidate the object.

Defaults:

Process	Default value
BIND PLAN	NO
BIND PACKAGE	NO
REBIND PLAN	NO
REBIND PACKAGE	Existing value

Catalog record: Column EXPLAIN of table SYSPACKAGE and column EXPLAIN of SYSPLAN.



Related concepts:

- ➡ Investigating SQL performance by using EXPLAIN (DB2 Performance)
- ➡ Interpreting data access by using EXPLAIN (DB2 Performance)

Related tasks:

- ➡ Capturing access path information in EXPLAIN tables (DB2 Performance)

Related reference:

- ➡ EXPLAIN (DB2 SQL)
- ➡ EXPLAIN tables (DB2 Performance)
- ➡ SYSIBM.SYSPACKAGE table (DB2 SQL)
- ➡ SYSIBM.SYSPLAN table (DB2 SQL)

EXTENDEDINDICATOR bind option

The EXTENDEDINDICATOR option determines whether DB2 recognizes extended indicator variables when the associated package is run.

EXTENDEDINDICATOR

EXTENDEDINDICATOR	(<u>NO</u>) (YES)	On: BIND and REBIND PACKAGE
		Not valid for REBIND of a native SQL procedure package.

(NO) Specifies that DB2 does not recognize extended indicator variable values.

Indicator variables are normal indicator variables; negative indicator variable values are null, and positive or zero indicator variable values are non-null.

(YES)

Specifies that DB2 recognizes extended indicator variable values. If you use any non-recognized indicator variable values or default or unassigned indicator variable-based values in a non-supported location, DB2 generates an error message when the bound statement is running.

Defaults:

Process	Default value
BIND PLAN	N/A
BIND PACKAGE	NO
REBIND PLAN	N/A
REBIND PACKAGE	Existing value

Catalog record: Column EXTENDEDINDICATOR of table SYSPACKAGE.

FILTER bind option

The FILTER option lets you delete a set of queries in the SYSIBM.SYSQUERY table under a {tag} value that is specified by the SYSQUERY.USERFILTER column.

FILTER	('<u>filter-name</u>')	On: FREE QUERY
---------------	---------------------------------	-----------------------

('filter-name')

Related reference:

FREE QUERY (DSN)

 [SYSIBM.SYSQUERY table \(DB2 SQL\)](#)

FLAG bind option

The FLAG option determines the messages to display during the bind process.

FLAG	(<u>I</u>)(W)(E)(C)	On: BIND and REBIND PLAN and PACKAGE, REBIND TRIGGER PACKAGE
-------------	------------------------------------	---

Not valid for REBIND of a native SQL procedure package.

- (I) All informational, warning, error, and completion messages
- (W) Only warning, error, and completion messages
- (E) Only error and completion messages
- (C) Only completion messages

Rebinding multiple plans or packages: When your REBIND command contains an asterisk (*) and affects many plans or packages, FLAG(E) is recommended to avoid running out of message storage.

Defaults:

Process	Default value
BIND PLAN	I
BIND PACKAGE	I
REBIND PLAN	I
REBIND PACKAGE	I
REBIND TRIGGER PACKAGE	I

GENERIC bind option

The **GENERIC** option specifies one or more bind options that are supported by the target server, but are not supported as options for **BIND PACKAGE** or **REBIND PACKAGE** on the DB2 for z/OS subsystem on which the **BIND** or **REBIND** command is issued.

GENERIC	(' <i>string</i> ')	On: BIND and REBIND PACKAGE
		Not valid for REBIND of a native SQL procedure package.

Do not use **GENERIC** to specify any of the following options:

- Bind options that are explicitly supported by DB2 for z/OS for **BIND PACKAGE** or **REBIND PACKAGE**
- SQL processing options that are explicitly supported by DB2 for z/OS

('*string*')

One or more pairs of a bind option and its corresponding value. Separate each item by one or more blank spaces.

For example, the following **GENERIC** option specifies several bind options and their corresponding values:

```
GENERIC( ' option-1 value-1 option-2 value-2..... option-n value-n ' )
```

In this example, *value-1* is specified for bind option *option-1*, *value-2* is specified for bind option *option-2*, and so on.

If you specify a bind option more than once in the **GENERIC** string, only the first occurrence is used. All subsequent occurrences of the bind option and its corresponding value are ignored. For example, suppose that you specify the following **GENERIC** option:

```
GENERIC( ' firstoption firstvalue secondoption secondvalue firstoption thirdvalue ' )
```

In this example, *firstoption* and *secondoption* are bind options that are sent to the target server with their corresponding values. The third option and value pair (*firstoption thirdvalue*) is ignored, because *firstoption* is already specified in the string. This pair is not sent to the target server.

The maximum length of the string is 4096 bytes. Within that string, each individual option name cannot exceed 255 characters, and each individual option value cannot exceed 255 characters.

GETACCELARCHIVE bind option

The GETACCELARCHIVE bind option specifies whether a static SQL query that is bound for acceleration retrieves archived data on the accelerator, instead of active data.

When a package that is bound with this bind option runs, the GETACCELARCHIVE value, instead of the GET_ACCEL_ARCHIVE subsystem parameter, provides the initial value for the CURRENT_GET_ACCEL_ARCHIVE special register. The CURRENT_GET_ACCEL_ARCHIVE special register specifies whether a query that references a table that is archived on an accelerator server uses the archived data. Therefore, by rebinding the application package with the GETACCELARCHIVE bind option, you can also specify that accelerated dynamic SQL queries in an application retrieve only archived data on the accelerator. You can specify this bind option instead of either modifying the application to add an explicit SET CURRENT_GET_ACCEL_ARCHIVE statement or setting the GET_ACCEL_ARCHIVE subsystem parameter. For dynamic SQL queries in a package, the bind option value that is specified overrides the GET_ACCEL_ARCHIVE subsystem parameter. This bind option does not have a default value, so if you do not specify this bind option, the GET_ACCEL_ARCHIVE subsystem parameter initializes the special register.

GETACCELARCHIVE	(NO) (YES)	On: BIND and REBIND PACKAGE
		Not valid for REBIND of a native SQL procedure package.

(NO)

Specifies that no static SQL query is bound to retrieve archived data from the accelerator. If the static query also is not bound for acceleration, the query is bound to run in DB2.

If the static query is bound for acceleration because the QUERYACCELERATION bind option was specified, the query is routed to the accelerator when the application runs; however, the query does not retrieve any archived data.

(YES)

Specifies that if all of the following criteria are met, the query is bound for acceleration and retrieves the archived data on the accelerator when the application runs:

- The QUERYACCELERATION bind option is also specified.
- The static SQL query references an accelerated table that has partitioned data archived on an accelerator.
- The static query satisfies the acceleration criteria that is specified by the QUERYACCELERATION bind option.

If the static query does not satisfy the acceleration criteria that is specified by the QUERYACCELERATION bind option, the BIND or REBIND PACKAGE operation fails with an error message for that query.

Defaults:

Process	Default value
BIND PACKAGE	None
REBIND PACKAGE	None

There is no default value for this bind option. The GETACCELARCHIVE bind option does not inherit the setting from the GET_ACCEL_ARCHIVE subsystem parameter.

IMMEDWRITE bind option

The IMMEDIATEWRITE option indicates whether immediate writes are to be done for updates that are made to group buffer pool dependent page sets or partitions.

IMMEDWRITE	(NO)	On: BIND and REBIND PLAN and PACKAGE, and REBIND TRIGGER PACKAGE
	(YES)	
	(INHERITFROMPLAN) (BIND PACKAGE and REBIND PACKAGE only)	Not valid for REBIND of a native SQL procedure package.

This option is only applicable for data sharing environments. The IMMEDIATEWRITE subsystem parameter has no effect on the IMMEDIATEWRITE bind option at bind time. The following table shows the implied hierarchy of this option as it affects run time. The IMMEDIATEWRITE option values are as follows:

(NO) Specifies that normal write activity is done. Updated pages that are group buffer pool dependent are written at or before phase one of commit or at the end of abort for transactions that have rolled back.

(YES)

Specifies that updated pages that are group buffer pool dependent are immediately written as soon as the buffer update completes. Updated pages are written immediately even if the buffer is updated during forward progress or during rollback of a transaction. Specifying this option might impact performance.

(INHERITFROMPLAN)

Enables a local package to inherit the value of the IMMEDIATEWRITE option from the plan, regardless of whether the package was bound remotely or locally.

If you bind a package remotely with the IMMEDIATEWRITE(INHERITFROMPLAN) option and the remote server does not understand the INHERITFROMPLAN value, the server might return an error.

The IMMEDIATEWRITE(INHERITFROMPLAN) value is not applied in the following situations, because no associated plan exists:

- If you bind the application locally and then copy the package to a remote server.
- If you bind an application that uses RRSAA.
- For any packages that are created for utilities

In these cases, IMMEDIATEWRITE(NO) is in effect for the package.

Table 11. The implied hierarchy of the IMMEDIATEWRITE option

IMMEDWRITE bind option	IMMEDWRI subsystem parameter	Value at run time
NO	NO	NO

Table 11. The implied hierarchy of the IMMEDIATE option (continued)

IMMEDIATE bind option	IMMEDIATE subsystem parameter	Value at run time
NO	PH1	PH1
NO	YES	YES
PH1	NO	PH1
PH1	PH1	PH1
PH1	YES	YES
YES	NO	YES
YES	PH1	YES
YES	YES	YES

Note: The NO and PH1 options are equivalent. The PH1 option is shown for backward compatibility only.

Interactions with the PLANMGMT option: If you plan to change this option and the PLANMGMT option in a REBIND command, see “PLANMGMT bind option” on page 144 for the implications.

Performance implications: You can use IMMEDIATE(NO) and IMMEDIATE(YES) for situations where a transaction spawns another transaction that can run on another DB2 member and that depends on uncommitted updates that were made by the originating transaction.

Specify IMMEDIATE(NO) to cause group buffer pool dependent pages to be written at or before phase 1 of commit.

Specify IMMEDIATE(YES) to cause the originating transaction to immediately write its updated GBP-dependent buffers (instead of waiting until the end of commit or rollback), which will ensure that the dependent transaction always gets the same results regardless of whether it runs on the same member or a different member as the originating transaction. IMMEDIATE(YES) should be used with caution because of its potential impact to performance. The impact will be more significant for plans and packages that do many buffer updates to GBP-dependent pages, and not as noticeable for plans or packages that perform few buffer updates to GBP-dependent pages. The following options can be considered as alternatives to using IMMEDIATE(YES):

- Always run the dependent transaction on the same DB2 member as the originating transaction.
- Run the dependent transaction with ISOLATION(RR).
- Wait until the completion of phase two of commit before spawning the dependent transaction.
- CURRENTDATA(YES) or ISOLATION(RS) can be used to solve the problem only if the originating transaction updates columns that are not in the WHERE clause of the dependent transaction.

Defaults:

Process	Default value
BIND PLAN	NO

Process	Default value
BIND PACKAGE	<ul style="list-style-type: none"> • For a local server: INHERITFROMPLAN • For a remote server: NO
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value
REBIND TRIGGER PACKAGE	Existing value

ISOLATION bind option

The ISOLATION option determines how far to isolate an application from the effects of other running applications.

ISOLATION	(<u>CS</u>)(RS)(RR)(UR)(NC)	On: BIND and REBIND PLAN and PACKAGE, REBIND TRIGGER PACKAGE
		Not valid for REBIND of a native SQL procedure package.

(CS) *Cursor stability*. Ensures, like repeatable read, that your application does not read a row that another process changes until that process releases that row. Unlike repeatable read, cursor stability does not prevent other applications from changing rows that your application reads before your program commits or terminates.

(RR) *Repeatable read*. Ensures that:

- Your application does not read a row that another process has changed until that process releases that row.
- Other processes do not change a row that your application reads until your application commits or terminates.

(RS) *Read stability*. Ensures that:

- Your application does not read a row that another process has changed until that process releases that row.
- Other processes do not change a row that satisfies the application's search condition until your application commits or terminates. It does allow other application processes to insert a row, or to change a row that did not originally satisfy the search condition.

If the server does not support RS, it uses RR.

(UR) *Uncommitted read*. Unlike repeatable read and cursor stability, does not ensure anything. With the exception of LOB data, uncommitted read avoids acquiring locks on data and allows:

- Other processes change any row your application reads during the unit of work.
- Your application read any row that another process has changed, even if the process has not committed the row.

You can use this option only with a read-only operation: SELECT, SELECT INTO, or FETCH using a read-only cursor. If you specify ISOLATION(UR) for any other operation, DB2 uses ISOLATION(CS) for that operation.

(NC) *No commit*. Used on packages that are bound to certain servers other than

DB2 for z/OS. DB2 for z/OS does not support NC. If the server does not support this isolation level, it uses UR.

Defaults:

Process	Default value
BIND PLAN	CS
BIND PACKAGE	<ul style="list-style-type: none"> • For a local server: The plan value • For a remote server: CS
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value. You cannot change ISOLATION from a specified value to a default of the plan value by using REBIND PACKAGE. To do that, you must use BIND PACKAGE ACTION(REPLACE).

Catalog record: Column ISOLATION of tables SYSPACKAGE and SYSPLAN.

Related tasks:

 [Choosing an ISOLATION option \(DB2 Performance\)](#)

KEEPDYNAMIC bind option

The KEEPDYNAMIC option determines whether DB2 keeps dynamic SQL statements after commit points. .

KEEPDYNAMIC	(<u>NO</u>)(YES)	On: BIND and REBIND PLAN and PACKAGE
		Not valid for REBIND of a native SQL procedure package.

(NO) Specifies that DB2 does not keep dynamic SQL statements after commit points.

(YES) Specifies that DB2 keeps dynamic SQL statements after commit points.

If you specify KEEPDYNAMIC(YES), the application does not need to prepare an SQL statement after every commit point, DB2 keeps the dynamic SQL statement until one of the following events occurs:

- The application process ends.
- A rollback operation occurs.
- The application executes an explicit PREPARE statement with the same statement identifier.

If you specify KEEPDYNAMIC(YES), and the prepared statement cache is active, DB2 keeps a copy of the prepared statement in the cache. If the prepared statement cache is not active, DB2 keeps only the SQL statement string past a commit point. DB2 then implicitly prepares the SQL statement if the application executes an OPEN, EXECUTE, or DESCRIBE operation for that statement.

If you specify KEEPDYNAMIC(YES), DDF server threads that are used to execute KEEPDYNAMIC(YES) packages will remain active. Active DDF server threads are subject to idle thread timeouts.

If you specify KEEP_DYNAMIC(YES), you must not specify REOPT(ALWAYS). KEEP_DYNAMIC(YES) and REOPT(ALWAYS) are mutually exclusive. However, you can use KEEP_DYNAMIC(YES) with REOPT(ONCE).


Performance hint: KEEP_DYNAMIC(YES) results in improved performance if your DRDA client application uses a cursor defined WITH HOLD. DB2 automatically closes a held cursor when there are no more rows to retrieve, which eliminates an extra network message.

Defaults:


Process	Default value
BIND PLAN	NO
BIND PACKAGE	NO
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column KEEP_DYNAMIC of table SYSPLAN and SYSPACKAGE.


Related concepts:

 Methods for keeping prepared statements after commit points (DB2 Performance)

Related tasks:

 Improving dynamic SQL performance (DB2 Performance)

Related reference:

 MAX KEPT DYN STMTS field (MAXKEEPD subsystem parameter) (DB2 Installation and Migration)

RELEASE bind option

LIBRARY bind option

The LIBRARY option determines what partitioned data sets (libraries) to search for the DBRMs listed in the MEMBER option.

LIBRARY	(<i>dbrm-pds-name</i>) (<i>dbrm-pds-name</i> , ...) (BIND PLAN only)	On: BIND PLAN, BIND PACKAGE
----------------	---	------------------------------------

The libraries must be cataloged.

The bind process searches for the libraries in the order that you list them. If the libraries do not contain some DBRM listed in the MEMBER option, and if a JCL statement exists for DBRMLIB DD, then the process searches for the member among the libraries that the JCL statement describes.

dbrm-pds-name is the data set name of a library.


For BIND PACKAGE, you can specify only one library to search.

For BIND PLAN, you can specify one or more libraries to search.


Defaults:

Process	Default value
BIND PLAN	Search only the libraries described by the DBRMLIB DD statement
BIND PACKAGE	Search only the libraries described by the DBRMLIB DD statement
REBIND PLAN	N/A
REBIND PACKAGE	N/A

Related concepts:

 Bind options for locks (DB2 Performance)

Related tasks:

 BIND options for distributed applications (DB2 Performance)

Related reference:

 CURRENT OPTIMIZATION HINT (DB2 SQL)

 Bind options for remote access (DB2 Application programming and SQL)

MEMBER bind option

The MEMBER option determines what database request modules (DBRMs) to include in the package.

MEMBER	(<i>dbrm-member-name</i>)	On: BIND PACKAGE
---------------	-----------------------------	-------------------------

The MEMBER option is deprecated for BIND PLAN. Use it only when you cannot run BIND PACKAGE to bind DBRMs into packages explicitly. When you specify MEMBER, DB2 binds the DBRMs into packages, and binds the packages to the specified plan.

dbrm-member-name

Specifies the name of a library member that contains a DBRM. You can name the partitioned data set, of which the DBRM is a member, either in the LIBRARY option or in the JCL statement for DBRMLIB DD.

Names beginning with DSN are reserved; you receive a warning message if *dbrm-member-name* begins with DSN.

For BIND PACKAGE, *dbrm-member-name* becomes the package name.

For BIND PLAN, DB2 first binds each member into a package with name *dbrm-member-name* and collection ID DSN_DEFAULT_COLLID_ *plan-name*. Then DB2 binds those packages into the specified plan. You receive a warning that indicates that DBRMs are first being bound into packages. Subsystem parameter DISALLOW_DEFAULT_COLLID must be set to NO for BIND PLAN to be successful.

For **BIND PACKAGE**, you can use only one member. If you do not use MEMBER, you must use COPY. You cannot use both options.

For **BIND PLAN**, you can list many members. DB2 sorts the member list in alphabetical order. If you do not use MEMBER, you must use PKLIST.

Defaults:

Process	Default value
BIND PLAN	None
BIND PACKAGE	None

Process	Default value
REBIND PLAN	N/A
REBIND PACKAGE	N/A

Catalog record: Column NAME of table SYSPACKAGE.

OPTHINT bind option

The OPTHINT option controls whether access paths that are specified in PLAN_TABLE instances are used for static SQL statements.

OPTHINT	(' <i>hint-id</i> ')	On: BIND and REBIND PLAN and PACKAGE
		Not valid for REBIND of a native SQL procedure package.

The OPTHINT option also sets the default value for special register CURRENT OPTIMIZATION HINT, which provides optimization hints for dynamic SQL.

(' *hint-id*')

A character string of up to 128 characters in length. The value is used during query optimization to identify PLAN_TABLE rows that specify the use of a specific access path. The delimiters can only be single quotation marks (').

If ' *hint-id* ' contains all blank characters, DB2 does not use optimization hints for static SQL statements.

DB2 uses optimization hints only when optimization hints are enabled for your system. To enable optimization hints, specify YES for the value of the OPTHINTS subsystem parameter.

Restriction: The PACKAGE does not inherit from the PLAN.

Defaults:


Process	Default value
BIND PLAN	All blanks, use normal optimization
BIND PACKAGE	All blanks, use normal optimization
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column OPTHINT of tables SYSPLAN and SYSPACKAGE.

Related tasks:

 Specifying access paths in a PLAN_TABLE instance (DB2 Performance)

Related reference:

 OPTIMIZATION HINTS field (OPTHINTS subsystem parameter) (DB2 Installation and Migration)

 CURRENT OPTIMIZATION HINT (DB2 SQL)

 SYSIBM.SYSPACKAGE table (DB2 SQL)

 SYSIBM.SYSPLAN table (DB2 SQL)

OPTIONS bind option

The OPTIONS option specifies which bind options are used for the new package when the COPY option is specified.

OPTIONS	<u>(COMPOSITE)</u> (COMMAND)	On: BIND PACKAGE COPY
---------	---------------------------------	-----------------------

COMPOSITE

The option values that are specified in the BIND PACKAGE COPY subcommand are used for the package copy. The values for the options that are not specified are the option values that are in the SYSPACKAGE catalog table row that describes the source package that is to be copied.

For a remote copy, OPTIONS(COMPOSITE) is only valid if the remote DB2 subsystem is DB2 Version 8 or higher.

COMMAND

The option values that are specified in the BIND PACKAGE COPY subcommand are used for the package copy. The values for options that are not specified are determined as follows:

- For a local copy, the DB2-defined BIND PACKAGE options defaults are used.
- For a remote copy, the server-defined BIND PACKAGE option defaults are used at the server. You must use OPTIONS(COMMAND) when copying to a down-level server or to a non-z/OS DB2 server. A down-level server is any server that is not DB2 10 for z/OS.

Defaults:

Process	Default value
BIND PLAN	N/A
BIND PACKAGE COPY	COMPOSITE
REBIND PLAN	N/A
REBIND PACKAGE	N/A

OWNER bind option

The OWNER option determines the authorization ID of the owner of a plan or package.

OWNER	(<i>authorization-id</i>)	On: BIND and REBIND PLAN and PACKAGE
		Not valid for REBIND of a native SQL procedure package.

The owner must have the privileges required to execute the SQL statements contained in the object.

If ownership changes, all grants for privileges on the object that the previous owner issued change to name the new owner as the grantor. The new owner has the privileges BIND and EXECUTE on the object and grants them to the previous owner.

You can bind or rebind only the objects for which the authorization ID has bind privileges. If you do not specify an authorization ID, the process rebinds only the objects for which the primary ID has bind privileges.

Interactions with the PLANMGMT option: If you plan to change this option and the PLANMGMT option in a REBIND command, see “PLANMGMT bind option” on page 144 for the implications.

OWNER for BIND and REBIND in trusted context: When BIND and REBIND commands are issued in a trusted context that has the ROLE AS OBJECT OWNER clause, the owner is determined as follows:

- If the OWNER option is not specified, the role associated with the binder becomes the owner.
- If the OWNER option is specified, the role specified in the OWNER option becomes the owner. In a trusted context, the OWNER specified must be a role. For the bind to succeed, the binder needs BINDAGENT privilege from the role specified in the OWNER option. The binder also receives BINDAGENT privilege, if the role associated with the binder has BINDAGENT privilege.

If the ROLE AS OBJECT OWNER clause is not in effect for the trusted context, then the current rules for BIND and REBIND ownership apply. If a role is associated in a trusted context, then the role privileges are included in the binder's privilege set to determine if the binder is allowed to perform the bind.

For remote BIND or REBIND PACKAGE only, the value of OWNER is subject to translation when sent to the remote system.

Defaults:

Process	Default value
BIND PLAN	Primary authorization ID of the agent that runs the bind process
BIND PACKAGE	Primary authorization ID of the agent that runs the bind process
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column OWNER of table SYSPACKAGE, column GRANTOR of table SYSPACKAUTH, and column CREATOR of table SYSPLAN.

PACKAGE bind option

The PACKAGE option determines the packages to bind or rebind.

PACKAGE	<i>(location-name.collection-id)</i> (BIND PACKAGE)	On: BIND and REBIND PACKAGE
	<i>(location-name.collection-id.package-id.(version-id))</i> (REBIND PACKAGE)	

You cannot use the BIND PACKAGE subcommand to:

- Bind a package with the same name as an existing trigger package
- Copy a trigger package

The following options identify the location, collection, package name, and version of the package. You can identify a location and collection. For BIND, the DBRM

supplies the package ID and version ID if you use the option MEMBER, or those IDs come from the option COPY. For REBIND, you must identify a package name, and you can also supply a version ID.

location-name

The location of the DBMS where the package binds or rebinds and where the description of the package resides. The location name must be defined in catalog table SYSIBM.LOCATIONS. If that table does not exist or if the DBMS is not in it, you receive an error message.

The default is the local DBMS.

collection-id or *

Specifies the collection to contain the package to bind, or that already contains the package to rebind. There is no default.

For REBIND, you can use an asterisk (*) to rebind all local packages with the specified *package-id* in all the collections for which you have bind privileges.

For REBIND PACKAGE, *collection-id* can be an undelimited or a delimited identifier. The delimiter for *collection-id* is double quotation marks ("). If *collection-id* is delimited, DB2 does not convert the value to uppercase.

package-id or * (For REBIND only)

Specifies the name of the package to rebind, as listed in column NAME of catalog table SYSPACKAGE. There is no default.

You can use an asterisk (*) to rebind all local packages in *collection-id* for which you have bind privileges.

For REBIND PACKAGE, *package-id* can be an undelimited or a delimited identifier. The delimiter for *package-id* is double quotation marks ("). If *package-id* is delimited, DB2 does not convert the value to uppercase.

version-id or * (For REBIND only)

Specifies the version of the package to rebind, as listed in column VERSION of catalog table SYSPACKAGE.

You can use an asterisk (*) to rebind all local versions of the specified *package-id* in *collection-id* for which you have bind privileges.

Using simply () rebinds the version of the package that is identified by the empty string.

If you omit *version-id*, the default depends on the how you specify *package-id*. If you use * for *package-id*, then *version-id* defaults to *. If you explicitly provide a value for *package-id*, then *version-id* defaults to the empty string version.

(*) (For REBIND only)

Rebinds all local DB2 packages for which the applicable authorization ID has the BIND privilege. Specifying (*) is the same as specifying the package name as (*.*(*)) or (*.*). The applicable authorization ID is:

- The value of OWNER, if you use that option
- The primary authorization ID of the process running the bind, if you do not use the option OWNER

Catalog record: Columns COLLID, NAME, and VERSION of table SYSPACKAGE.

PATH bind option

The PATH option determines the SQL path that DB2 uses to resolve unqualified stored procedure names in CALL statements, in user-defined data types, and in functions.

PATH	(<i>schema-name</i>) (USER) (<i>schema-name</i> , USER , ...)	On: BIND and REBIND PLAN and PACKAGE Not valid for REBIND of a native SQL procedure package.
-------------	--	---

For the **PATH** option, consider the following guidelines when you specify a *schema-name*:

- The specified schema names are *not* folded to uppercase by DB2. This behavior is different than that for schema names in SQL statements, which are folded to uppercase before being stored in the catalog. If you do not specify these nondelimited schema names in upper case, DB2 cannot find a match in the catalog for those schema names.
- You can specify delimited identifiers in both mixed and uppercase characters.

The **PATH** keyword is mutually exclusive with the **PATHDEFAULT** keyword. Do not specify both keywords in the same REBIND command.

(*schema-name*)
Identifies a schema.

(*schema-name* , ...)
Identifies a list of schemas. The same schema name should not appear more than once in the SQL path.

The number of schemas that you can specify is limited by the length of the resulting SQL path, which cannot exceed 2048 bytes. To calculate the length of the resulting SQL path:

1. Take the length of each schema.
2. Add 2 for delimiters around each *schema-name* in the list.
3. Add 1 for each comma after each schema. Do **not** add 1 for the last schema.

USER

Represents a maximum 128-byte *schema-name* . At bind time, DB2 includes this 128-byte length in the total length of the list of schema names specified for the PATH bind option. The maximum length for a list of schema names, including comma separators, delimiters, and the 128-byte USER value, is 2048 bytes. If you exceed this limit, DB2 generates an error message at bind time.

At run time, DB2 substitutes the run time value of the USER special register, which contains the primary authorization ID of the run time process, for the *schema-name* in the position of USER in the PATH *schema-name* list.

If you specify USER in a list of schema names, do not use delimiters around the USER keyword.

DB2 does not validate that the specified schema actually exists at precompile or at bind time.

You do not need to explicitly specify the SYSIBM, SYSFUN, and SYSPROC schemas; DB2 implicitly assumes that these schemas are at the beginning of the SQL path. DB2 adds these schemas in the order listed. If you do not specify the SYSIBM, SYSFUN, and SYSPROC schemas, they are not included in the 2048-byte length.

Interactions with the PLANMGMT option: If you plan to change this option and the PLANMGMT option in a REBIND command, see “PLANMGMT bind option” on page 144 for the implications.

Defaults:

Process	Default value
BIND PLAN	SYSIBM, SYSFUN, SYSPROC, <i>plan-qualifier</i>
BIND PACKAGE	SYSIBM, SYSFUN, SYSPROC, <i>package-qualifier</i>
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column PATHSCHEMAS of table SYSPACKAGE.

The default values for BIND PLAN and BIND PACKAGE are not stored in the catalog. The default values in the catalog are a zero-length string.

DB2 for z/OS accepts FUNCPATH as a synonym for PATH when FUNCPATH is received from a remote requester. However, FUNCPATH cannot be specified on a BIND or REBIND command.

PATHDEFAULT bind option

The PATHDEFAULT option resets the PATH value for a package or plan to SYSIBM, SYSFUN, SYSPROC, *plan qualifier*, or to SYSIBM, SYSFUN, SYSPROC, *package qualifier* .

PATHDEFAULT	On: REBIND PLAN and PACKAGE
	Not valid for REBIND of a native SQL procedure package.

The PATHDEFAULT keyword is mutually exclusive with the PATH keyword. Do not specify both keywords in the same REBIND command.

Interactions with the PLANMGMT option: If you plan to change this option and the PLANMGMT option in a REBIND command, see “PLANMGMT bind option” on page 144 for the implications.

Defaults:

Process	Default value
BIND PLAN	N/A
BIND PACKAGE	N/A
REBIND PLAN	None
REBIND PACKAGE	None

PKLIST and NOPKLIST bind options

The PKLIST option determines the packages to include in a package list. The NOPKLIST option deletes packages from a package list.

PKLIST NOPKLIST	(<i>location-name.collection-id.package-id</i> , ...) PKLIST only	On: BIND and REBIND PLAN
------------------------	---	---------------------------------

For PKLIST, the order in which you list packages with partial identifiers determines the search order at run time and can affect performance.

NOPKLIST is used with REBIND PLAN only. NOPKLIST determines that the plan is rebound without a package list. If a package list already exists, NOPKLIST deletes it.

If you specify REBIND PLAN for a plan that was bound prior to DB2 10 and includes DBRMs, DB2 binds those existing DBRMs in the plan into packages. In this case, the PKLIST and NOPKLIST options have the following effects:

- If you specify PKLIST, the resulting plan includes both the newly created DBRM packages and the packages in the specified package list. The DBRM packages are included at the front of the newly created package list.
- If you specify NOPKLIST, any existing package lists are removed from the plan. NOPKLIST does not delete the package list that is created by binding the existing DBRMs into packages. However, if you use NOPKLIST a second time on the same plan that only contains package lists, all package lists are deleted.

location-name or *

Names the location of the DBMS where the package resides, or defers that choice until run time. Use either a particular location name or an asterisk (*), or omit this part of the identifier. The default is the local DBMS.

- If you use a particular location name, then that DBMS should be defined in catalog table SYSIBM.LOCATIONS. If that table does not exist or if the DBMS is not in it, you receive warning messages.
- If you use an asterisk, at run time the location comes from the special register CURRENT SERVER. DB2 checks privileges to use the SQL statements in the package at that location.

collection-id or *

Names the collection that contains the package or defers that choice until run time. Use either a particular collection ID or an asterisk (*). No default exists.

If you use an asterisk, then DB2 checks the privileges to use the SQL statements that are embedded in the package at run time. At that time also, DB2 determines the collection ID as follows:

- If the value in the special register CURRENT PACKAGESET is not blank, then that value is the collection ID.
- If the value of CURRENT PACKAGESET is blank, DB2 skips the entry unless it is the last entry in the package list. If it is the last or only entry, an error message is issued.

package-id or *

Names a particular package or specifies, by the asterisk, all packages in the collection. Because you cannot specify a *version-id* for the packages included in the package list, all versions are effectively included.

Defaults:

Process	Default value
BIND PLAN	None. If you do not specify PKLIST, you must specify MEMBER.
BIND PACKAGE	N/A
REBIND PLAN	Existing value
REBIND PACKAGE	N/A

Catalog record: Table SYSPACKLIST.

PLAN bind option

The PLAN option determines the plan or plans to bind or rebind.

PLAN	(<i>plan-name</i>) * (*) (REBIND PLAN only)	On: BIND and REBIND PLAN
------	---	---------------------------------

(*plan-name*)

Specifies the name of the application plan.

For REBIND only, the value of column NAME in the catalog table SYSPLAN; you can use a list of plan names.

(*) (For REBIND only)

Rebinds all plans for which the applicable authorization ID has the BIND privilege. The applicable ID is:

- The value of OWNER, if you use that option
- The authorization ID of the process running the bind, if you do not use the option OWNER

Catalog record: Column NAME of table SYSPLAN.

PLANMGMT bind option

The PLANMGMT option retains, during a rebind operation, all relevant package information (such as metadata, query text, dependencies, authorizations, and access paths) in catalog tables and in the directory.

PLANMGMT	(OFF) (BASIC) (EXTENDED)	On: REBIND PACKAGE and REBIND TRIGGER PACKAGE
----------	--------------------------------------	--

If performance regression occurs, you can direct DB2 to fall back to the older copy of a package. Over time, a package can have multiple copies that exist on disk storage, but only one of those copies is the active or current copy. All other copies are inactive.

(OFF)

If none of the following options are changed, the current access path is removed and replaced with the incoming copy, and any previous or original copies are unaffected.

- OWNER
- QUALIFIER
- ENABLE
- DISABLE
- PATH

- PATHDEFAULT
- IMMEDIATEWRITE

If any of the previously listed options are changed, all package copies (current, previous, and original) are purged.

(BASIC)

Discard the previous copy of a package. The current copy becomes the previous copy, and the incoming copy becomes the current copy. If an original copy of a package exists, it remains available.

If the PLANMGMT option is specified explicitly and you change any of the following options, DB2 issues an error message:

- OWNER
- QUALIFIER
- ENABLE
- DISABLE
- PATH
- PATHDEFAULT
- IMMEDIATEWRITE

(EXTENDED)

Discard the previous copy of a package. The current copy becomes the previous copy, and the original copy is managed as follows:

- If no original copy exists, the current copy is cloned to become the original.
- If an original copy exists, it is retained as the original.

In each case, the incoming copy of a package becomes the new current copy.

If the PLANMGMT option is specified explicitly and you change any of the following options, DB2 issues an error message:

- OWNER
- QUALIFIER
- ENABLE
- DISABLE
- PATH
- PATHDEFAULT
- IMMEDIATEWRITE

Note: If the PLANMGMT setting is inherited from the value of the PLANMGMT subsystem parameter and the value of one of the previously listed options changes, DB2 internally turns off the PLANMGMT option. As a result, DB2 purges the prior copies of a package. However, if one of the previously listed options is specified but the value does not change, DB2 processes the command as if the option were not specified. If none of the previously listed options has a change in value, an error message is not issued, and the PLANMGMT option is not turned off.

Pattern-matching characters: PLANMGMT settings remain valid when pattern-matching characters (*) are used in the REBIND syntax. When you use REBIND PACKAGE to rebind more than one package, DB2 retains previous and original copies for each package separately.

Defaults:

Process	Default value
BIND PLAN	N/A
BIND PACKAGE	N/A
REBIND PLAN	N/A
REBIND PACKAGE	Value of subsystem parameter PLANMGMT
REBIND TRIGGER PACKAGE	Value of subsystem parameter PLANMGMT

Catalog record: Column PLANMGMT of table SYSPACKAGE.

QUALIFIER bind option

The QUALIFIER option determines the implicit qualifier for unqualified names of tables, views, indexes, and aliases contained in the plan or package.

QUALIFIER	(<i>qualifier-name</i>)	On: BIND and REBIND PLAN and PACKAGE
		Not valid for REBIND of a native SQL procedure package.

(*qualifier-name*)

Specifies the value of the implicit qualifier. This value is not subject to translation when sent to a remote system for BIND or REBIND PACKAGE.

Interactions with the PLANMGMT option: If you plan to change this option and the PLANMGMT option in a REBIND command, see “PLANMGMT bind option” on page 144 for the implications.

Defaults:

Process	Default value
BIND PLAN	Authorization ID of the plan owner
BIND PACKAGE	Authorization ID of the package owner
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column QUALIFIER of tables SYSPACKAGE and SYSPLAN.

QUERYACCELERATION bind option

The QUERYACCELERATION bind option specifies whether a static SQL query is bound for acceleration, and if so, with what behavior.

When a package that is bound with this bind option runs, the QUERYACCELERATION value, instead of the QUERY_ACCELERATION subsystem parameter, provides the initial value for the CURRENT QUERY ACCELERATION special register. The CURRENT QUERY ACCELERATION special register specifies the acceleration behavior for dynamic SQL queries. Therefore, by rebinding the application package with the QUERYACCELERATION bind option, you can also accelerate dynamic SQL queries in an application. You can specify this bind option instead of either modifying the application to add an explicit SET CURRENT QUERY ACCELERATION statement or setting the QUERY_ACCELERATION subsystem parameter. For dynamic SQL queries in a package, the bind option value that you specify overrides the

QUERY_ACCELERATION subsystem parameter. This bind option does not have a default value, so if you do not specify this bind option, the QUERY_ACCELERATION subsystem parameter initializes the special register.

Restriction: The QUERYACCELERATION bind option applies only to cursor queries, the SELECT portion of the SQL INSERT from SELECT statement, and static SQL SELECT INTO statements that are run locally. If a static SQL SELECT INTO statement is bound for acceleration but run remotely, DB2 fails the statement at run time and returns a negative SQL code.

QUERYACCELERATION	(NONE) (ENABLE) (ENABLEWITHFAILBACK) (ELIGIBLE) (ALL)	On: BIND and REBIND PACKAGE Not valid for REBIND of a native SQL procedure package.
-------------------	---	--

(NONE)

Specifies that no static SQL query in the application is bound for acceleration or will be accelerated when the application is run.

(ENABLE)

Specifies that a static SQL query is bound for acceleration if it satisfies the acceleration criteria, including the cost and heuristics criteria. The query is routed to an accelerator when the application runs. Otherwise, if the static query does not satisfy the acceleration criteria, the query is bound for execution in DB2.

If an error condition, such as one of the following examples, occurs while executing the accelerated static query when the application is run, DB2 fails the static query and returns a negative SQL code to the application:

- A failure occurs while running the static query on the accelerator.
- The accelerator returns an error for the query.
- The accelerator is not started and DB2 cannot route the static query to the accelerator for execution.

(ENABLEWITHFAILBACK)

Results in the same behavior as ENABLE, except if one of the error conditions occurs on the first OPEN of the accelerated static query when the application is run. In this case, instead of failing the static query and returning a negative SQL code to the application, DB2 performs a temporary *statement-level* incremental bind of the query and runs the query in DB2. The application does not see the acceleration failure. Failback to DB2 is not possible after the application does a successful OPEN for the query on the accelerator.

(ELIGIBLE)

Specifies that a static SQL query is bound for acceleration if the query meets the basic acceleration criteria, regardless of the cost or heuristics criteria. The query is routed to the accelerator when the application runs.

Like the behavior for ENABLE, if an error condition occurs while executing the accelerated static query when the application is run, DB2 fails the static query and returns a negative SQL code to the application.

(ALL)

Specifies that all of the static SQL queries in the application are to be bound for acceleration and routed to the accelerator when the application runs. If DB2 determines that a static query cannot be bound to run on the accelerator and the query references a user base table or view, the BIND or REBIND

PACKAGE operation fails with an error message for that query. (A failure exception is made for declared global temporary tables (DGTTs) and created global temporary tables and (CGTTs) because these tables cannot be accelerated.)

Like the behavior for ENABLE, if an error condition occurs while executing the accelerated static query when the application is run, DB2 fails the static query and returns a negative SQL code to the application.

The DB2 catalog table SYSIBM.SYSPACKSTMT has a STATUS column value of '0' for static SQL queries that have been bound for acceleration. For cursor-based static queries, the DECLARE CURSOR, OPEN, FETCH, and CLOSE statements each have a SYSPACKSTMT.STATUS column value of '0'. For a static INSERT from SELECT statement, where the SELECT has been bound for acceleration, the SYSPACKSTMT.STATUS column value is also '0'.

Defaults:

Process	Default value
BIND PACKAGE	None
REBIND PACKAGE	None

There is no default value for this bind option. The QUERYACCELERATION bind option does not inherit the setting from the QUERY_ACCELERATION subsystem parameter.

Related concepts:

 [How DB2 determines whether to accelerate eligible queries \(DB2 Performance\)](#)

Related tasks:

 [Enabling DB2 for IBM DB2 Analytics Accelerator for z/OS \(DB2 Performance\)](#)

QUERYID bind option

The QUERYID option frees entries in the SYSIBM.SYSQUERY table, and the SYSIBM.SYSQUERYPLAN table or the SYSIBM.SYSQUERYOPTS table.

QUERYID	(<i>number</i>) (ALL)	On: FREE QUERY
---------	------------------------------	----------------

number

Frees an entry in the SYSIBM.SYSQUERY table that has the same QUERYID value, and the corresponding entries in the SYSIBM.SYSQUERYPLAN table or the SYSIBM.SYSQUERYOPTS table.

ALL

Frees all the entries from the SYSIBM.SYSQUERY table and the corresponding entries from the SYSIBM.SYSQUERYPLAN table or the SYSIBM.SYSQUERYOPTS table.

RELEASE bind option

The RELEASE option determines when to release resources that a program uses, either at each commit point or when the program terminates.

RELEASE	(COMMIT) (DEALLOCATE) (INHERITFROMPLAN)	On: BIND and REBIND PLAN and PACKAGE, REBIND TRIGGER PACKAGE
		Not valid for REBIND of a native SQL procedure package.

Partition locks follow the same rules as table space locks, and all partitions are held for the same duration. Thus, if one package is using RELEASE(COMMIT) and another is using RELEASE(DEALLOCATE), all partitions use RELEASE(DEALLOCATE).

The RELEASE option does not affect page, row, LOB or XML locks.

(COMMIT)

Releases resources at each commit point, unless cursors are held. If the application accesses the object again, it must acquire the lock again.

- All tables and table spaces are unlocked when:

TSO, Batch, and CAF

An SQL COMMIT or ROLLBACK statement is issued, or your application process terminates

IMS A CHKP or SYNC call (for single-mode transactions), a GU call to the I/O PCB, or a ROLL or ROLB call is completed

CICS A SYNCPOINT command is issued.

Exception:

If the cursor is defined WITH HOLD, table or table space locks necessary to maintain cursor position are held past the commit point.

- Table, partition, or table space locks are released at the next commit point unless the cursor is defined WITH HOLD.
- The least restrictive lock needed to execute each SQL statement is used except when a more restrictive lock remains from a previous statement. In that case, that lock is used without change.

DEALLOCATE

Releases resources only when the thread terminates.

If a lock is to be held past commit and it is an S, SIX, or X lock on a table space or a table in a segmented table space, DB2 sometimes demotes that lock to an intent lock (IX or IS) at commit. DB2 demotes a gross lock if it was acquired for one of the following reasons:

- DB2 acquired the gross lock because of lock escalation.
- The application issued a mass delete (DELETE FROM *object* without a WHERE clause or TRUNCATE).

RELEASE(DEALLOCATE) has no effect on dynamic SQL statements, which always use RELEASE(COMMIT), with one exception: When you use RELEASE(DEALLOCATE) and KEEPYNAMIC(YES), and your subsystem is

installed with the CACHEDYN subsystem parameter set to YES, the RELEASE(DEALLOCATE) option is honored for dynamic SELECT, INSERT, UPDATE and DELETE statements.

Packages that are executed on a DB2 server through a DRDA connection with a client system honor the RELEASE(DEALLOCATE) bind option. However, if the MODIFY DDF PKGREL(COMMIT) command has been issued at a DB2 server, the RELEASE(DEALLOCATE) option has no effect on packages that are executed on that server through a DRDA connection with a client system.

Locks that are acquired for dynamic statements are held until one of the following events occurs:

- The application process ends (deallocation).
- The application issues a PREPARE statement with the same statement identifier. (Locks are released at the next commit point.)
- The statement is removed from the cache because it has not been used. (Locks are released at the next commit point.)
- An object that the statement is dependent on is dropped or altered, or a privilege that the statement needs is revoked. (Locks are released at the next commit point.)

RELEASE(DEALLOCATE) can increase the package or plan size, because additional items become resident in the package or plan.

INHERITFROMPLAN

Enables a local package to inherit the value of the RELEASE option from the plan, regardless of whether the package was bound remotely or locally.

If you bind a package remotely with the RELEASE(INHERITFROMPLAN) option and the remote server does not understand the INHERITFROMPLAN value, the server might return an error.

The RELEASE(INHERITFROMPLAN) value is not applied in the following situations, because no associated plan exists:

- If you bind the application locally and then copy the package to a remote server.
- If you bind an application that uses RRSAF.
- For any packages that are created for utilities


In these cases, RELEASE(COMMIT) is in effect for the package.

Defaults:

Process	Default value
BIND PLAN	COMMIT
BIND PACKAGE	<ul style="list-style-type: none">• For a local server: Plan value• For a remote server: COMMIT
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value
REBIND TRIGGER PACKAGE	Existing value

Catalog record: Column RELEASE of tables SYSPACKAGE and SYSPLAN.

Related tasks:

 [Choosing a RELEASE option \(DB2 Performance\)](#)

Related reference:

REOPT bind option

The REOPT option specifies whether DB2 determines an access path at run time by using the values of host variables, parameter markers, and special registers.

REOPT	(<u>NONE</u>) (ALWAYS) (ONCE) (AUTO)	On: BIND and REBIND PLAN and PACKAGE Not valid for REBIND of a native SQL procedure package.
-------	---	---

For dynamic statements, the REOPT bind option also controls whether DB2 considers the literal values for access path selection when CONCENTRATE STATEMENTS WITH LITERALS is specified when statements are prepared. The literal values are considered only when REOPT(ONCE) or REOPT(AUTO) is specified.

(NONE)

Does not determine an access path at run time. You can use NOREOPT(VARS) as a synonym for REOPT(NONE).

(ALWAYS)

Determines the access path again at run time each time the statement is run. DB2 determines access paths at both bind time and run time for statements that contain one or more of the following variables:

- Host variables
- Parameter markers
- Special registers

At run time, DB2 uses the values in those variables to determine the access paths. You can use REOPT(VARS) as a synonym for REOPT(ALWAYS).

(ONCE)

Determines the access path for any dynamic statement only once, at the first run time or at the first time the statement is opened. This access path is used until the prepared statement is invalidated or removed from the dynamic statement cache and needs to be prepared again.

(AUTO)

Autonomically determines if a new access path needs to be generated to further optimize the performance for each execution. DB2 determines the access path at the first run time and then at each subsequent execution, DB2 evaluates whether a new access path is needed. If so, DB2 generates a new access path and uses that access path until another new access path is generated. For cached dynamic statements that reference parameter markers, a new path can be generated at any execution. The new path is based on changes to estimated filter factors for predicates that result from changes to parameter marker values.

Usage notes:

If you specify the bind option REOPT(ALWAYS), REOPT(AUTO), or REOPT(ONCE), DB2 sets the bind option DEFER(PREPARE) automatically. However, when you specify REOPT(ONCE), DB2 determines the access path for a statement only once (at the first run time).

You cannot use REOPT(ALWAYS) with the KEEPDYNAMIC(YES) option.

The following restrictions apply to REOPT(ONCE):


- REOPT(ONCE) is ignored if you use it with static SQL statements because DB2 for z/OS caches only dynamic statements.
- If a dynamic statement in a plan or package that is bound with REOPT(ONCE) runs when dynamic statement caching is turned off, the statement runs as if REOPT(ONCE) is not specified.
- You **cannot** use both REOPT(ONCE) and NODEFER(PREPARE).
- You **can** use both REOPT(ONCE) and KEEPYNAMIC(YES).

Defaults:

Process	Default value
BIND PLAN	NONE
BIND PACKAGE	NONE
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column REOPTVAR of the SYSPLAN and SYSPACKAGE catalog tables.

Related concepts:

 Differences between static and dynamic SQL (DB2 Application programming and SQL)

Related tasks:

 Reoptimizing SQL statements at run time (DB2 Performance)

Related reference:

 SYSIBM.SYSPLAN table (DB2 SQL)

 SYSIBM.SYSPACKAGE table (DB2 SQL)

ROUNDING bind option

The ROUNDING option specifies the rounding mode at bind time. Use rounding mode to manipulate DECFLOAT data.

ROUNDING	(CEILING)	On: BIND and REBIND PLAN and PACKAGE
	(DOWN)	
	(FLOOR)	Not valid for REBIND of a native SQL procedure package.
	(HALFDOWN)	
	(HALFEVEN)	
	(HALFUP)	
	(UP)	

CEILING

Round toward +infinity. If all of the discarded digits are zero or if the sign is negative the result is unchanged other than the removal of discarded digits. Otherwise, the result coefficient should be incremented by 1 (round up).

DOWN

Round toward 0 (TRUNCATION). The discarded digits are ignored.

FLOOR

Round toward -infinity. If all of the discarded digits are zero or if the sign

is positive the result is unchanged other than the removal of discarded digits. Otherwise, the sign is negative and the result coefficient should be incremented by 1.

HALFDOWN

Round to the nearest; if equidistant, round down. If the discarded digits represent greater than half (0.5) of the value of a one in the next left position then the result coefficient should be incremented by 1 (rounded up). Otherwise, (the discarded digits are 0.5) or less) the discarded digits are ignored.

HALFEVEN

Round to the nearest; if equidistant, round so that the final digit is even. If the discarded digits represent greater than half (0.5) the value of a one in the next left position then the result coefficient should be incremented by 1 (rounded up). If they represent less than half, then the result coefficient is not adjusted (that is, the discarded digits are ignored). Otherwise, (they represent exactly half) the result coefficient is unaltered if its rightmost digit is even, or incremented by 1 (rounded up) if its rightmost digit is odd (to make an even digit).

This option is the default.

HALFUP

Round to nearest; if equidistant, round up. If the discarded digits represent greater than or equal to half (0.5) of the value of a one in the next left position then the result coefficient should be incremented by 1 (rounded up). Otherwise, the discarded digits are ignored.

UP

Round away from 0. If all the discarded digits are zero the result is unchanged other than they removal of discarded digits. Otherwise, the result coefficient should be incremented by 1 (rounded up).

The default value of ROUNDING option is DEF DECFLOAT ROUNDING MODE from application defaults load module.

Defaults:

Process	Default value
BIND PLAN	HALFEVEN
BIND PACKAGE	HALFEVEN
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column ROUNDING of tables SYSPACKAGE and SYSPLAN, and column ROUNDING_MODE of table SYSENVIRONMENT.

SQLERROR bind option

The SQLERROR option determines whether to create a package if SQL errors occur.

SQLERROR	(NOPACKAGE) (CONTINUE) (CHECK)	On: BIND PACKAGE
----------	--------------------------------------	------------------

(NOPACKAGE)

Creates no package if an error occurs.

(CONTINUE)

Creates a package, even if errors occur when binding SQL statements. The statements in error cannot execute. Any attempt to execute them at run time causes errors.

(CHECK)

Performs all syntax and semantic checks on the SQL statements being bound when the authorization ID for BIND does not have the privilege to execute the statements. A package is not created as part of this process. If an existing package with the same name and version is encountered during bind processing, the existing package is not dropped or replaced, even if **ACTION(REPLACE)** was specified.

Defaults:

Process	Default value
BIND PLAN	N/A
BIND PACKAGE	NOPACKAGE
REBIND PLAN	N/A
REBIND PACKAGE	N/A. Because you cannot use the option SQLERROR for REBIND PACKAGE , the value for the previous package remains in effect when you rebind that package. If you rebind a package that uses SQLERROR(CONTINUE) , SQL statements that were in error at BIND time are not rebound.

Catalog record: Column **SQLERROR** of table **SYSPACKAGE**.

SQLRULES bind option

The **SQLRULES** option determines whether you can execute a type 2 **CONNECT** statement to an existing SQL connection, according to DB2 rules.

SQLRULES	(<u>DB2</u>) (<u>STD</u>)	On: BIND and REBIND PLAN
-----------------	----------------------------------	---------------------------------

Alternatively, the statement causes an error, according to the ANSI/ISO SQL standard of 1992. This option applies to any application process that uses the plan and executes type 2 **CONNECT** statements. It has no effect on type 1 **CONNECT** statements.

(DB2)

No error occurs if **CONNECT** identifies an existing SQL connection. If X is an existing SQL connection, **CONNECT TO X** makes X the current connection. If X is already the current connection, **CONNECT TO X** has no effect on the state of any connections.

(STD)

An error occurs if **CONNECT** identifies an existing SQL connection. Therefore, if X is a dormant SQL connection, you must use the SQL statement **SET CONNECTION** to make X the current connection.

For local operations, the value of **SQLRULES** is used for the initial value of the SQL special register **CURRENT RULES**.

Defaults:

Process	Default value
BIND PLAN	DB2
BIND PACKAGE	N/A
REBIND PLAN	Existing value
REBIND PACKAGE	N/A

Catalog record: Column SQLRULES of table SYSPLAN.

SWITCH bind option

The SWITCH option causes the previous copy or original copy of a package to become the current copy of the package.

SWITCH	(PREVIOUS)(ORIGINAL)	On: REBIND PACKAGE and REBIND TRIGGER PACKAGE
--------	--------------------------	--

This option allows you to revert to an older copy of a package in the event of a performance regression. You cannot specify SWITCH with any other rebind options. If the package that you specify does not have the previous or original copy to switch to, DB2 issues an error message, and continues processing the rest of the list. Use this option with the pattern-matching character (*) in the syntax.

(PREVIOUS)

Switches between the current and previous copies of a package or a plan. The current copy takes the place of the previous copy, and the previous copy takes the place of the current copy.


(ORIGINAL)

Moves the current copy to take the place of the previous copy. Any previous copy that exists is purged. The original copy is cloned to take the place of the current copy.

Defaults:

Process	Default value
BIND PLAN	N/A
BIND PACKAGE	N/A
REBIND PLAN	N/A
REBIND PACKAGE	Existing value
REBIND TRIGGER PACKAGE	Existing value

Related concepts:

 [Package copies \(DB2 Performance\)](#)

Related tasks:

 [Switching to previous access paths \(DB2 Performance\)](#)

VALIDATE bind option

The VALIDATE option determines whether to recheck, at run time, errors of the types "OBJECT NOT FOUND" and "NOT AUTHORIZED" that are found during bind or rebind.

VALIDATE	(<u>RUN</u>) (BIND)	On: BIND and REBIND PLAN and PACKAGE
		Not valid for REBIND of a native SQL procedure package.

The option has no effect if all objects and needed privileges exist.

(RUN)

Indicates that if not all objects or privileges exist at bind time, the process issues warning messages, but the bind succeeds. DB2 checks existence and authorization again at run time for SQL statements that failed those checks during bind. The checks use the authorization ID of the plan or package owner.

If you specify the VALIDATE(RUN) bind option, and the application to be bound contains an error with a SET host-variable assignment statement, the bind process still issues only warning messages, not error messages.

(BIND)

Indicates that if not all objects or needed privileges exist at bind time, the process issues error messages, and does not bind or rebind the plan or package, *except that*:

For BIND PACKAGE only, if you use the option SQLERROR(CONTINUE), the bind succeeds, but the SQL statements in it that have errors cannot execute.

With VALIDATE(BIND), DB2 does not check authorization for the LOCK TABLE statement and some CREATE, ALTER, and DROP statements until run time.

Defaults:

Process	Default value
BIND PLAN	RUN
BIND PACKAGE	RUN
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

Catalog record: Column VALIDATE of tables SYSPACKAGE and SYSPLAN.

Chapter 19. Specification of DB2 for z/OS bind options from IBM Data Server clients and drivers

When you bind packages on DB2 for z/OS data servers from IBM Data Server clients and drivers, you need to specify bind options that are supported by DB2 for z/OS but not by DB2 for Linux, UNIX, and Windows as generic bind options.

You can use the following IBM Data Server client and driver interfaces to specify DB2 for z/OS-only bind options:

- For Java clients, when you use IBM Data Server Driver for JDBC and SQLJ utility `DB2Binder` or method `DB2Binder.runJDBCBinder` to bind IBM Data Server Driver for JDBC and SQLJ packages or rebind user packages, specify the DB2 for z/OS bind options as sub-options of the `-bindOptions` parameter. For example:

```
java com.ibm.db2.jcc.DB2Binder -url jdbc:db2://srv1.svl.ibm.com:446/DB1
-user myid -password mypass -action replace
-bindOptions "IMMEDWRITE YES"
```

- For non-Java clients, when you use CLI method `SQLCreatePkg` to bind packages, specify the DB2 for z/OS bind options as sub-options of the `GENERIC` option in the `szBindOpts` string. For example:

```
strcpy (bindFileName, "/u/user1/sqllib/bnd/@all.lst");
cliRC = SQLCreatePkg(hdbc,
                    bindFileName,
                    strlen(bindFileName),
                    "GENERIC=IMMEDWRITE NO"
                    -3, // SQL_NTS);
```

You can specify the following DB2 for z/OS bind options as generic bind options:

- APCOMPARE
- APRETAINDUP
- APREUSE
- CONCURRENTACCESSRESOLUTION
- DBPROTOCOL
- DEFER(PREPARE) or NODEFER(PREPARE)
- EXPLAIN
- EXTENDEDINDICATOR
- IMMEDWRITE
- OPTHINT
- PATHDEFAULT (rebind only)
- PLANMGMT (rebind only)
- REOPT
- ROUNDING
- SWITCH (rebind only)

Chapter 20. -CANCEL THREAD (DB2)

The DB2 command **CANCEL THREAD** cancels processing for specific local or distributed threads.

Abbreviation: -CAN THD

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or a CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

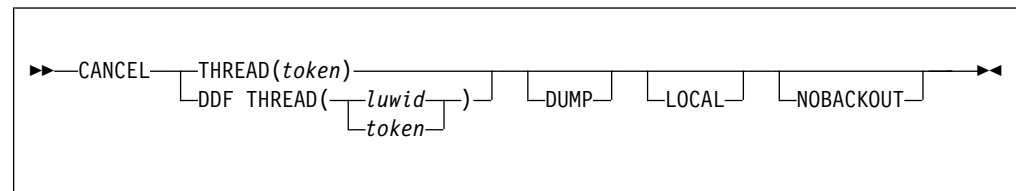
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SYSOPR authority
- SYSTRM authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

THREAD (*token*)

Identifies a specific thread, either distributed or not, whose processing you want to cancel. DB2 assigns a token to each thread that is unique for that DB2 subsystem, but not necessarily unique across subsystems.

The token is a one- to six-digit decimal number. You can determine what the token is by issuing the DB2 command DISPLAY THREAD or by using an IFI READS call for IFCID 0147 or 0148. The token can also appear after the equal sign in DB2 messages that display an LUWID. If the token is 0, the thread of that token cannot be canceled.

DDF THREAD(*luwid*)

Identifies distributed threads for which you want to cancel processing. *luwid* is a logical unit of work identifier (LUWID), consisting of:

- A fully qualified LU network name, which consists of:

- A one- to eight-character network ID
- A period
- A one- to eight-character network LU name
- An LUW instance number, which consists of 12 hexadecimal characters that uniquely identify the unit of work

If you enter three fields separated by periods, DB2 assumes that you are entering an LUWID.

Two or more distributed threads might have the same LUWID. All distributed threads with the same LUWID are canceled.

The LUWID can be determined from the DB2 DISPLAY THREAD command and other DB2 messages.

DUMP

Specifies that a dump is generated when CANCEL THREAD is issued, and the thread is active in DB2. The dump can be used for diagnostic purposes.

When you cancel a thread that is not currently active in DB2, DB2 performs a hard cancel and no dump is provided. A thread is considered to be not currently active in DB2 when it has left DB2 to perform application work.

LOCAL

Specifies that DB2 search the home address space of the allied agent for a thread that matches *token*. If DB2 finds a thread, it performs a soft cancel on that thread. The LOCAL option applies only to a CANCEL THREAD command that is issued through the IFI interface. The IFI application must run in the same address space as the thread that needs to be canceled, and the thread must be connected to DB2 through the RRS attachment facility.

NOBACKOUT

Specifies that DB2 is not to attempt to back out the data during transaction rollback processing. Canceling the thread with NOBACKOUT leaves objects in an inconsistent state. Do not issue this command with NOBACKOUT unless you have a plan to resolve the data inconsistency.

Multiple NOBACKOUT requests are allowed. However, if the thread is active and the request is accepted, subsequent requests are ignored. You can choose to issue a subsequent request if a request fails (as indicated by message DSNI032I). Objects that the thread modifies are recovered (backed out). If back out processing fails, the objects are marked REFRESH PENDING (REFP) and either RECOVER PENDING (RECP) or REBUILD PENDING (RBDP or PSRBD) in the database exception table. Resolve the REFP status of the object by running the RECOVER utility to recover the object to a prior point in time or by running LOAD REPLACE on the object.

Usage notes

Canceling distributed threads: Canceling a distributed thread can cause the thread to enter the indoubt state. Message DSNL450I is issued if the CANCEL command causes the DDF thread to be converted from active to indoubt. DB2 releases the resources that the thread holds when the indoubt state is resolved by automatic indoubt resolution with the coordinator, or by resolution with the command RECOVER INDOUBT.

If a thread that is specified in the command is part of a global transaction, the command is executed against all threads in the global transaction.

The CANCEL command schedules a thread to be terminated in DB2. To terminate, the thread must be processing within DB2. If the thread does not terminate, it could be:

- Processing outside of DB2, possibly in the application. If that is the case, the thread does not terminate until the application makes a request to DB2. Use the z/OS CANCEL command to terminate the application immediately.
- Hung up in a network operation. Use VTAM or TCP/IP commands to cause the network operation to return processing to DB2, which will then allow the thread to be terminated. See the following topic for details.

Canceling threads on applications that update NOT LOGGED tables spaces: Take care when you issue a CANCEL command for an application that updates a NOT LOGGED table space. The CANCEL THREAD command can cause the NOT LOGGED table space to be placed in the LPL, so that the table space must be recovered.

Canceling local threads: The CANCEL command schedules a thread to terminate. Threads that are not in DB2 terminate immediately.

Canceling SNA distributed threads with VTAM Commands: If the CANCEL command does not terminate a distributed thread, it is possible that it is hung up in VTAM. Use the VTAM VARY NET,TERM command to cancel the thread's VTAM sessions.

To cancel a thread's VTAM session, you need to know the VTAM session IDs (SIDs) that correspond to the thread. Take the following steps:

1. Issue the DB2 command DISPLAY THREAD(*) LUWID(nnnn) DETAIL. (The value of *nnnn* is the token or LUWID provided by CANCEL DDF THREAD.)

This gives you the VTAM session IDs that must be canceled. Sessions are identified by the column header SESSID as shown in the following DISPLAY THREAD output:

```
-DIS THD(*) LUWID(123) DETAIL
DSNV401I - DISPLAY THREAD REPORT FOLLOWS:
DSNV402I - ACTIVE THREADS:
NAME      ST A  REQ ID          AUTHID  PLAN      ASID TOKEN
BATCH    TR *   5 BKH2C          SYSADM  BKH2      000D  123
V444-DB2NET.LUND0.C4B23F1F4D06=123 ACCESSING DATA AT
( 1)SAN JOSE-LUND1
V447--INDEX SESSID          A ST TIME
V448--( 1) 00D3590EA1E89701  S1   0923816181452
V448--( 1) 00D3590EA1E89822  N R1  0923816181584
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

The N indicates the thread is processing in VTAM.

2. Record positions 3 through 16 of SESSID for the threads to be canceled. (In the preceding DISPLAY THREAD output, the values are D3590EA1E89701 and D3590EA1E89822.)
3. Issue the VTAM command DISPLAY NET to display the VTAM session IDs. The ones you want to cancel match the SESSIDs in positions 3 through 16 and the corresponding session IDs are in bold. The following is an output example of this command:

```
D NET, ID=LUND0, SCOPE=ACT
```

```
IST097I DISPLAY ACCEPTED
```

```
IST075I NAME = LUND0, TYPE = APPL
```

```
IST486I STATUS= ACTIV, DESIRED STATE= ACTIV
```

```
IST171I ACTIVE SESSIONS = 0000000005, SESSION REQUESTS = 0000000000
```

```
IST206I SESSIONS:
```

IST634I	NAME	STATUS	SID	SEND	RECV	VR	TP	NETID
IST635I	LUND1	ACTIV-S	D24B171032B76E65	0051	0043	0	0	NET2
IST635I	LUND1	ACTIV-S	D24B171032B32545	0051	0043	0	0	NET2
IST635I	LUND1	ACTIV-R	D2D3590EA1E89701	0022	0031	0	0	NET2
IST635I	LUND1	ACTIV-R	D2D3590EA1E89802	0022	0031	0	0	NET2
IST635I	LUND1	ACTIV-R	D2D3590EA1E89822	0022	0031	0	0	NET2
IST314I	END							

4. Issue the VTAM command VARY NET,TERM for each of the VTAM SIDs associated with the DB2 thread. In this case, you might need to cancel only the session ID that DISPLAY THREAD shows to be processing in VTAM (D2D3590EA1E89822).

Canceling TCP/IP Distributed Threads with TCP/IP Commands: If the CANCEL command does not terminate a distributed thread, the thread might be hung up in TCP/IP. Use the TCP/IP DROP command to cancel the thread's connection ID. To do this, you need to first determine the TCP/IP connection ID that corresponds to the thread.

Depending on whether the thread is a DB2 requester or server thread, take the following steps:

- Terminating TCP/IP connection for a requester thread:
 1. Issue the DB2 command DISPLAY THREAD(*) LUWID(nnnn) DETAIL. (The value of *nnnn* is the token or LUWID provided by DISPLAY THREAD.)

Find the IP address and local port for the connection to the partner, as shown in the following DISPLAY THREAD output:

```
#display thread(*) detail
```

```
DSNV401I # DISPLAY THREAD REPORT FOLLOWS -
DSNV402I # ACTIVE THREADS -
NAME      ST A   REQ ID          AUTHID  PLAN    ASID  TOKEN
TEST0001 TR      4 CTHDCORID001 SYSADM  DONSQ1  0027  19
V444-USIBMSY.SYEC715B.C4B220851392=19 ACCESSING DATA AT
( 1)-STL714A-::FFFF:9.112.114.102..446
V447--INDEX SESSID          A ST TIME
V448--( 1) 1028:446          N R2 0923814011448
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I # DSNVDT '-DIS THD' NORMAL COMPLETION
```

In this case, the partner's IP address and port is 9.112.114.102 446, and the local port is 1028. N indicates that the thread is processing in TCP/IP.

2. Determine the associated TCP/IP connection ID:

```
d tcpip,,netstat,conn,ipaddr=9.112.114.102
```

```
EZZ2500I NETSTAT CS V2R10 TCPIP
USER ID  CONN    LOCAL SOCKET          FOREIGN SOCKET        STATE
V71BDIST 0000049D 9.112.114.103..1028  9.112.114.102..446   ESTBLSH
1 OF 1 RECORDS DISPLAYED
```

3. Terminate the connection:

```
v tcpip,,drop,conn=0000049d
```

```
EZZ0060I PROCESSING COMMAND: VARY TCPIP,,DROP,
CONN=0000049D
EZZ0053I COMMAND VARY DROP COMPLETED SUCCESSFULLY
```


- Terminating TCP/IP connection for a server thread::
 1. Issue the DB2 command DISPLAY THREAD(*) LUWID(nnnn) DETAIL. (The value of nnnn is the token or LUWID provided by CANCEL THREAD.)

Find the IP address and local port for the connection to the partner, as shown in the following DISPLAY THREAD output:

```
!display thread(*) detail

DSNV401I ! DISPLAY THREAD REPORT FOLLOWS -
DSNV402I ! ACTIVE THREADS -
NAME      ST A   REQ ID          AUTHID  PLAN    ASID TOKEN
TEST0001 RA *    2 CTHDCORID001 SYSADM  DONSQ1  002D    11
V445-USIBMSY.SYEC715B.C4B24232F81D=11 ACCESSING DATA FOR
( 1)::FFFF:9.112.114.103
V447--INDEX SESSID          A ST TIME
V448--( 1) 446:1029         W R2 0923816315680
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I ! DSNVDT '-DIS THD' NORMAL COMPLETION
```

In this case, the partner's IP address is 9.112.114.103 and the local port is 1029.

2. Determine the associated TCP/IP connection ID:

```
d tcpip,,netstat,conn,ipaddr=9.112.114.103

EZZ2500I NETSTAT CS V2R8 TCPIP
USER ID  CONN    LOCAL SOCKET          FOREIGN SOCKET        STATE
V61ADIST 0000048E 9.112.114.102..446    9.112.114.103..1029  ESTABLS
1 OF 1 RECORDS DISPLAYED
```

Find the entry where the foreign socket shows the partner's IP address and port (9.112.114.103 1029) and note the CONN.

3. Terminate the connection:

```
v tcpip,,drop,conn=0000048e

EZZ0060I PROCESSING COMMAND: VARY TCPIP,,DROP,
CONN=0000048E
EZZ0053I COMMAND VARY DROP COMPLETED SUCCESSFULLY
```

Using TCP/IP commands to cancel accelerated threads: If the CANCEL command does not terminate an accelerated thread (a thread that has active accelerator processes running), the thread might be hung up in TCP/IP. Use the TCP/IP DROP command to cancel the thread's connection ID. To do this, you need to first determine the TCP/IP connection ID that corresponds to the thread.

- To terminate a TCP/IP connection for an accelerated thread:
 1. Issue the DB2 command DISPLAY THREAD(*) ACCEL(*) DETAIL.
 2. Find the IP address and local port for the connection to the partner, as shown in the following DISPLAY THREAD output:

```
)DISPLAY THREAD(*) ACCEL(*) DETAIL

DSNV401I ) DISPLAY THREAD REPORT FOLLOWS -
DSNV402I ) ACTIVE THREADS - 681
NAME      ST A   REQ ID          AUTHID  PLAN    ASID TOKEN
BATCH     AC *   39 ACCEL1       SYSADM  DSNTEP2 002B    3
V666 ACC=BLINK1,ADDR=:::FFFF:9.30.30.177..446:1080
V436-PGM=DSNTEP2.DSNTEP2, SEC=1, STMNT=2256
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I ) DSNVDT '-DIS THD' NORMAL COMPLETION
```

In this case, the partner's IP address is 9.30.30.177, the port is 446, and the local port is 1080. AC indicates that the thread is processing in an accelerator.

3. Determine the associated TCP/IP connection ID:

```

|
| D TCPIP,,NETSTAT,CONN,IPADDR=9.30.30.177
|
| EZD0101I NETSTAT CS V1R10 TCPIP
| USER ID  CONN    STATE
| V91ADBM1 000009E3 ESTBLSH
| LOCAL SOCKET:  ::FFFF:9.30.222.34..1080
| FOREIGN SOCKET: ::FFFF:9.30.30.177..446
| 1 OF 1 RECORDS DISPLAYED
| END OF THE REPORT
|

```

In this case, the associated TCP/IP connection ID is 000009E3.

4. Terminate the connection:

```

| V TCPIP,,DROP,CONN=000009E3
|
| EZZ0060I PROCESSING COMMAND: VARY TCPIP,,DROP,CONN=000009E3
| EZZ0053I COMMAND VARY DROP COMPLETED SUCCESSFULLY
|

```

Examples

Example: Canceling a local thread

To cancel a non-distributed thread whose token you found through the DISPLAY THREAD command and to produce a diagnostic dump, issue:

```
-CANCEL THREAD (123) DUMP
```

Example: Canceling a distributed thread

To cancel a distributed thread whose LUWID you found through the DISPLAY THREAD command, issue:


```
-CANCEL DDF THREAD (LUDALLAS.DB2SQL1.3042512B6425)
```

Suppose that the output from -DISPLAY THREAD shows that the thread-ID and token associated with this LUWID is 45162. You can also cancel this thread by issuing either of the following commands:

```
-CANCEL DDF THREAD (45162)
```

```
-CANCEL THREAD (45162)
```

Related information:

 VTAM DISPLAY NET command (SNA Operation)

 VTAM VARY NET,TERM command (SNA Operation)

Chapter 21. /CHANGE (IMS)

The IMS command /CHANGE resets an indoubt unit of recovery as identified by the OASN keyword of the /DISPLAY command. That command deletes the item from the standpoint of IMS, but it does not communicate to DB2.

Abbreviation: /CHA

Subsections:

- “Environment”
- “Authorization”
- “Syntax”
- “Option descriptions”
- “Usage note” on page 166
- “Examples” on page 166

Environment

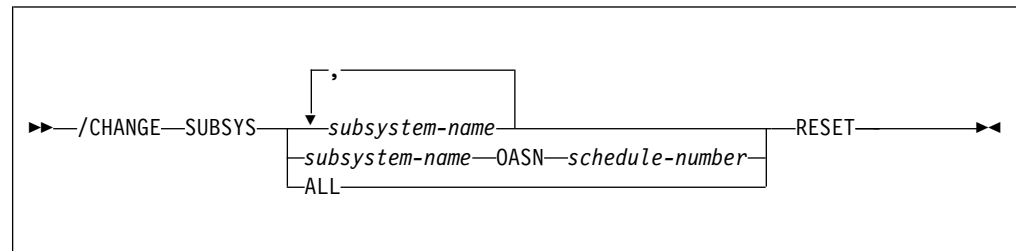
This command can be issued only from an IMS terminal.

Data sharing scope: Member

Authorization

This command requires an appropriate level of IMS authority.

Syntax



Option descriptions

SUBSYS

Deletes IMS recovery elements from one or more subsystems. You must code one of the following subparameters:

subsystem-name

Specifies one or more subsystems, separated by commas, from which recovery elements will be deleted.

subsystem-name OASN schedule-number

Deletes one or more origin application schedule numbers, separated by commas, from one subsystem, specified by *subsystem-name*.

schedule-number can be a list of up to 32768 origin application schedule numbers. The numbers are displayed using the OASN parameter of the /DISPLAY command.

ALL

Deletes IMS recovery elements from all subsystems.

RESET

Deletes the indoubt recovery unit. The recovery unit represents an incomplete unit of work assigned to an external subsystem as the result of an application request.

Usage note

The preceding description of the /CHANGE command is a partial description only. The complete description is documented in the IMS library.

Examples

Example: Resetting all indoubt units of recovery for a single subsystem

Issue the following command to reset all indoubt units of recovery for the subsystem named DB2:

```
/CHA SUBSYS DB2 RESET
```

Example: Resetting all indoubt units of recovery

Issue the following command to reset all indoubt units of recovery for all subsystems:

```
/CHA SUBSYS ALL RESET
```

Example: Specifying the indoubt units of recovery to reset by specifying origin application schedule numbers

Issue the following command to reset indoubt recovery units with OASN numbers 99, 685, and 2920 for subsystem DB2:

```
/CHA SUBSYS DB2 OASN 99 685 2920 RESET
```

Chapter 22. DCLGEN (DECLARATIONS GENERATOR) (DSN)

The declarations generator (DCLGEN) produces an SQL DECLARE TABLE statement and a COBOL, PL/I, or C data declaration for a table or a view named in the catalog.

Environment

The declarations generator is executed by the DSN subcommand DCLGEN. That subcommand can be issued from a DSN session, running in either foreground or background mode, or it can be issued through DB2I.

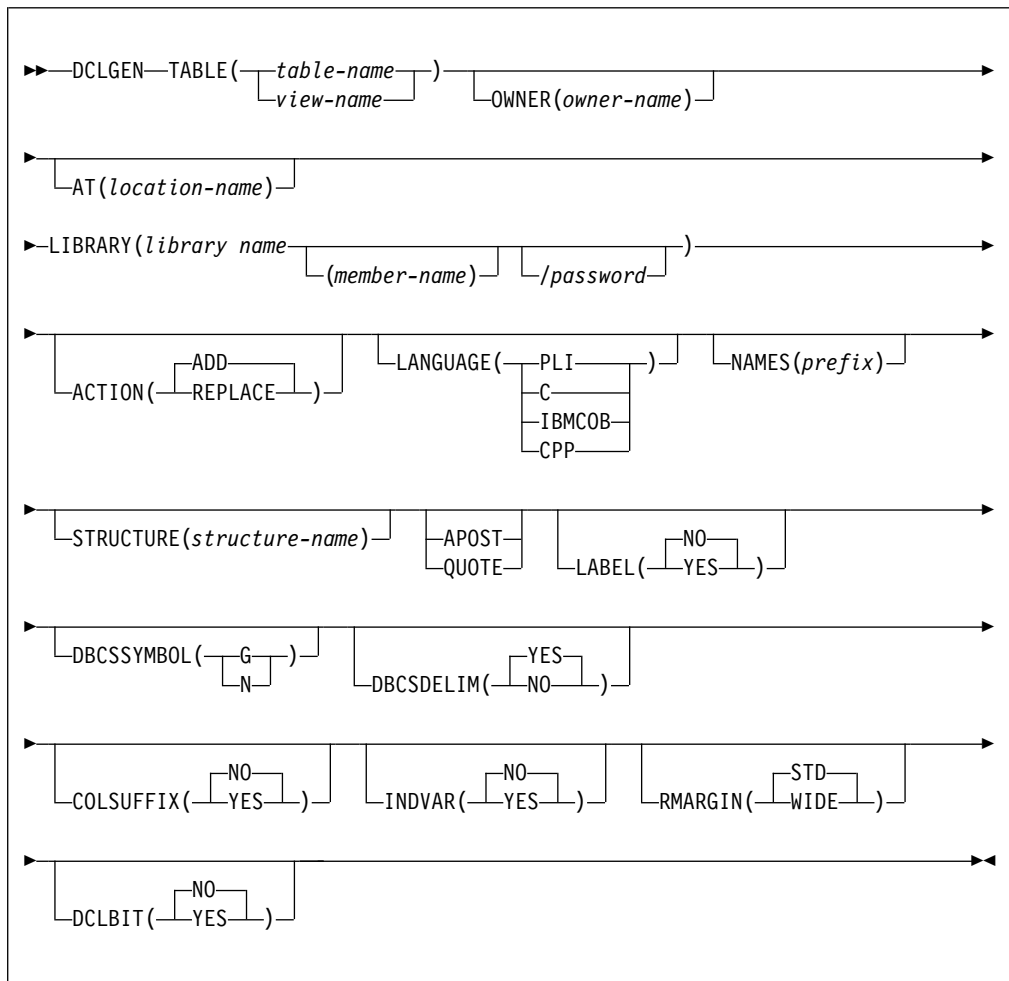
Data sharing scope: Group

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- Ownership of the table or view
- SELECT privilege on the table or view
- DBADM authority on the database containing the table
- SYSADM authority
- SYSCTRL authority (catalog tables only)

Syntax



Option descriptions

TABLE

Specifies the table or view for which a declaration is generated. *table-name* or *view-name* is the qualified or unqualified name of the table or view.

The name must follow these rules:

- If the name is a single-byte or mixed string and contains special characters other than underscores (`_`), it must be enclosed between apostrophes (`'`). If the language is COBOL, single-byte underscores in the name are translated into hyphens (`-`) by DCLGEN. Double-byte character set (DBCS) names need not be enclosed in apostrophes.
- If the name contains single-byte apostrophes, each one must be doubled (`"`). (Some host languages do not permit apostrophes in variable names.)

A table or view name that contains a period and is not enclosed by apostrophes is a qualified table name. The characters to the left of the period constitute the table owner, and those to the right of the period constitute the table name. Any table name enclosed in apostrophes is an unqualified table name. To understand how DCLGEN determines the table name qualifier, see the description of the OWNER option, which follows.

OWNER(*owner-name*)

Specifies a qualifier for the table name. *owner-name* is the qualifier for the table name.

If you specify a qualified table name for the TABLE(*table-name*) option, and you also specify OWNER(*owner-name*), the qualifier portion of *table-name* supersedes *owner-name* as the table name qualifier. If you specify an unqualified table name for the TABLE(*table-name*) option, and you do not specify OWNER(*owner-name*), the SQL authorization ID is the table name qualifier.

DCLGEN supports the use of underscore (_) as a valid character in the *owner-name* keyword parameter.

The following table illustrates the decision process for determining the DCLGEN table name qualifier.

Table 12. Decision process for determining the DCLGEN table name qualifier

Table name	OWNER(<i>owner-name</i>) specified	OWNER(<i>owner-name</i>) not specified
TABLE(<i>table-name</i>) qualified	<i>table-name</i> qualifier	<i>table-name</i> qualifier
TABLE(<i>table-name</i>) unqualified	<i>owner-name</i>	SQL authorization ID

AT(*location-name*)

Identifies the location of the table or view name specified in TABLE (*table-name*). *location-name* , which can consist of 1 to 16 characters, uniquely identifies an instance of a table or view in a network.

If you specify AT, *location-name* is used as the prefix for the table name, and *table-name* or *table-view* must be a qualified name.

DCLGEN supports the use of underscore (_) as a valid character in the *location-name* keyword parameter.

LIBRARY(*library-name* (*member-name*) / *password*)

Specifies the data set into which the declarations go. This data set must already exist and be accessible to the declarations generator. It can be either sequential or partitioned. *password* is optional.

If the library name is not enclosed within apostrophes, DCLGEN constructs the following full data set name:

user-prefix.library-name.language.(member-name)

where:

user-prefix

The user prefix of the primary authorization ID of the transaction.

language

The value of the LANGUAGE option: PLI, C, or COBOL.

(*member-name*)

Optional; if not used, the output goes to a sequential data set.

ACTION

Indicates whether to add or replace the data set.

(ADD)

Adds the data set as a new member, if it does not already exist.

The **default** is **ACTION (ADD)**.

(REPLACE)

Replaces an existing member or data set with the new one. If the output is to a partitioned data set, and no member exists with the given name, one is added.

LANGUAGE

Specifies the language of the generated declaration.

Possible languages are:

- **(PLI)**, for PL/I
- **(C)**, for C/370™
- **(IBMCOB)**, for IBM COBOL
- **(CPP)**, for C++

NAMES(*prefix*)

Allows field names to be formed in the declaration.

Avoid possible name conflicts between DCLGEN output and the source program. If a conflict occurs, use **NAMES** or **STRUCTURE**, or manually edit the generated declaration or source program.

prefix can contain double-byte characters.

The field names consist of *prefix* concatenated with a number from one to three digits in length. *prefix* can have up to 28 characters. If *prefix* is a single-byte or mixed string and the first character is not alphabetic, it must be enclosed in apostrophes. For example, if *prefix* is ABCDE, the field names will be ABCDE1, ABCDE2, and so on, up to a maximum of ABCDE999. Special characters can be used, but use caution to avoid possible name conflicts.

For COBOL and PL/I, if the prefix is a DBCS string, the field name will be the DBCS prefix concatenated with the DBCS representation of the number. For example, if *prefix* is <D1D2D3> (where "<" and ">" represent shift-out and shift-in characters, respectively, and D1D2D3 represent double-byte characters), generated field names will be <D1D2D3.1>, <D1D2D3.2>, and so on. The period (.) represents X'42'.

The column names in the table are taken as default names for the fields in the output.

STRUCTURE(*structure-name*)

Specifies the generated data structure.

structure-name can have up to 31 characters. If *structure-name* is a single-byte or mixed string and the first character is not alphabetic, it must be enclosed in apostrophes. You can use special characters, but use caution to avoid possible name conflicts.

structure-name can contain double-byte characters.

For SQL output, the name is the same as the table or view name. If the host language is C, the default structure name is the prefix DCL concatenated with the table name. If the host language is COBOL or PL/I and the table name is a single-byte or mixed string, the default structure name is also the prefix DCL concatenated with the table name. If the host language is COBOL or PL/I and the table name is a DBCS string, the default structure name is the prefix <.D.C.L> concatenated with the table or view name. "<" and ">" represent shift-out and shift-in characters, respectively. You must guard against possible

conflicts with names in the source program. DCLGEN allows the specified structure name to be the same as the table or view name, but will issue a warning message.

APOST or QUOTE

Specifies the string delimiter character used in the host language. This option is effective only for COBOL programs.

APOST specifies the apostrophe (') as the host language string delimiter; the SQL delimiter is the quotation mark (").

QUOTE specifies the quotation mark (") as the host language delimiter; the SQL delimiter is the apostrophe (').

If neither APOST nor QUOTE is specified, the **default** is either APOST or QUOTE for COBOL, depending on what was specified on DB2 installation panel DSNTIPF.

The string delimiter delimits strings in host language statements. The SQL escape character delimits table and column names in the SQL DECLARE TABLE statement produced by DCLGEN. It is possible, by a choice made during DB2 installation, to make both delimiters the quotation mark or both the apostrophe.

LABEL

Indicates whether to include column labels in the output as comments. (Column labels can be assigned by the LABEL ON statement.)

(NO)

Omits the column labels.

(YES)

Includes the column labels.

DBCSSYMBOL

Specifies the symbol used to denote a graphic data type in a COBOL PICTURE clause.

(G)

Graphic data is denoted using G.

(N)

Graphic data is denoted using N.

DBCSDELIM

Specifies whether the DBCS table and column names in the generated DECLARE table statement will be delimited.

(YES)

DBCS table and column names will be delimited in the DCLGEN table declaration.

(NO)

DBCS table and column names will not be delimited in the DCLGEN table declaration.

COLSUFFIX

Determines whether to form field names by attaching the column name to the prefix given by the NAMES option.

(NO)

The column name is not used as a suffix, and field names are controlled by the option NAMES.

(YES)

If NAMES is specified, DCLGEN forms field names by adding column names as a suffix to the value of NAMES. For example, if the prefix given by NAMES is "NEW" and the column name is EMPNO, the field name is "NEWEMPNO".

If NAMES is **not** specified, DCLGEN issues a warning message and uses the column names as the field names.

INDVAR

Determines whether to create an indicator variable array for the host variable structure.

(NO)

DCLGEN does not create an indicator variable array.

(YES)

DCLGEN creates an indicator array for the host variable structure. The array name is the table name with a prefix of "I" (or DBCS letter "<I>" if the table name is double-byte).

RMARGIN

Specifies the break point for statement tokens that must be wrapped onto one or more subsequent records of DCLGEN output.

(STD)

Statement tokens will be wrapped after column 72.

(WIDE)

Statement tokens will be wrapped after column 80.

DCLBIT

Specifies whether to generate a DECLARE VARIABLE statement of SQL variables for columns that are defined as FOR BIT DATA. This statement is required in IBM COBOL applications that have host variables for FOR BIT DATA columns and are prepared by using the SQLCCSID option of the integrated DB2 coprocessor. The statement is also valid, but not currently required, for C/C++ and PL/I, and for COBOL that is not prepared by using the SQLCCSID option of the integrated DB2 coprocessor.

(NO)

Does not generate a DECLARE VARIABLE statement of SQL variables for columns that are defined as FOR BIT DATA.

(YES)

Generates a DECLARE VARIABLE statement of SQL variables for columns that are defined as FOR BIT DATA. If the table or view does not have any FOR BIT DATA columns, the statement is not generated.

Usage notes

Parsing of the DCLGEN command conforms to standard TSO parsing conventions.

The DECLARE statement: The DECLARE statement generated by DCLGEN will define all columns created with a data type of VARCHAR or LONG VARCHAR as VARCHAR. Columns created with a data type of VARGRAPHIC or LONG VARGRAPHIC will be defined as VARGRAPHIC.

Comments: The output for all host languages includes comments. The leading comment block echoes the DCLGEN subcommand that requested the declarations. The trailing comment block indicates the number of variables declared.

Using the output: To include the DCLGEN output in an application program, use the SQL INCLUDE statement. The same member name specified in the DCLGEN LIBRARY parameter is specified on the INCLUDE statement.

Prompts: Online TSO will prompt for missing or incorrectly specified options.

Editing the output: It is expected that the output of DCLGEN will not meet every need. You can freely edit the output before including it in a program. For example, you might want to change a variable name, or include SQL escape characters.

You can edit the output to add WITH DEFAULT to NOT NULL for columns that do not allow null values. If you edit the output, you must provide a default value.

If your column names contain embedded blanks, they will also be reflected in the host variable declarations, and you will have to remove, or translate, any blank characters to some other value.

For a column with an XML data type, DCLGEN generates the following output: SQL TYPE IS XML AS CLOB(1M). The default length value for the XML host variable is 1MB. You can manually update the DCLGEN output if you want a larger or smaller size.

C: DCLGEN support of the C language is unique in the following ways:

- DCLGEN does not fold the STRUCTURE, NAMES, or TABLE values to uppercase.
- For any DB2 column that has the data type CHAR(*n*), where *n* > 1, DCLGEN generates the corresponding host variable as CHAR(*n* + 1) to avoid the DB2 warning. For *n* = 1, the corresponding host variable is CHAR.

| **COBOL and binary integers:** DB2 uses the full size of binary integers. It can place
| larger values than allowed in the specified number of digits in a COBOL variable,
| which can result in truncated values. To avoid this problem, you can modify the
| DCLGEN output to declare COBOL variables that correspond to binary integer
| columns with USAGE COMP-5 instead of USAGE COMP or USAGE BINARY.

| **COBOL and the underscore character:** DCLGEN translates any underscore
| characters in the table's column names into hyphens (-) for use in the generated
| structure.

| **COBOL and DBCS:** Although DB2 accepts values outside of the range from X'41'
| to X'FE', in COBOL data definition statements, both bytes of each double-byte
| character in data names must be within this range. Data names must also contain
| at least one DBCS character that does not have X'42' as its first byte.

| **Data declarations for arrays of indicator variables:** If DCLGEN creates an array of
| indicator variables, data declarations have the following form:

Language	Data declaration
C	short int <i>Itable-name</i> [<i>n</i>];
COBOL	01 <i>Itable-name</i> PIC S9(4) USAGE COMP OCCURS <i>n</i> TIMES
PL/I	DCL <i>Itable-name</i> (<i>n</i>) BIN FIXED (15);

n is the number of columns in the table.

Examples

Example: Using DCLGEN to generate PL/I host variable declarations for columns in a DB2 table

The following subcommand generates PL/I declarations for table VEMPL and writes them to data set member *prefix*.SRCLIB.DATA(DSN8MPPEM). The host structure and field names are generated from the table and column names.

```
DCLGEN TABLE(VEMPL) -  
      LIBRARY('prefix.SRCLIB.DATA(DSN8MPPEM)') -  
      LANGUAGE(PLI) -  
      APOST
```

The output looks like this:

```
/******  
/* DCLGEN TABLE(VEMPL) - */  
/* LIBRARY('prefix.SRCLIB.DATA(DSN8MPPEM)') - */  
/* LANGUAGE(PLI) - */  
/* APOST */  
/* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS */  
/******  
EXEC SQL DECLARE VEMPL TABLE  
      ( EMPNO CHAR(6) NOT NULL,  
        FIRSTNME VARCHAR(12) NOT NULL,  
        MIDINIT CHAR(1) NOT NULL,  
        LASTNAME VARCHAR(15) NOT NULL,  
        WORKDEPT CHAR(3) NOT NULL  
      ) ;  
/******  
/* PLI DECLARATION FOR TABLE VEMPL */  
/******  
DCL 1 DCLVEMPL,  
      5 EMPNO CHAR(6),  
      5 FIRSTNME CHAR(12) VAR,  
      5 MIDINIT CHAR(1),  
      5 LASTNAME CHAR(15) VAR,  
      5 WORKDEPT CHAR(3);  
/******  
/* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 5 */  
/******
```

Example: Using DCLGEN NAMES and STRUCTURE options to specify a field name prefix and structure name for the generated output

The following subcommand generates PL/I declarations for table VEMPL and writes them to data set member *prefix*.SRCLIB.DATA(DSN8MPPEM). The generated PL/I declarations are in a structure named EMPRECORD, and all host variable names consist of the string FIELD, followed by a number.

```
DCLGEN TABLE(VEMPL) -  
      LIBRARY('prefix.SRCLIB.DATA(DSN8MPPEM)') -  
      LANGUAGE(PLI) -  
      NAMES(FIELD) -  
      STRUCTURE(EMPRECORD) -  
      APOST
```

The output looks like this:

```
/******  
/* DCLGEN TABLE(VEMPL) - */  
/* LIBRARY('prefix.SRCLIB.DATA(DSN8MPPEM)') - */  
/* LANGUAGE(PLI) - */  
/* NAMES(FIELD) - */  
/* STRUCTURE(EMPRECORD) - */  
/* APOST */
```

```

/* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS */
/*****
EXEC SQL DECLARE VEMPL TABLE
      ( EMPNO          CHAR(6) NOT NULL,
        FIRSTNME      VARCHAR(12) NOT NULL,
        MIDINIT        CHAR(1) NOT NULL,
        LASTNAME       VARCHAR(15) NOT NULL,
        WORKDEPT       CHAR(3) NOT NULL
      ) ;

/*****
/* PLI DECLARATION FOR TABLE VEMPL */
/*****
DCL 1 EMPRECORD,
      5 FIELD1  CHAR(6),
      5 FIELD2  CHAR(12) VAR,
      5 FIELD3  CHAR(1),
      5 FIELD4  CHAR(15) VAR,
      5 FIELD5  CHAR(3);
/*****
/* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 5 */
/*****

```

Example: Generating DECLARE variable statements for columns that are defined with FOR BIT DATA

The following subcommand generates COBOL declarations for table MYTABLE, which contains FOR BIT DATA columns. The DCLBIT(YES) option is specified to cause DCLGEN to generate DECLARE VARIABLE statements for the FOR BIT DATA columns so that applications that include the generated declaration compile correctly when they are compiled with the SQLCCSID option of the DB2 coprocessor.

```

DCLGEN TABLE(MYTABLE)
      LIBRARY('prefix.SRCLIB.DATA(MYTABLE))
      LANGUAGE(COBOL)
      APOST
      DCLBIT(YES)

```

The output looks like this:

```

*****
* DCLGEN TABLE(MYTABLE) *
* LIBRARY('prefix.SRCLIB.DATA(MYTABLE)) *
* LANGUAGE(COBOL) *
* APOST *
* DCLBIT(YES) *
* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS *
*****
EXEC SQL DECLARE MYTABLE TABLE
      ( COL1          CHAR(10) NOT NULL,
        COL2          CHAR(10),
        COL3          VARCHAR(12) NOT NULL,
        COL4          VARCHAR(12) NOT NULL
      ) END-EXEC.
*****
* DECLARED VARIABLES FOR 'FOR BIT DATA' COLUMNS *
*****
EXEC SQL DECLARE
      :COL2
      ,:COL4
      VARIABLE FOR BIT DATA END-EXEC.
*****
* COBOL DECLARATION FOR TABLE MYTABLE *
*****
01 DCLMYTABLE.
   10 COL1          PIC X(10).
   10 COL2          PIC X(10).

```

```
10 COL3.  
  49 COL3-LEN      PIC S9(4) USAGE COMP.  
  49 COL3-TEXT     PIC X(12).  
10 COL4.  
  49 COL4-LEN      PIC S9(4) USAGE COMP.  
  49 COL4-TEXT     PIC X(12).
```

```
*****  
* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 4 *  
*****
```

Chapter 23. /DISPLAY (IMS)

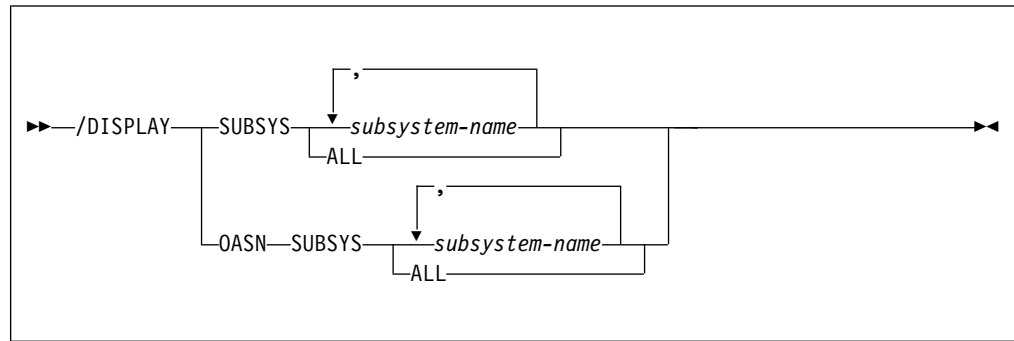
The IMS command /DISPLAY displays the status of the connection between IMS and an external subsystem (as well as all application programs communicating with the external subsystem), or the outstanding recovery units that are associated with the subsystem.

Data sharing scope: Member

Authorization

This command requires an appropriate level of IMS authority.

Syntax



Option descriptions

One of the following options is required:

SUBSYS

Identifies the subsystems to display information about.

subsystem-name , ...

Specifies one or more subsystems.

ALL

Displays information about all subsystems.

OASN SUBSYS

Displays the outstanding recovery units (origin application schedule numbers, or OASN) associated with the external subsystems. The OASN is assigned by IMS when it schedules an application into a dependent region. That, coupled with the IMS ID, becomes the recovery token for units of work distributed to other subsystems.

subsystem-name , ...

Specifies one or more subsystems to display information about.

ALL

Displays the outstanding recovery units associated with all external subsystems.

Output

The command recognition character (CRC) is displayed for each external subsystem. Subsystem status is one of the following conditions:

CONNECTED

An IMS control region or dependent region has successfully connected to the external subsystem. At this point, the two systems can begin a normal dialog.

NOT CONNECTED

The external subsystem is in an idle state. That is, either it has not been the object of the /START SUBSYS command, or the external subsystem initialization exit routine indicated not to connect.

CONNECT IN PROGRESS

The connection process for the specified subsystem is in progress.

STOPPED

The specified subsystem has been stopped with the /STOP SUBSYS command. All region connections to the specified external subsystem have been terminated.

STOP IN PROGRESS

The /STOP SUBSYS command is in progress. Before it completes successfully, all active connections to the specified subsystem from all IMS regions must be quiesced.

INVALID SUBSYSTEM NAME = *subsystem-name*

The indicated subsystem name has not been defined to the IMS subsystem PROCLIB member. Add the subsystem definition to the subsystem member and issue the /START SUBSYS command.

SUBSYSTEM *subsystem-name* NOT DEFINED BUT RECOVERY OUTSTANDING

The indicated subsystem name has not been defined to IMS in the external subsystem PROCLIB member, but IMS still has outstanding recovery elements from a previous execution when the name was known. To resolve the recovery element problem, either add the indicated subsystem definition to the external subsystem PROCLIB member and then issue the /START SUBSYS command, or issue the /DISPLAY OASN SUBSYS command to determine the identification of the OASNs and then manually resolve the recovery elements by issuing the /CHANGE SUBSYS RESET command.

TERM IN PROGRESS

An internal termination of the subsystem is underway. This type of termination was instigated by IMS abnormal condition processing, an external subsystem exit, or the external subsystem.

A thread between an IMS dependent region and an external subsystem is created when an application program in the region establishes a connection to the external subsystem. The status of threads to an external subsystem is listed under the status of the subsystem. The absence of a list of threads under a connected subsystem indicates that no threads to the specified subsystem have been established.

Thread status can be one of the following conditions:

CONNECTED(CONN)

An IMS control region or dependent region has successfully connected to the external subsystem.

ACTIVE

An IMS application program has established communication with an external subsystem.

The absence of a PSB name for a thread indicates that a connection to the external subsystem exists, but an application program is not currently occupying the region. The presence or absence of an LTERM name indicates whether a region is message-driven.

Note: The preceding description of the /DISPLAY command is a partial description only.

Examples

Example of displaying the status of connections between IMS and all subsystems

Issue the following command to display the status of all connections with IMS:

```
/DISPLAY SUBSYS ALL
```

Output similar to this output is generated:

SUBSYS	CRC	REGID	PROGRAM	LTERM	STATUS
SSTR	?				CONN
		1	DDLTLM17	PTERM01	CONN,ACTIVE
		2	DDLTLM06	PTERM02	CONN

85202/065933

Chapter 24. -DISPLAY ACCEL (DB2)

The DISPLAY ACCEL command displays information about accelerator servers.

Abbreviation: -DIS ACCEL

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program that uses the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the SCOPE option.

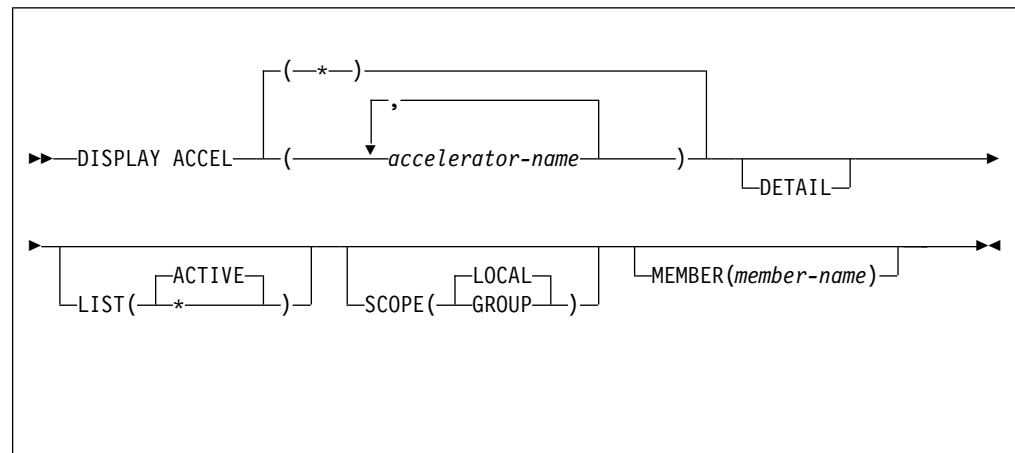
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- DISPLAY privilege
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization by using primary and secondary authorization IDs.

Syntax



Option descriptions

(*accelerator-name*)

The accelerator server name. This option limits the display to the specified accelerator servers.

- (*) Requests a list of all accelerator servers, whether the servers are currently in use or not. Specifying an asterisk (*) as the accelerator name indicates that the display must include all accelerator servers.

DETAIL

Displays additional information for one or more of the accelerator servers. If DETAIL is not specified, a basic summary report is produced.

LIST

Produces a list of accelerator servers that are currently in use or that have been started. Valid values are:

(ACTIVE)

Restricts the list of accelerator servers to those that are currently in use.

- (*) Requests a list of all accelerator servers, whether they are currently in use or not. An accelerator that has just been created appears in the list only if it has been started by using the START ACCEL command.

SCOPE

Specifies the scope of the command. In a non-data-sharing environment, the option is ignored. Valid values are:

(LOCAL)

Displays information only for the accelerator servers for the current data sharing member.

(**GROUP**)

Displays information for accelerator servers for all members of the data sharing group.

MEMBER

Restricts the display for the identified accelerator server to specific members of the data sharing group. The default behavior is to display accelerator servers for the local member. In a non-data-sharing environment, this option is ignored.

Output

Message DSNX830I (DB2 Messages) indicates the beginning of the output of the command.

Examples

Example 1: Displaying information about all accelerator servers for a data sharing group

The following command displays information about all of the accelerator servers for all members in a data sharing group:

```
-DIS ACCEL(*) SCOPE(GROUP) LIST(*)
```

The output is similar to the following example:

```
DSNX810I  -DB1A DSNX8CMD DISPLAY ACCEL FOLLOWS -
DSNX830I  -DB1A DSNX8CDA
ACCELERATOR          MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
ACCEL1              DB1A  STARTED   32769     2    5   23
ACCEL2              DB1A  STOPPED   37235     1    7   17
ACCEL3              DB1A  STARTED   3256     5   23   41
DISPLAY ACCEL REPORT COMPLETE
DSN9035I  -DB1A BEGIN OF DISPLAY FOR MEMBER: DB1B
-----
```

```

DSNX830I -DB1B DSNX8CDA
ACCELERATOR          MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
ACCEL1                DB1B  STARTED    23456    1    0    2
ACCEL2                DB1B  STOPPED     47    0    0    0
ACCEL3                DB1B  STOPPED     92    0    0    2
DISPLAY ACCEL REPORT COMPLETE
-----END OF DISPLAY FOR MEMBER: DB1B -----
DSN9035I -DB1A BEGIN OF DISPLAY FOR MEMBER: DB1C
-----
DSNX830I -DB1C DSNX8CDA
ACCELERATOR          MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
ACCEL1                DB1C  STARTED     734    0    0    4
ACCEL2                DB1C  STARTED      2    0    0    0
ACCEL3                DB1C  STARTED     87    0    0    7
DISPLAY ACCEL REPORT COMPLETE
-----END OF DISPLAY FOR MEMBER: DB1C -----
DSN9035I -DB1A BEGIN OF DISPLAY FOR MEMBER: DB1D
-----
DSNX830I -DB1D DSNX8CDA
ACCELERATOR          MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
ACCEL1                DB1D  STARTED    9210    0    0    1
ACCEL2                DB1D  STOPPED      0    0    0    0
ACCEL3                DB1D  STOPPED     21    0   11   11
DISPLAY ACCEL REPORT COMPLETE
-----END OF DISPLAY FOR MEMBER: DB1D -----
DSN9022I -DB1A DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION

```

Example 2: Displaying detailed information about specific accelerator servers

The following command displays detailed information about active accelerator servers ACCEL1 and ACCEL2 for data sharing member DB1D:

```

-DIS ACCEL(ACCEL1,ACCEL2)
  DETAIL
  LIST(ACTIVE)
  SCOPE(LOCAL)
  MEMBER(DB1D)

```

The output that is displayed depends on the accelerators that you are using. For example, the following output is displayed for IBM DB2 Analytics Accelerator for z/OS V5 and earlier:

```

ACCELERATOR          MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
ACCEL1                DB1D  STARTED    9210    7    5    9
LOCATION=ACCELERATOR1 HEALTHY
DETAIL STATISTICS
  LEVEL = AQT05015
  STATUS = ONLINE
  FAILED REQUESTS = 3
  AVERAGE QUEUE WAIT = 99
  MAXIMUM QUEUE WAIT = 400
  TOTAL NUMBER OF ACTIVE PROCESSORS = 4
  AVERAGE CPU UTILIZATION ON COORDINATOR NODES = 45.00%
  AVERAGE CPU UTILIZATION ON WORKER NODES = 40.00%
  AVERAGE DISK IO UTILIZATION = 10.00%
  NUMBER OF ACTIVE WORKER NODES = 2
  TOTAL DISK STORAGE = 93000 MB
  DISK STORAGE IN USE FOR THIS DB2 SYSTEM = 27 MB
  DISK STORAGE IN USE FOR ALL DB2 SYSTEMS = 27 MB
  TOTAL CPU FOR REQUESTS FOR THIS DB2 SYSTEM = 30 MS
  TOTAL CPU FOR DATA MAINTENANCE FOR THIS DB2 SYSTEM = 110 MS
  TOTAL CPU FOR REPLICATION FOR THIS DB2 SYSTEM = 0 MS
DISPLAY ACCEL REPORT COMPLETE
DSN9022I ) DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION

```

Related reference:

 [Support Portal: IBM DB2 Analytics Accelerator for z/OS](#)

Chapter 25. -DISPLAY ARCHIVE (DB2)

The DB2 command DISPLAY ARCHIVE displays input archive log information.

Abbreviation: -DIS ARC

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- DISPLAY system privilege
- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



```
▶▶—DISPLAY ARCHIVE————▶▶
```

Output

Message DSNJ322I (DB2 Messages) indicates the beginning of the output of the command.

Usage note

Data sharing members: Although the command ARCHIVE LOG SCOPE(GROUP) or ARCHIVE LOG MODE(QUIESCE) initiates archive processing for all members of a data sharing group, the command DISPLAY ARCHIVE shows information only for the member for which it is issued. To display input archive log information for all members of a data sharing group, enter the command on each member.

Examples

Example: Displaying information about archive log data sets

Issue the following command to display information about archive log data sets that are in use:

```
-DISPLAY ARCHIVE
```

The output is similar to the following output:

```
DSNJ322I -DISPLAY ARCHIVE REPORT FOLLOWS
          COUNT      TIME
          (TAPE UNITS) (MIN,SEC)
DSNZPARG      2        0,00
CURRENT      2        0,00
=====
ADDR STAT CORR-ID  VOLSER DATASET_NAME
03B0 BUSY SHEDDEN  A00001 DSNT2AR1.DT25.D04169.T1328583.A0012701
          RCAL 03RCRSC  MIGRAT DSNT2AR1.DT25.D04169.T1334426.A0012704
A99B BUSY 14DRSTRT ARN690 DSNT2AR1.DT25.D04169.T1346176.A0012705
BDDD BUSY 10LPLALR ARN738 DSNT2AR1.DT25.D04170.T1506437.A0012743
END OF DISPLAY ARCHIVE REPORT.
```

The report shows the following information:

- The subsystem parameter values for MAX RTU (COUNT) and DEALLC PERIOD TIME as recorded in the DSNZPxxx load module
- Current specifications for the COUNT and TIME parameters
- Availability status of allocated archive log data sets
- Volume and data set names that are associated with current archive log read requests

Chapter 26. -DISPLAY BUFFERPOOL (DB2)

The DB2 command DISPLAY BUFFERPOOL displays the current status for one or more active or inactive buffer pools.

Abbreviation: -DIS BPOOL

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

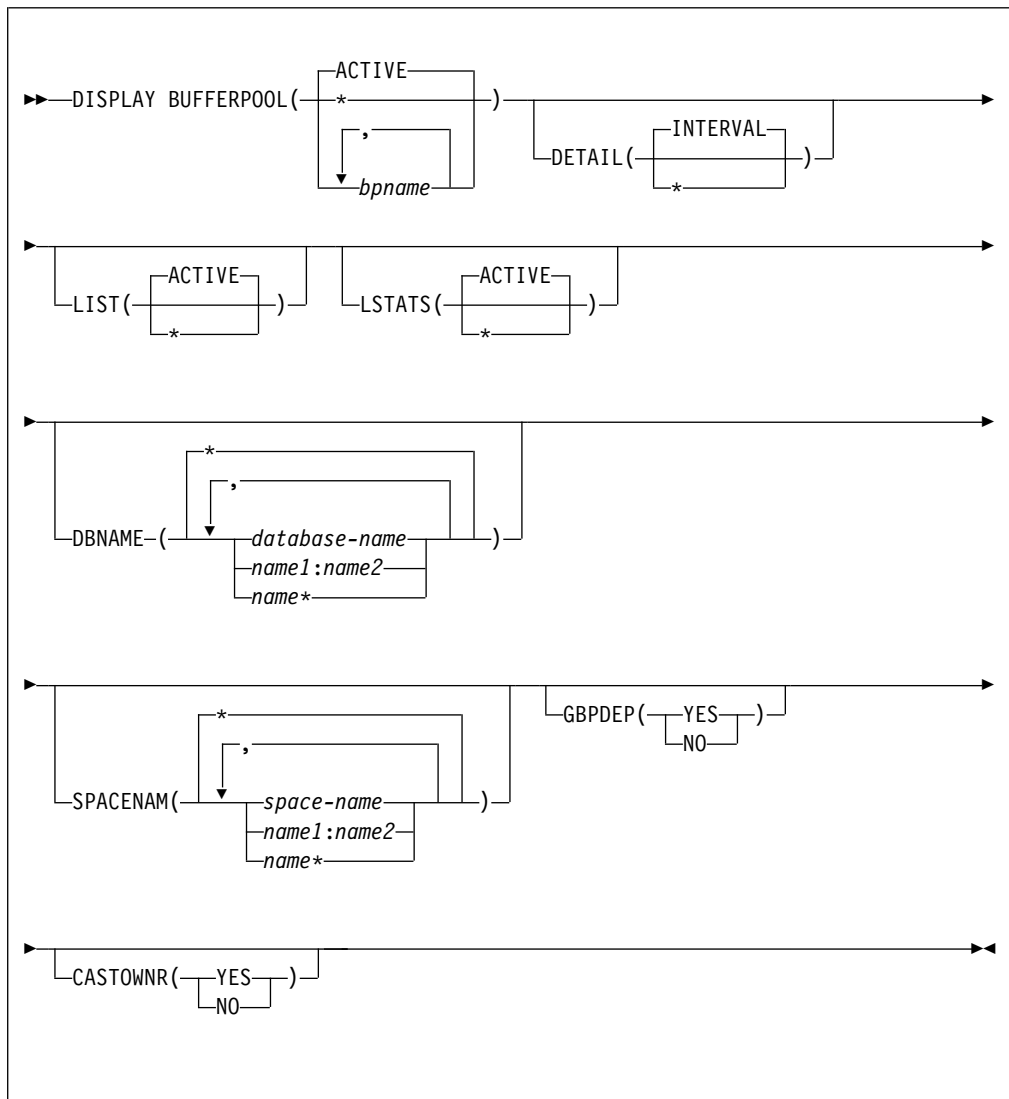
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- DISPLAY system privilege
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

(ACTIVE)

Displays the current buffer pool status for all active buffer pools.

(*)

Displays the current buffer pool status for all active or inactive buffer pools.

(bpname)

Names the buffer pool for which current status is to be displayed.

- 4-KB page buffer pools are named BP0 through BP49
- 8-KB page buffer pools are named BP8K0 through BP8K9
- 16-KB page buffer pools are named BP16K0 through BP16K9
- 32-KB page buffer pools are named BP32K through BP32K9

DETAIL

Produces a detail report for one or more buffer pools. If `DETAIL` is not specified, a summary report is produced.

(INTERVAL)

Requests statistics accumulated since the last incremental display, or since the buffer pool was first activated if no previous incremental display exists.

(*)

Requests statistics accumulated since the buffer pool was first activated.

LIST

Lists the open index spaces and table spaces associated with the buffer pools included in the report. Basic information is provided for non-data-sharing systems while more detail is provided if data sharing is active.

(ACTIVE)

Restricts the list of open index spaces and table spaces to those that are currently in use.

(*)

Requests a list of all open index spaces and table spaces, whether currently in use or not.

LSTATS

Lists data set statistics for the open index spaces and table spaces associated with the buffer pools included in the report. The statistics displayed are incremental since the last time they were displayed.

(ACTIVE)

Restricts the list statistics to those data sets that are currently in use.

The **default** is **ACTIVE** when LIST is not specified or if LIST is specified with no parameter. If LIST is specified with a parameter and LSTATS has no parameter, the parameter specified for LIST is used for LSTATS.

(*)

Includes statistics for all open index spaces and table spaces, whether currently in use or not.

DBNAME

Specifies which databases are included in the LIST display and the LSTATS display. If you specify DBNAME without LIST, LIST(ACTIVE) is assumed.

Abbreviation: DBN

(database-name, ...)

Identifies one or more databases to be included in the LIST and LSTATS displays. *database-name* can have any of the forms in the following list. In the list, *name1* and *name2* represent strings of one- to eight-characters. *name* represents a string of one- to eight-characters.

Form Displays the status of...

name1 The database *name1*

name1:name2

All databases with names from *name1* to *name2* in a sorted list of database names.

*name** All databases whose names begin with the string *name*

(*)

Displays information on all databases that match the LIST specification. This is the default.

SPACENAM

Specifies which table spaces or index spaces within the specified databases to

include in the LIST display and the LSTATS display. If you use SPACENAM without DBNAME, DBNAME(*) is assumed.

Abbreviation: SPACE

(*) Displays information about all table spaces and index spaces of the specified databases. This is the default.

(space-name, ...)

Identifies one or more spaces to be included in the LIST and LSTATS displays. You can write *space-name* like *database-name* to designate:

- The name of a single table space or index space
- A range of names
- A partial name followed by a pattern-matching character

GBPDEP

Indicates whether to restrict the list of data sets to those that are group buffer pool dependent. This option is not valid if this is a non-data sharing DB2.

(YES)

Restricts the list of page sets to those that are group buffer pool dependent (GBP-dependent). An index space or table space is GBP-dependent if either of these conditions are true:

- Inter-DB2 R/W interest exists in it.
- Changed pages from it exist in the group buffer pool that have not yet been written to disk.

(NO)

Restricts the list of page sets to those that are non-group buffer pool dependent.

CASTOWNR

Indicates whether to restrict the list of data sets to those for which this DB2 member is the castout owner. This option is not valid if this is a non-data sharing DB2.

(YES)

Restricts the list of page sets for which this DB2 member is the castout owner.

(NO)

Restricts the list of page sets for which this DB2 member is not the castout owner.

Output

Message DSNB401I (DB2 Messages) indicates the beginning of the output of the command.

Examples

Example: Displaying a summary buffer pool report

A summary report is the default report if the DETAIL option is not specified. The following example shows a summary report that can be produced by the command:

```
-DISPLAY BUFFERPOOL(BP0) LIST(*) DBNAME(DSN8*)
```

The output is similar to this output:

```

DSNB401I - BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 20
DSNB402I - BUFFERPOOL SIZE = 3000 BUFFERS  AUTOSIZE = YES
          ALLOCATED      = 3000          TO BE DELETED   = 0
          IN-USE/UPDATED =    0
          BUFFERS ACTIVE          = 3000
DSNB406I - PGFIX ATTRIBUTE -
          CURRENT                = NO
          PENDING                 = YES
          PAGE STEALING METHOD     = LRU
DSNB404I - THRESHOLDS -
          VP SEQUENTIAL           = 80
          DEFERRED WRITE          = 85
          VERTICAL DEFERRED WRT   = 80, 0
          PARALLEL SEQUENTIAL     = 50
          ASSISTING PARALLEL SEQT  = 0
DSNB9022I - DSNB1CMD '-DISPLAY BUFFERPOOL' NORMAL COMPLETION

```

Example: Displaying a detailed buffer pool report

A detail report can be generated that includes all summary report information and additional buffer pool related statistics. The following example shows a detail report that can be produced by the command:

```
-DISPLAY BUFFERPOOL(BP0) DETAIL
```

The output looks similar to this output:

```

DSNB401I - BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 10
DSNB402I - BUFFERPOOL SIZE = 3000 BUFFERS  AUTOSIZE = YES
          ALLOCATED      = 3000          TO BE DELETED   = 0
          IN-USE/UPDATED =  200
          BUFFERS ACTIVE = 3000
DSNB406I - PGFIX ATTRIBUTE -
          CURRENT                = NO
          PENDING                 = YES
          PAGE STEALING METHOD     = LRU
DSNB404I - THRESHOLDS -
          VP SEQUENTIAL           = 80
          DEFERRED WRITE          = 50
          VERTICAL DEFERRED WRT   = 10, 0
          PARALLEL SEQUENTIAL     = 50
          ASSISTING PARALLEL SEQT  = 0

DSNB409I - INCREMENTAL STATISTICS SINCE 10:32:48 OCT 23, 1993

DSNB411I - RANDOM GETPAGE      =      230 SYNC READ I/O (R) =      180
          SEQ.  GETPAGE        =      610 SYNC READ I/O (S) =      20
          DMTH HIT              =           0 PAGE-INS REQ   =      40
          SEQUENTIAL            =      200 VPSEQT HIT       =           0
          RECLASSIFY            =           0

DSNB412I - SEQUENTIAL PREFETCH -
          REQUESTS              =           0 PREFETCH I/O   =           0
          PAGES READ            =           0

DSNB413I - LIST PREFETCH -
          REQUESTS              =           0 PREFETCH I/O   =           0
          PAGES READ            =           0

DSNB414I - DYNAMIC PREFETCH -
          REQUESTS              =           0 PREFETCH I/O   =           0
          PAGES READ            =           0

DSNB415I - PREFETCH DISABLED -
          NO BUFFER              =           0 NO READ ENGINE =           0
DSNB420I - SYS PAGE UPDATES    =           0 SYS PAGES WRITTEN =           0
          ASYNC WRITE I/O       =           0 SYNC WRITE I/O  =           0
          PAGE-INS REQ          =           0
DSNB421I - DWT HIT              =           0 VERTICAL DWT HIT  =           0

```

```

DSNB440I - PARALLEL ACTIVITY -
          PARALL REQUEST =          0  DEGRADED PARALL =          0
DSNB9022I - DSNB1CMD '-DISPLAY BUFFERPOOL' NORMAL COMPLETION
    
```

Example: Listing open table spaces and index spaces that are associated with a buffer pool

With the summary or detail report, you can list open table spaces and index spaces associated with the buffer pool. You can also request a display of statistics for each listed table space and index space. An example of a report generating this information could be produced by the command:



```
-DISPLAY BUFFERPOOL(BP0) LIST LSTATS
```

The output looks similar to this output:


```

DSNB401I - BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 3
DSNB402I - BUFFERPOOL SIZE = 3000 BUFFERS  AUTOSIZE = YES
          ALLOCATED      = 3000          TO BE DELETED   = 0
          IN-USE/UPDATED = 200
          BUFFERS ACTIVE = 3000
DSNB406I - PGFIX ATTRIBUTE -
          CURRENT          = NO
          PENDING          = YES
          PAGE STEALING METHOD = LRU
DSNB404I - THRESHOLDS -
          VP SEQUENTIAL    = 80
          DEFERRED WRITE   = 50
          VERTICAL DEFERRED WRT = 10, 0
          PARALLEL SEQUENTIAL = 50
          ASSISTING PARALLEL SEQT = 0
DSNB455I - SYNCHRONOUS I/O DELAYS -
          AVERAGE DELAY    = 22
          MAXIMUM DELAY    = 35
          TOTAL PAGES      = 23
DSNB9022I - DSNB1CMD '-DISPLAY BUFFERPOOL' NORMAL COMPLETION
    
```

Related tasks:

-  [Monitoring buffer pools \(DB2 Administration Guide\)](#)
-  [Monitoring and tuning buffer pools by using online commands \(DB2 Performance\)](#)

Related information:

-  [DSNB401I \(DB2 Messages\)](#)

DSNB401I **BUFFERPOOL NAME** *bp-name*,
BUFFERPOOL ID *bp-id*, **USE COUNT**
use-count

Explanation: This message displays output from the DISPLAY BUFFERPOOL command. For each buffer pool, this message marks the beginning of multiple lines of information about that buffer pool. Some lines in the output have their own message numbers or alphanumeric identifiers to assist with identification.

The first line (DSNB401I) contains the following information:

bp-name

The external name of the buffer pool. *bp-name* can be one of the following values: BP0, BP1 - BP49, BP8KB, BP16KB, BP32K, BP32K1 - BP32K9.

bp-id The internal identifier for the buffer pool. *bp-id* can be one of the following values: 0 - 49, 80 - 89.

use-count

The number of open table spaces or index spaces that use this buffer pool. A value of zero means that the buffer pool is inactive.

The remaining output for each buffer pool consists of one or more of the following sections, in the indicated order:

- DSNB402I: Information about allocation status
- DSNB404I: Threshold information
- DSNB406I: PGFIX and PGSTEAL information
- DSNB409I: Start time of the statistics interval for DETAIL(INTERVAL)

- DSNB410I: Start time of the statistics interval for DETAIL(*)
- DSNB411I: Page read statistics
- DSNB412I: Sequential prefetch statistics
- DSNB413I: List prefetch statistics
- DSNB414I: Dynamic prefetch statistics
- DSNB415I: Prefetch statistics
- DSNB420I: Page write statistics
- DSNB421I: Page-write threshold statistics
- DSNB440I: Parallel activity statistics
- DSNB441I: LPL activity statistics
- DSNB453I: Cached and changed page statistics
- DSNB455I: Synchronous I/O delay statistics
- DSNB456I: Asynchronous I/O delay statistics
- DSNB460I: Page set and partition list information (for a data-sharing environment)
- DSNB464I: Page set and partition list information (for a non-data-sharing environment)
- DSNB466I: Page set and partition statistics
- DSNB467I: Data set statistics

If information cannot be reported, one or more of the following messages is returned:

- DSNB408I: No detail statistics available
- DSNB459I: Open failure for a data set
- DSNB463I: No objects matched selection criteria

The DISPLAY BUFFERPOOL output ends with one of the following messages:

- DSN9022I: Normal completion
- DSNB499I: Display terminated because of insufficient space

DSNB402I: Information about allocation status:

The basic buffer pool information is followed by a description of the allocation status of the buffer pool.

BUFFER POOL SIZE = *pool-size* BUFFERS

The user-specified buffer pool size.

AUTOSIZE = *autosize*

The buffer pool AUTOSIZE attribute that is applicable to the current allocation of the buffer pool.

YES The buffer pool uses Workload Manager (WLM) services, if available, to automatically adjust the size of the buffer pool. The size is adjusted based on dynamic monitoring of the workload goals and the storage that is available on the system.

NO The buffer pool does not use WLM services for automatic sizing adjustment of buffer pools.

ALLOCATED = *allocated-buffers*

The number of allocated buffers in an active buffer pool.

TO BE DELETED = *delete-buffers*

The number of buffers to be deleted in an active buffer pool because of pool contraction.

IN-USE/UPDATED = *current-buffers*

The number of currently active buffers in the buffer pool. These buffers cannot be stolen.

DSNB404I: Threshold information:

The information about allocation status is followed by information about the user-modifiable thresholds for the buffer pool.

```
DSNB404I - THRESHOLDS -
                VP SEQUENTIAL           = vpseq
                DEFERRED WRITE          = dwt
                VERTICAL DEFERRED WRT   = dwv1,dwv2
                PARALLEL SEQUENTIAL     = vppseqt
                ASSISTING PARALLEL SEQT  = vpxpseqt
```

vpseq The sequential steal threshold for the virtual pool, expressed as a percentage of the total buffer pool size.

spseq The sequential steal threshold for the simulated pool, expressed as a percentage of the total simulated buffer pool size.

dwt The free-buffer deferred write threshold for the buffer pool, expressed as a percentage of the total buffer pool size.

dwv1 The vertical deferred write threshold for the buffer pool, expressed as a percentage of the total buffer pool size.

dwv2 The vertical deferred write threshold for the buffer pool, expressed as an absolute number of buffers. *dwv2* is used to determine the threshold only if *dwv1* is 0 and *dwv2* is non-zero. Otherwise, *dwv1* is used to determine the threshold.

vppseqt The sequential threshold for parallel query processing, expressed as a percentage of the virtual sequential steal threshold. When the threshold is set to 0, parallel query processing is not supported.

vpxpseqt The assisting parallel sequential threshold, expressed as a percentage of the sequential threshold for parallel query processing. Use the *vpxpseqt* value to control how much buffer resource is used when this DB2 member is assisting another member of the group in parallel processing. When the threshold is set to 0, this buffer pool is not used to assist other data-sharing members in processing a query. In a non-data-sharing environment, this value is ignored.

DSNB406I: PGFIX and PGSTEAL information:

DSNB401I

The DISPLAY BUFFERPOOL output includes a description of the PGFIX and PGSTEAL attributes for the buffer pool.

CURRENT = *current-pgfix*

The value of the page fix (PGFIX) attribute for the current allocation of the buffer pool.

- YES** The buffer pool is fixed in real storage for the long term.
- NO** The buffer pool is fixed in real storage only during an I/O operation.

PENDING = *pending-pgfix*

The value of the PGFIX attribute that is to be applied for the next allocation of the virtual buffer pool.

- YES** The buffer pool is to be fixed in real storage for the long term.
- NO** The buffer pool is to be fixed in real storage only during an I/O operation.

PAGE STEALING METHOD = *current-pgsteal*

The page stealing method (PGSTEAL) that is in use for the buffer pool.

- LRU** The least recently used (LRU) algorithm is used to manage page stealing. This method is the default value.
- FIFO** The first-in, first-out (FIFO) algorithm is used to manage page stealing.
- NONE** No page stealing occurs. Objects that use this buffer pool are kept resident.

Related information:

Fixing a buffer pool in real storage (DB2 Performance)

Choosing a page-stealing algorithm (DB2 Performance)

DSNB408I: No detail statistics available:

DSNB408I indicates that, although you specified the DETAIL option on the DISPLAY BUFFERPOOL command, no detail statistics are available for the requested buffer pool. The statistics are not available because the pool has not been activated since DB2 started.

BUFFER POOL *bp-name*

The name of the buffer pool.

DSNB409I: Start time of the statistics interval for DETAIL(INTERVAL):

When you specify the DETAIL(INTERVAL) option, the output includes the start time of the interval for which the statistics were accumulated.

DSNB409I - INCREMENTAL STATISTICS SINCE *base-time*

base-time

The start time. This value is either the time of the previous incremental display or, if no previous incremental display exists, the time that the buffer pool was first activated.

The format is *hh:mm:ss month dd, yyyy*

hh:mm:ss

The time expressed as hour:minutes:seconds.

month

An alphanumeric abbreviation for the month. For example, a value of OCT means October.

dd

The day of the month.

yyyy

The year.

DSNB410I: Start time of the statistics interval for DETAIL(*):

When you specify the DETAIL(*) option, the output includes the start time of the interval for which the statistics were accumulated.

DSNB410I - CUMULATIVE STATISTICS SINCE *base-time*

base-time

The start time. This value is the time the buffer pool was first activated.

The format is *hh:mm:ss month dd, yyyy*

hh:mm:ss

The time expressed as hour:minutes:seconds.

month

An alphanumeric abbreviation for the month. For example, a value of OCT means October.

dd

The day of the month.

yyyy

The year.

DSNB411I: Page read statistics:

When you specify the DETAIL option, the output includes the page read statistics for the buffer pool.

DSNB411I - RANDOM GETPAGE = *rgp*
SYNC READ I/O (R) = *srr*
SEQ. GETPAGE = *sgp*
SYNC READ I/O (S) = *srs*
DMTH HIT = *dmt*
PAGE-INS REQ = *pir*
SEQUENTIAL = *seq*
VPSEQT HIT = *vsh*
RECLASSIFY = *rcy*

RANDOM GETPAGE = *rgp*

The number of nonsequential GETPAGE requests.

SYNC READ I/O (R) = *srr*

The number of synchronous read I/O operations for nonsequential GETPAGE requests.

SEQ. GETPAGE = *sgp*
The number of sequential GETPAGE requests.

SYNC READ I/O (S) = *srs*
The number of synchronous read I/O operations for sequential GETPAGE requests.

DMTH HIT = *dmt*
The number of times that the data management threshold was reached.

PAGE-INS REQ = *pir*
The number of page-in operations that are required for read I/O.

SEQUENTIAL = *seq*
The number of buffers on the sequential least-recently-used (SLRU) chain.

VPSEQT HIT = *vsh*
The number of times that the size of the SLRU chain reached the sequential steal threshold (the VPSEQT value) for the buffer pool.

RECLASSIFY = *rcy*
A statistic that is used by IBM for serviceability.

Related information:

Buffer pool thresholds that you can change (DB2 Performance)

Chapter 11, “-ALTER BUFFERPOOL (DB2),” on page 61

DSNB412I: Sequential prefetch statistics:

When you specify the DETAIL option, the output includes the sequential prefetch statistics for the buffer pool.

REQUESTS = *pft*
The number of times that sequential prefetch was requested.

PREFETCH I/O = *pio*
The number of sequential prefetch read I/O operations.

PAGES READ = *ppf*
The number of pages that are read because of sequential prefetch.

Related information:

Sequential prefetch (PREFETCH='S') (DB2 Performance)

DSNB413I: List prefetch statistics:

When you specify the DETAIL option, the output includes the list prefetch statistics for the buffer pool.

REQUESTS = *pft*
The number of times that list prefetch was requested.

PREFETCH I/O = *pio*
The number of list prefetch read I/O operations.

PAGES READ = *ppf*
The number of pages that are read because of list prefetch.

Related information:

List prefetch (PREFETCH='L' or 'U') (DB2 Performance)

DSNB414I: Dynamic prefetch statistics:

When you specify the DETAIL option, the output includes the dynamic prefetch statistics for the buffer pool.

REQUESTS = *pft*
The number of times that dynamic prefetch was requested.

PREFETCH I/O = *pio*
The number of dynamic prefetch read I/O operations.

PAGES READ = *ppf*
The number of pages that are read because of dynamic prefetch.

Related information:

Dynamic prefetch (PREFETCH='D') (DB2 Performance)

DSNB415I: Prefetch statistics:

When you specify the DETAIL option, the output includes the prefetch statistics for the buffer pool.

NO BUFFER = *pdf*
The number of times that prefetch was disabled for one of the following reasons:

- The buffer pool reached the prefetch disabled threshold (90% full).
- A user disabled prefetch by setting the VPSEQT threshold for the buffer pool to zero.

NO READ ENGINE = *ree*
The number of times that prefetch was disabled because an asynchronous read processor was not available.

Related information:

Read operations and prefetch I/O (DB2 Performance)

DSNB420I: Page write statistics:

When you specify the DETAIL option, the output includes the page write statistics for the buffer pool.

SYS PAGE UPDATES = *pages-updated*
The number of buffer updates.

DSNB401I

SYS PAGES WRITTEN = *pages-written*

The number of pages that are written to disk.

ASYNCR WRITE I/O = *async-writes*

The number of asynchronous write I/O operations.

SYNC WRITE I/O = *sync-writes*

The number of synchronous write I/O operations.

PAGE-INS REQ = *page-ins*

The number of page-ins that are required for write I/O.

DSNB421I: Page-write threshold statistics:

When you specify the **DETAIL** option, the output includes the page-write threshold statistics for the buffer pool.

DWT HIT = *dwt*

The number of times that the deferred write threshold was reached.

VERTICAL DWT HIT = *vdw*

The number of times that the vertical deferred write threshold was reached.

Related information:

Buffer pool thresholds that you can change (DB2 Performance)

DSNB440I: Parallel activity statistics:

When you specify the **DETAIL** option, the output includes statistics about parallel activities for the buffer pool.

PARALLEL REQUEST = *tpa*

The total number of negotiations with the buffer pool for the requested number of sequential prefetch streams.

DEGRADED PARALLEL = *dpa*

The total number of times that the negotiation resulted in the degraded mode of parallel operations.

DSNB441I: LPL activity statistics:

When you specify the **DETAIL** option, the output includes statistics about LPL activity for the buffer pool.

DSNB441I - LPL ACTIVITY -
PAGES ADDED = *pages*

pages The total number of pages for all page sets that are added to the logical page list (LPL) in this buffer pool. This value is equal to the number of DSNB250E messages that are written to the system log after the most recent execution of the **DISPLAY BUFFERPOOL** command with the **DETAIL** option.

Related information:

DSNB250E (DB2 Messages)

Displaying the logical page list (DB2 Administration Guide)

Characteristics of pages that are in error (DB2 Administration Guide)

DSNB453I: Cached and changed page statistics:

The output includes the number of cached pages and changed pages in the buffer pool for a data set if all of the following conditions are true:

- You specified the **LSTATS** option on the **DISPLAY BUFFERPOOL** command.
- The buffer pool is an active buffer pool.
- The number of cached and changed pages are not zero.

The relevant table space or index space is identified in either line DSNB464I or line DSNB465I. The data set is identified in line DSNB466I.

CURRENT = *vcount*

The number of cached pages in the virtual pool for the data set. This value is the number of buffers that contain pages for the data set in the buffer pool.

MAX = *mvcount*

The maximum number of cached pages in the virtual pool for the data set since the last **DISPLAY BUFFERPOOL** command with the **LSTATS** option was issued.

CHANGED = *ccount*

The number of changed pages in the virtual pool for the data set. This value is the number of buffers that were changed in the buffer pool for the data set.

MAX = *mccount*

The maximum number of changed pages in the virtual pool for the data set since the last **DISPLAY BUFFERPOOL** command with the **LSTATS** option was issued.

DSNB455I: Synchronous I/O delay statistics :

The output includes synchronous I/O delay statistics if all of the following conditions are true:

- You specified the **LSTATS** option on the **DISPLAY BUFFERPOOL** command.
- The buffer pool is an active buffer pool.
- The values of the synchronous I/O delay statistics are not zero.

These synchronous I/O delay statistics are reported for a data set for an open table space or index space that is associated with the buffer pool. The values that are listed are the statistics that were gathered since the last display for the data set.

The relevant table space or index space is identified in either line DSNB464I or line DSNB465I. The relative data set within the table space or index space is identified in line DSNB466I.

```
DSNB455I - SYNCHRONOUS I/O DELAYS -
          AVERAGE DELAY      = avd
          MAXIMUM DELAY      = mxd
          TOTAL PAGES        = tpg
```

- avd* The average I/O delay in milliseconds for pages in the data set.
- mxd* The maximum I/O delay in milliseconds for pages in the data set.
- tpg* The total number of pages that are read or written for the data set.

Related information:

Read operations and prefetch I/O (DB2 Performance)

DSNB456I: Asynchronous I/O delay statistics :

The output includes asynchronous I/O delay statistics if all of the following conditions are true:

- You specified the LSTATS option on the DISPLAY BUFFERPOOL command.
- The buffer pool is an active buffer pool.
- The values of the asynchronous I/O delay statistics are not zero.

These asynchronous I/O delay statistics are reported for a data set for an open table space or index space that is associated with the buffer pool. The values that are listed are the statistics that were gathered since the last display for the data set.

The relevant table space or index space is identified in either line DSNB464I or line DSNB465I. The relative data set within the table space or index space is identified in line DSNB466I.

```
DSNB456I - ASYNCHRONOUS I/O DELAYS -
          AVERAGE DELAY      = avd
          MAXIMUM DELAY      = mxd
          TOTAL PAGES        = tpg
          TOTAL I/O COUNT    = tio
```

- avd* The average I/O delay in milliseconds for pages in the data set.
- mxd* The maximum I/O delay in milliseconds for pages in the data set.
- tpg* The total number of pages that are read or written for the data set.
- tpg* The total number of I/O operations that are issued for the data set.

DSNB459I: Open failure for a data set:

Message DSNB459I indicates that a previous attempt to access a data set failed because of an allocation or open error. This message is displayed only when you specify

the LIST option on the DISPLAY BUFFERPOOL command for an active buffer pool and this error condition occurs.

The relevant table space or index space is identified in either line DSNB464I or line DSNB465I.

csect-name

The name of the control section that issued the message.

DATASET = *dsn*

The data set number. This value is the relative data set number within a table space or index space.

Other data set and buffer pool information might be displayed in subsequent messages.

DSNB460I: Page set and partition list information (for a data-sharing environment):

When you specify the LIST option, the output includes information about page sets and partition lists.

The message output begins with the introductory text PAGE SET/PARTITION LIST INFORMATION. The introductory text is followed by column headers and multiple lines of information.

The output contains the following columns, in the indicated order:

DATABASE

The name of the database. This field is blank when the line provides information about the same database as the preceding line or lines.

SPACE NAME

The name of the table space. This field is blank when the line provides information about the same table space as the preceding line or lines.

INST The instance number.

PART One of the following values:

- The partition number.
- For a simple table space or simple index space: a blank.
- For non-partitioned indexes on a partitioned table space: the logical partition number preceded by the character L (for example, L01).

TS IX The type of object: either TS for table space or IX for index space.

GBP DEP

An indicator of group buffer pool (GBP) dependency. The indicator can be either of the following values:

Y The page set or partition is GBP-dependent.

N The page set or partition is not GBP-dependent.

DSNB401I

MEMBER NAME

The name of the member that the detail line pertains to.

CASTOUT OWNER

An indicator of whether the member is the castout owner. The indicator can be either of the following values:

- Y** The member is the castout owner.
Blank The member is not the castout owner.

USE COUNT

The number of active claimers or drainers for the page set or partition for the member.

P-LOCK STATE

The P-lock state that the member currently holds. The state can be any of the following values:

- IS** R/O interest. Other members have interest in this page set or partition. The page set or partition is GBP-dependent.
- IX** R/W interest. Other members have interest in this page set or partition. The page set or partition is GBP-dependent.
- S** R/O interest. Other members might be reading the page set or partition. The page set or partition is not GBP-dependent.
- SIX** R/W interest. Other members might be reading the page set or partition. The page set or partition is GBP-dependent.
- NSU** R/W interest. The page set or partition is GBP-dependent.
- X** R/W interest. No other members are accessing the page set or partition. The page set or partition is not GBP-dependent.
- US** A temporary state that can be held by a restarting DB2 when "waiting for retained locks" is enabled.

number A number for use as a diagnostic aid. A number is displayed only when a DISPLAY BUFFERPOOL command encounters an undefined lock state.

If the DISPLAY BUFFERPOOL LIST command finds more than 255 lines of output to display, the output is presented in multiple sections. Each section is a new instance of message DSNB460I with the addition of "(CONTINUED)" in the message heading:

```
DSNB460I @ (CONTINUED)
```

Message DSNB460I is issued in a data-sharing environment. In a non-data-sharing environment, message DSNB464I is issued instead.

If no information is available, message DSNB460I is followed by message DSNB463I.

DSNB463I: No objects matched selection criteria:

Message DSNB463I indicates that DB2 did not find any page sets or partitions that matched the selection criteria.

For example, this message is displayed for DIS BPOOL(BP0) GBPDEP(Y) if DB2 did not find any page sets or partitions that are group-buffer-pool dependent (GBP-dependent).

When message DSNB463I is returned, the DISPLAY BUFFERPOOL command terminates normally.

If you expected to see a list of page sets or partitions, review and correct the syntax of the DISPLAY BUFFERPOOL command as needed. Consider adding or changing the filter keywords to obtain a list of page sets or partitions.

DSNB464I: Page set and partition list information (for a non-data-sharing environment):

When you specify the LIST option, the output includes information about page sets and partition lists.

The message output begins with the introductory text PAGE SET/PARTITION LIST INFORMATION. The introductory text is followed by column headers and multiple lines of information.

```
DSNB464I @  
PAGE SET/PARTITION LIST INFORMATION  
TS USE  
DATABASE SPACE NAME INST PART IX COUNT  
=====
```

DATABASE

The name of the database. This field is blank when the line provides information about the same database as the preceding line or lines.

SPACE NAME

The name of the table space. This field is blank when the line provides information about the same table space as the preceding line or lines.

INST The instance number.

PART One of the following values:

- The partition number.
- For a simple table space or simple index space, a blank.
- For non-partitioned indexes on a partitioned table space, the logical partition number preceded by the character L (for example, L01).

TS IX The type of object: either TS for table space or IX for index space.

USE COUNT

The number of active claimers or drainers for the page set or partition for the member.

If the DISPLAY BUFFERPOOL LIST command finds more than 255 lines of output to display, the output is presented in multiple sections. Each section is a new instance of message DSNB464I with the addition of "(CONTINUED)" in the message heading:

```
DSNB464I @ (CONTINUED)
```

Message DSNB464I is issued in a non-data-sharing environment. In a data-sharing environment, message DSNB460I is issued instead.

If no information is available, message DSNB464I is followed by message DSNB463I.

DSNB466I: Page set and partition statistics:

When you specify the LSTATS option, the output includes statistics for data sets. Those statistics are introduced by message DSNB466I.

DSNB467I: Data set statistics:

When you specify the LSTATS option, the output includes statistics about data sets for objects that are associated with active buffer pools.

STATISTICS FOR *object-type*

The type of object: either TABLE SPACE or INDEX SPACE.

database-name

The name of the database.

space-name

The name of the table space or index space.

instance-number

The instance number of the table space or index space.

DATA SET #: *set-number*

The relative data set number within the table space or index space.

USE COUNT: *application-count*

The number of applications that have a claim or drain on the page set or partition.

This message is followed by one or more of the following messages: DSNB453I, DSNB455I, or DSNB456I.

Related information:

Claims and drains (DB2 Performance)

DSN9022I: Normal completion:

The DISPLAY THREAD output normally ends with message DSN9022I.

Related information:

DSN9022I (DB2 Messages)

DSNB499I: Display terminated because of insufficient space:


If the DISPLAY BUFFERPOOL output is too long, the output ends with message DSNB499I. This message indicates that the command was unable to obtain storage for more messages. This situation occurs only for a long display request, such as a detail display for many buffer pools.


The output is truncated.

Reissue the DISPLAY BUFFERPOOL command, and specify a smaller number of buffer pools.

System action: Processing continues.

Related tasks:

 Monitoring buffer pools (DB2 Administration Guide)

 Monitoring and tuning buffer pools by using online commands (DB2 Performance)

Related reference:

-DISPLAY BUFFERPOOL (DB2)

-ALTER BUFFERPOOL (DB2)

Chapter 27. -DISPLAY DATABASE (DB2)

The DB2 command DISPLAY DATABASE displays status information about DB2 databases.

The DISPLAY DATABASE command displays information about the status of the following objects:

- DB2 databases
- Table spaces
- Tables in segmented table spaces
- XML table spaces
- LOB table spaces
- Index spaces within a database
- Indexes on auxiliary tables
- Partitions of partitioned table spaces
- Partitions of index spaces

DISPLAY DATABASE RESTRICT indicates if a table space, index space, or partition is in any pending status. Use the ADVISORY option without the RESTRICT option to display any objects that are in an advisory pending status, such as the informational COPY-pending status or auxiliary warning advisory status.

In a data sharing environment, the command can be issued from any DB2 subsystem in the group that has access to the database.

Abbreviation: -DIS DB

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group

Authorization

The DISPLAY system privilege allows you to display status information for any database. The resulting display lists those databases for which the primary authorization ID or any of the secondary authorization IDs has the DISPLAYDB privilege. Error messages are produced for those databases specified over which the set of privileges does not include one of the following privileges or authorities:

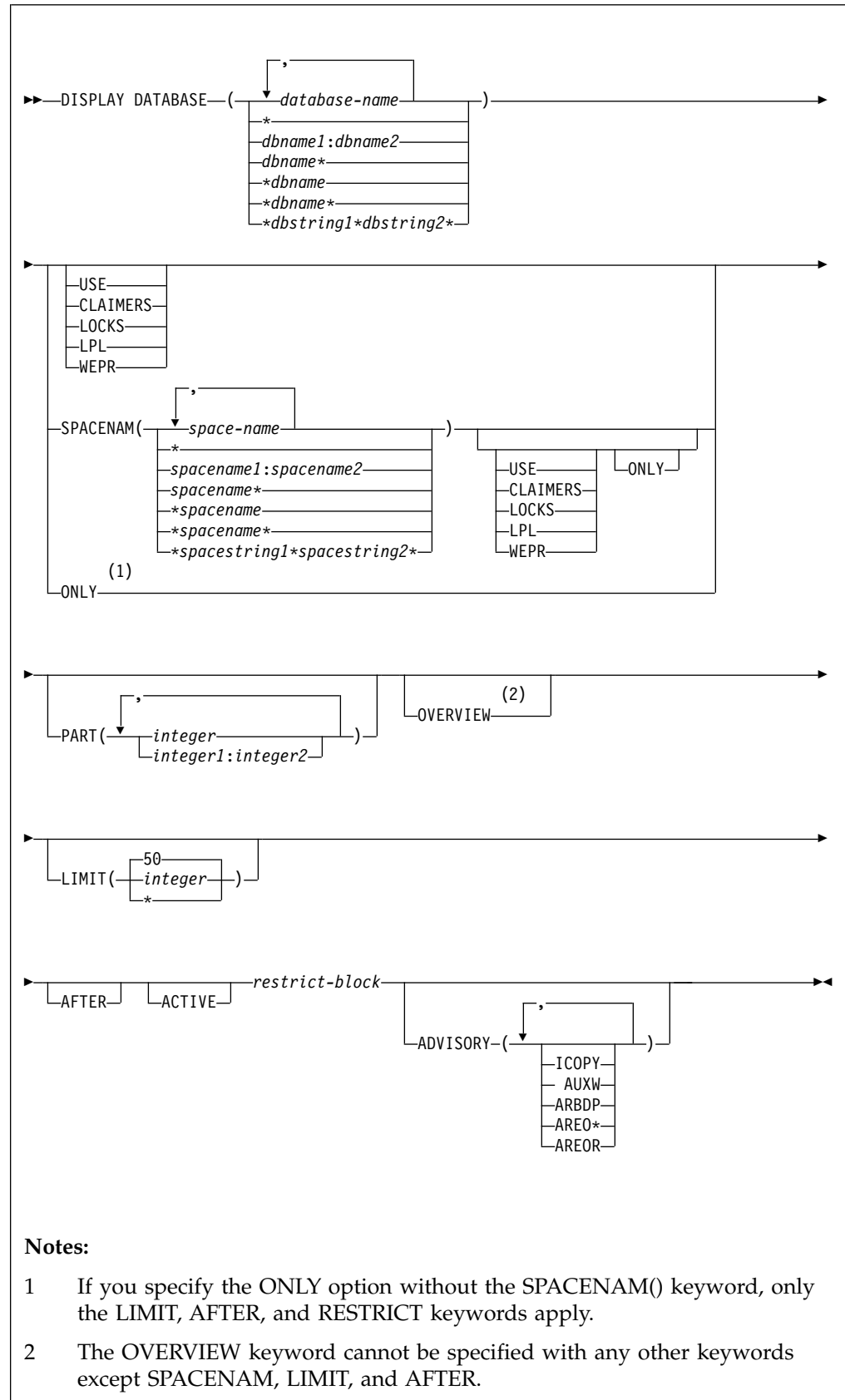
- DISPLAYDB privilege
- DISPLAY privilege
- DBMAINT authority
- DBCTRL authority
- DBADM authority
- System DBADM authority
- SYSOPR authority

- SYSCTRL authority
- SYSADM authority

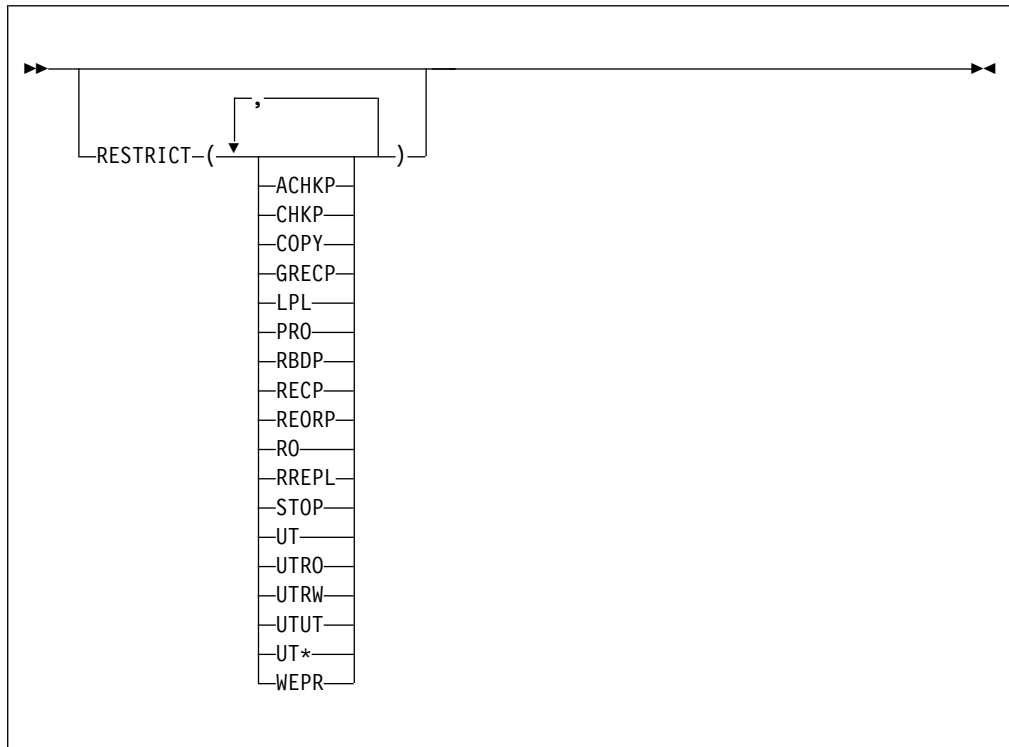
For implicitly created databases, the database privilege or authority can be held on the implicitly created database or on DSNDB04. If the DISPLAY DATABASE command is issued on specific table spaces or index spaces in an implicitly created database, ownership of the table spaces is sufficient to display status information about them. This means that the owner can display information about an implicitly created table space or index space if the command explicitly specifies that table space or index space name.

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



restrict block:



Option descriptions

(database-name, ...)

Identifies one or more databases whose status is to be displayed.

(*)

Displays information on all databases that are defined to the DB2 subsystem for which the privilege set of the process has the required authorization.

dbname and *dbstring* can have any of the forms listed in the following table (where *dbname1* and *dbname2* represent any strings of from one to eight characters, and *dbname* represents any string of from one to seven characters).

Table 13. Forms of *dbname* and *dbstring*

Form	Displays the status of...
<i>dbname1:dbname2</i>	All databases whose names, in UNICODE, are between <i>dbname1</i> and <i>dbname2</i> inclusive
<i>dbname*</i>	All databases whose names begin with the string <i>dbname</i>
<i>*dbname</i>	All databases whose names end with the string <i>dbname</i>
<i>*dbname*</i>	All databases whose names contain the string <i>dbname</i>
<i>*dbstring1*dbstring2*</i>	All databases whose names contain the strings <i>dbstring1</i> and <i>dbstring2</i>

SPACENAM

Specifies what space to display. If you use SPACENAM, you must also specify the corresponding database name. If (*) is used to specify multiple databases, SPACENAM(*) can be specified to display all objects in these databases.

Abbreviation: SPACE, SP

(*space-name, ...*)

Lists one or more spaces whose status is to be displayed. You can write *space-name* like *database-name* to designate:

- The name of a single table space or index space
- A range of names
- A partial name, including a beginning or ending pattern-matching character (*), a pattern-matching character between two strings, or any combination of these, with the following exception. Consecutive pattern-matching characters (*) are not allowed, and you cannot specify two pattern-matching characters in the middle of a keyword string.

(*)

Displays information about all table spaces and index spaces of the specified database.

spacename and *spacestring* can have any of the forms listed in the following table (where *spacename1* and *spacename2* represent any strings of from one to eight characters, and *spacename* represents any string of from one to seven characters).

Table 14. Forms of *spacename* and *spacestring*

Form	Displays the status of...
<i>spacename1:spacename2</i>	All table spaces or index spaces whose names, in UNICODE, are between <i>spacename1</i> and <i>spacename2</i> inclusive
<i>spacename*</i>	All table spaces or index spaces whose names begin with the string <i>spacename</i>
<i>*spacename</i>	All table spaces or index spaces whose names end with the string <i>spacename</i>
<i>*spacename*</i>	All table spaces or index spaces whose names contain the string <i>spacename</i>
<i>*spacestring1*spacestring2*</i>	All table spaces or index spaces whose names contain the strings <i>spacestring1</i> and <i>spacestring2</i>

USE

Displays the following information:

- The applications and subsystems of the database or space that have internal DB2 resources allocated
- The applications and subsystems of the database or space on whose behalf locks for the space are held or waited for
- The connection IDs, correlation IDs, and authorization IDs for all applications allocated to spaces and partitions whose statuses are displayed
- The LUWID and location of any remote threads accessing the local database

CLAIMERS

Displays the following information:

- The claims on all table spaces, index spaces and partitions whose statuses are displayed
- The LUWID and location of any remote threads accessing the local database
- The connection IDs, correlation IDs, and authorization IDs for all applications allocated to spaces whose statuses are displayed

- The logical partitions that have logical claims and their associated claims
- The agent token for the claimer, if the claimer is local

CLAIMERS overrides both LOCKS and USE. If you specify CLAIMERS, any references to LOCKS or USE are ignored.

LOCKS

Displays the following information:

- The applications and subsystems on whose behalf locks are held, waited on, or retained for the database or space
- The transaction locks for all table spaces, tables, index spaces and partitions whose statuses are displayed
- The connection IDs, correlation IDs, and authorization IDs for all applications allocated to spaces whose statuses are displayed
- The LUWID and location of any remote threads accessing the local database
- The drain locks for a resource held by running jobs
- The logical partitions that have drain locks and the drain locks that are associated with them
- The retained locks for a resource
- The page set or partition physical locks (P-locks) for a resource
- The agent token for the lock holder, if the lock holder is local

LOCKS overrides USE. If both LOCKS and USE are specified, USE is ignored.

LPL

Displays logical page list entries.

WEPR

Displays write error page range information.

ONLY

Displays information about the specified object.

without SPACENAM() keyword

Displays only database information. DB2 does not display information for the spaces within the database you specified with the DISPLAY DATABASE command. If you specify ONLY, the following keywords are valid:

- RESTRICT
- LIMIT
- AFTER

with SPACENAM() keyword

Displays the table spaces or indexes that have information requested by the DISPLAY DATABASE command. If you specify SPACENAM() ONLY, you must also specify one of the following keywords:

- USE
- CLAIMERS
- LOCKS
- LPL
- WEPR

DB2 displays tables with table locks when you specify both the LOCKS and ONLY keywords.

PART (*integer*, ...)

Indicates the partition number of one or more partitions whose status is to be displayed. The *integer* specified must identify a valid partition number for the corresponding space name and database name. *integer* can be written to designate one of the following values:

- A list of one or more partitions
- A range of all partition numbers that collate greater than or equal to *integer1* and less than or equal to *integer2*
- A combination of lists and ranges

OVERVIEW

Displays each object in the database on its own line, providing an easy way to see all objects in the database.

OVERVIEW limits the display to only the space names and space types that exist in the specified databases. The number of parts is displayed for any partitioned spaces.

The OVERVIEW keyword cannot be specified with any other keywords except SPACENAM, LIMIT, and AFTER.

LIMIT

Limits the number of messages to be displayed by the command.

(*integer*)

Is the maximum number of messages that are to be displayed. The **default** is **50**. The maximum number of messages that can be displayed is limited by the space available.

(*)

Limits the display to the space available.

AFTER

Displays the following information:

- If only a database name is used, AFTER continues the display of all other databases whose names collate greater than that name.
- If SPACENAM and a table space or index space name are used, AFTER continues the display to all other table spaces or index spaces in the same database whose names collate greater than that name.

AFTER cannot be used with more than one database name, table space name, index space name, with any pattern-matching character (*) within the database name, or with the **SPACENAM()** keyword.

ACTIVE

Limits the display to table spaces or index spaces that have had internal DB2 resources allocated to applications and are in a started state or to databases that contain such spaces.

Abbreviation: A

Default: Using neither ACTIVE nor RESTRICT displays information on all databases defined to DB2.

RESTRICT

Limits the display to databases, table spaces, or indexes in a restricted status. This includes those page sets that have logical page list entries. Specifying one or more keywords further limits the display to the named objects only.

Abbreviation: RES

Use of a database is restricted if the database is in any of the following situations:

- It is started for read-only processing.
- It is started for read-or-replication-only processing.
- It is started for utility-only processing.
- It is stopped.

Use of a table space or index space is restricted if the table space or index space is in any of the following situations:

- It is in one of the three situations listed previously.
- It is being processed by a utility.
- It is in COPY-pending, CHECK-pending, RECOVER-pending, group buffer pool RECOVER-pending, auxiliary CHECK-pending, or REORG-pending status.
- It contains a page error range.
- It contains pages in the logical page list (LPL).

Specify one or more of the following keywords to limit objects that are to be listed.

ACHKP

Displays objects in the auxiliary warning advisory status.

CHKP Display objects that are in CHECK-pending status.

COPY Display objects that are in COPY-pending status.

GRECP

Displays objects that are in group buffer pool RECOVER-pending status.

LPL Displays logical page list entries.

PRO Displays table space partitions that are in Persistent Read Only (PRO) restricted status.

RBDP Displays index objects that are in REBUILD- or RECOVER-pending status. This includes the restricted statuses RBDP, RBDP*, PSRBDP, LPL, and WEPR.

RECP Displays objects that are in RECOVER-pending status, including the restricted status RECP, RECP*, LPL, and WEPR (write error page range).

REORP

Displays objects that are in REORG-pending status.

RO Displays objects that are in read-only mode.

RREPL

Displays objects that are in read-or-replication-only mode.

STOP Displays objects that are stopped, including the restricted statuses STOP, STOPE, STOPP, and LSTOP.

UT Displays objects that are in utility access mode.

UTRO Display objects that are serialized for utility access and available for read-only access.

UTRW

Display objects that are serialized for utility access and available for read-write access.

UTUT Displays objects that are serialized for utility access and unavailable.

UT* Displays objects that are in any utility access mode: UT, UTRW, UTRO, or UTUT.

WEPR Displays write error page range information.

ADVISORY

Limits the display to indexes and table spaces to which read-write access is allowed, but for which some action is recommended.

Abbreviation: ADV

Use the DISPLAY DATABASE ADVISORY command without the RESTRICT option to determine when:

- An index space is in the informational COPY-pending (ICOPY) advisory status.
- A base table space or LOB table space is in the auxiliary warning (AUXW) advisory status.
- An index space is in the REBUILD-pending (ARBDP) advisory status.
- An index space is in the REORG (AREO*) advisory status.
- A table space or an index space is in the REORG(AREOR) advisory status.

Specify one or more of the following keywords to limit the objects listed.

AUXW

Displays objects that are in the auxiliary warning advisory status.

ICOPY

Displays objects that are in the informational COPY-pending advisory status.

ARBDP

Displays objects that are in the advisory REBUILD-pending status.

AREO*

Displays objects that are in the advisory REORG-pending status.

AREOR

Displays objects that are in the advisory REORG-pending status.

Usage notes**Displaying DB2 catalog tables:**

You can always display the DB2 catalog tables. However, if a table space in the catalog containing information about user databases or user table spaces is stopped, those databases or table spaces cannot be displayed. Trying to display them will cause an error.

If you issue DISPLAY DATABASE LOCKS on the catalog (DSNDB06), you might see a lock held on SYSDBASE with the correlation ID 020.DBCMD_05 or 020.DBCMD_06. This information indicates the lock that DISPLAY DATABASE itself needs and is normal.

Displaying restricted and advisory status objects:

To display all resources that are in restricted status, you must issue the DISPLAY DATABASE command twice. To display table spaces and indexes in restricted status, use the SPACENAM parameter with RESTRICT. To

display databases in restricted status, do **not** use the SPACENAM parameter. Spaces could be unavailable even if they show RW mode if the database is in restricted status.

To display all resources that are in advisory status, issue the DISPLAY DATABASE ADVISORY command without the RESTRICT option.

Communications Database and Resource Limit Facility:

If the command specifies a table space or index space in the communications database or in the active resource limit facility database, the USE option displays the names of all members of the data sharing group that are using the specified table space or index space. Knowing which other members of the data sharing group might be using these spaces is useful when considering whether to drop table spaces and index spaces in the communications database and the resource limit facility database.

Displaying logical partitions:

If you issue DISPLAY DATABASE with the PART parameter for a logical partition of a type 2 index, DB2 does not display physical claimers and physical locks in the output. Nonpartitioned indexes are displayed with a type of 'IX' and with partition numbers displayed either as 'L' followed by a four-digit number, or as 'L*.' When the information for all the logical partitions is the same, partition numbers are displayed as 'L*.' If there is unique information to be displayed for a particular logical partition, L is used with that partition number to represent the logical part.

Displaying databases for declared temporary tables:

DISPLAY DATABASE can display information about databases that are created with the AS TEMP option and the associated table spaces, but does not display information for declared temporary tables or index spaces that the database contains.

Displaying data-partitioned secondary indexes (DPSIs):

DISPLAY DATABASE can display information about data-partitioned secondary indexes. DPSIs are displayed with a type of 'IX'. The partition number is displayed as 'D' followed by the four-digit partition number, ranging from 0001 to 4096.

Displaying XML table spaces:

DISPLAY DATABASE can display information about the status of XML table spaces. XML table spaces are displayed with a type of 'XS'.

Displaying clone table information:

The information about base table objects and their clones is automatically displayed if a clone table exists. The information for both base objects and clones is displayed because clone objects can have different states than their base counterparts. DISPLAY indicates the current base and clone data set instance numbers.

Base table objects that have been cloned are noted with the character 'B' in the TYPE column of the DISPLAY output. Cloned objects are noted with the character 'C' in the TYPE column. Immediately following the 'B' or 'C' character is the data set instance number. The data set number is always a '1' or '2'. All the base table objects have the same data set instance number. All of the clone table objects have the same instance number, which is different than the base table objects' instance number. Data set instance number associations change during each data exchange.

You can also query the INSTANCE column of the SYSTABLESPACE catalog table to determine the current instance number associated with a particular base table.

If you drop a cloned table of a base table with a instance number '2' then the DISPLAY output of the base table still indicates that a clone once existed. The DISPLAY command output still shows 'B2' in the TYPE column to indicate that the base object is using instance number '2'. If you drop a cloned table and the base object instance number is '1' then the DISPLAY command outputs information that does not indicate that a clone ever existed. There is no 'B' character or instance number in the TYPE column.

Output

Message DSNT361I (DB2 Messages) indicates the beginning of the output of the command.

Examples

Example: Displaying information about a single table space

The following command displays information about table space TBS33 in database CB3. The USE option causes *connection-name* (CONNID), *correlation-id* (CORRID), and *authorization-ID* (USERID) information to be displayed.

```
-DISPLAY DATABASE(CB3) SPACENAM(TBS33) USE
```

The output is similar to this output:

```
DSNT360I - *****
DSNT361I - * DISPLAY DATABASE SUMMARY
          * GLOBAL USE
DSNT360I - *****
DSNT362I - DATABASE = CB3 STATUS = RW
          DBD LENGTH = 4028

DSNT397I -
NAME     TYPE PART  STATUS          CONNID  CORRID  USERID
-----
TBS33   TS    0001 RW           LSS001  DSN2SQL  SYSADM
        -THRU 0004
TBS33   TS

***** DISPLAY OF DATABASE CB3 ENDED *****
DSN9022I . DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
```

Example: Display information about database locks

The following command display information about table space TBS33 in database CB3. The LOCKS option displays lock information for table spaces and tables specified; LUWIDs and locations of any remote threads; and *connection-name*, *correlation-id*, and *authorization ID* information.

```
-DISPLAY DATABASE(CB3) SPACENAM(TBS33) LOCKS
```

The output is similar to this output:

```
DSNT360I - *****
DSNT361I - * DISPLAY DATABASE SUMMARY
          * GLOBAL LOCKS
DSNT360I - *****
DSNT362I - DATABASE = CB3 STATUS = RW
          DBD LENGTH = 4028

DSNT397I -
NAME     TYPE PART  STATUS          CONNID  CORRID  LOCKINFO
```

```

-----
TBS33  TS    0001 RW
      -THRU  0003
TBS33  TS    0004 RW          LSS004  DSN2SQL    H-IS,S,C
TBS33  TS    0004 RW          LSS005  DSN2SQL    H-IS,S,C
TBS33  TS
***** DISPLAY OF DATABASE CB3          ENDED          *****

```

Example: Displaying information about claimers

The following command displays information about table space TBS33 in database CB3. The CLAIMERS option displays claim types and durations; LUWIDs and locations of any remote threads; and *connection-name*, *correlation-id*, and *authorization ID* information.

```
-DISPLAY DATABASE(CB3) SPACENAM(TBS33) CLAIMERS
```

The output is similar to this output:

```

DSNT360I - *****
DSNT361I - *  DISPLAY DATABASE SUMMARY
          *  GLOBAL CLAIMERS
DSNT360I - *****
DSNT362I - DATABASE = CB3 STATUS = RW
          DBD LENGTH = 4028

DSNT397I -
NAME     TYPE PART  STATUS          CONNID  CORRID  CLAIMINFO
-----
TBS33   TS    0001 RW
      -THRU  0003
TBS33   TS    0004 RW          LSS001  DSN2SQL  (RR,C)
TBS33   TS    0004 RW          LSS001  DSN2SQL  (WR,C)
TBS33   TS
***** DISPLAY OF DATABASE CB3          ENDED          *****

```

Example: Displaying information about locks in a data sharing environment

The following command displays information about locks that are held for a table space. The database is in a data sharing environment. The application that is identified as LSS001 on member DB1G has locked partitions 1 and 2. LSS002 on member DB2G has locked partitions 1 and 3. Partition 4 has no locks held on it.

```
-DISPLAY DATABASE(DSN8D10A) SPACENAM(TSPART) LOCKS
```

The output is similar to this output:

```

NAME     TYPE PART  STATUS          CONNID  CORRID  LOCKINFO
-----
TSPART   TS    0001 RO          LSS001  DSN2SQL  H-IS,P,C
-
-          MEMBER NAME DB1G
TSPART   TS    0001 RO          LSS001  DSN2SQL  H-S,PP,I
-
-          MEMBER NAME DB1G
TSPART   TS    0001 RO          LSS002  DSN2SQL  H-IS,P,C
-
-          MEMBER NAME DB2G
TSPART   TS    0001 RO          LSS002  DSN2SQL  H-S,PP,I
-
-          MEMBER NAME DB2G
TSPART   TS    0002 RW          LSS001  DSN2SQL  H-IS,P,C
-
-          MEMBER NAME DB1G
TSPART   TS    0002 RW          LSS001  DSN2SQL  H-S,PP,I
-
-          MEMBER NAME DB1G
TSPART   TS    0003 RW          LSS002  DSN2SQL  H-IS,P,C
-
-          MEMBER NAME DB2G
TSPART   TS    0003 RW          LSS002  DSN2SQL  H-S,PP,I
-
-          MEMBER NAME DB2G
TSPART   TS    0004 RW
TSPART   TS

```

If DB2 cannot selectively lock the partitions, it must lock all of the partitions and the display looks similar to the following output. The LOCKINFO field shows a value of S, indicating that this is a table space lock. If partitions are held in different statuses, those statuses are listed below the table space locks.

NAME	TYPE	PART	STATUS	CONNID	CORRID	LOCKINFO
TSPART	TS			LSS001	DSN2SQL	H-IS,S,C
-			MEMBER NAME	DB1G		
TSPART	TS			LSS002	DSN2SQL	H-IS,S,C
-			MEMBER NAME	DB2G		
TSPART	TS	0001	RO			H-S,PP,I
-			MEMBER NAME	DB1G		
TSPART	TS	0002	RW			H-S,PP,I
-			MEMBER NAME	DB2G		
TSPART	TS	0003	RW			H-S,PP,I
-			MEMBER NAME	DB2G		
TSPART	TS	0004	RW			
TSPART	TS					

Example: Displaying information about table spaces with entries in the logical page list

The following command displays information about table spaces in database DSNDB01 that have entries in the logical page list. The LIMIT parameter limits the number of messages displayed to the space available.

```
-DB1G DISPLAY DATABASE(DSNDB01) SPACENAM(*) LIMIT(*) LPL
```

The output is similar to this output:

```
*****
DSNT361I -DB1G * DISPLAY DATABASE SUMMARY
              * GLOBAL LPL
DSNT360I -DB1G
*****
DSNT362I -DB1G DATABASE = DSNDB01 STATUS = RW
              DBD LENGTH = 8000
DSNT397I -DB1G
NAME      TYPE PART STATUS          LPL PAGES
-----
DBD01    TS      RW,LPL,GRECP 000001,000004,00000C,000010
-----
BP1TS    TS      RW, LPL      00000002
SPT01    TS      RW
SCT02    TS      RW
SYSLGRNG TS      RW
SYSUTILX TS      RW
SYSLGRNX TS      RW,LPL,GRECP 000000-FFFFFF
DSNSCT02 IX      RW
DSNSPT01 IX      RW
DSNSPT02 IX      RW
DSNLUX01 IX      RW
DSNLUX02 IX      RW
DSNLLX01 IX      RW
DSNLLX02 IX      RW
***** DISPLAY OF DATABASE DSNDB01 ENDED *****
DSN9022I -DB1G DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
```

Example: Displaying lock information for partitions in a table space when DB2 cannot do selective partition locking

Suppose that DB2 is unable to selectively lock the partitions of table space TSPART, which is in database DSN8D10A. When you specify the following command, two applications are accessing TSPART, and the partitions have different statuses.

```
-DB1G DISPLAY DATABASE(DSN8D10A) SPACE(TSPART) PART(1,4) LOCKS
```

DB2 displays the locks as table space locks, as shown in the following output:

NAME	TYPE	PART	STATUS	CONNID	CORRID	LOCKINFO
TSPART	TS			LSS001	DSN2SQL	H-IS,S,C
TSPART	TS			LSS002	DSN2SQL	H-IS,S,C
TSPART	TS	0001	RO			
TSPART	TS	0004	RW			

Example: Displaying lock information for partitions in a table space when DB2 can do selective partition locking

Suppose that you have executed the ALTER TABLESPACE statement on table space TSPART so that TSPART is now defined with LOCKPART YES. LOCKPART YES causes DB2 to do selective partition locking on TSPART. When you issue the following command, two applications are accessing TSPART. The application identified by connection ID LSS001 has locked partitions 1 and 2. The application identified by connection ID LSS002 has locked partitions 1 and 3.

```
-DB1G DISPLAY DATABASE(DSN8D10A) SPACE(TSPART) PART(1:4) LOCKS
```

DB2 displays the locks as partition locks, as shown in the following output:

NAME	TYPE	PART	STATUS	CONNID	CORRID	LOCKINFO
TSPART	TS	0001	RO	LSS001	DSN2SQL	H-IS,P,C
TSPART	TS	0001	RO	LSS002	DSN2SQL	H-IS,P,C
TSPART	TS	0002	RW	LSS001	DSN2SQL	H-IS,P,C
TSPART	TS	0003	RW	LSS002	DSN2SQL	H-IS,P,C
TSPART	TS	0004	RW			

Example: Displaying information about table spaces and index spaces that are in a restrictive status

The following command displays information about all table spaces and index spaces in the range of databases from DBKD0101 to DBKD0106 that are in a restrictive status. The LIMIT parameter limits the number of messages that are displayed to the available space.

```
-DISPLAY DATABASE(DBKD0101,DBKD0103) SPACENAM(*) RESTRICT LIMIT(*)
```

The output is similar to this output:

```
DSNT360I - *****
DSNT361I - * DISPLAY DATABASE SUMMARY
          * RESTRICTED
DSNT360I - *****
DSNT362I - DATABASE = DBKD0101 STATUS = RW
          DBD LENGTH = 4028
DSNT397I -
NAME      TYPE PART STATUS          PHYERRLO PHYERRHI CATALOG PIECE
-----
TLKD0101 TS      RW,RESTP
IUKD011A IX      RW,RESTP
IXKD011B IX      RW,RESTP
```

Example: Displaying information about table spaces that are in an auxiliary warning status or informational status

The following command displays information about all table spaces that have the auxiliary warning advisory status (AUXW), and all index spaces that are in informational COPY-pending status (ICOPY) in database DBIQUQ01.

```
-DISPLAY DATABASE(DBIQUQ01) SPACENAM(*) LIMIT(*) ADVISORY
```

The output is similar to this output:

```

DSNT360I - *****
DSNT361I - * DISPLAY DATABASE SUMMARY
          * ADVISORY
DSNT360I - *****
DSNT362I - DATABASE = DBIQUQ01 STATUS = RW
          DBD LENGTH = 8066
DSNT397I -
NAME      TYPE PART STATUS          PHYERRLO PHYERRHI CATALOG PIECE
-----
TPIQUQ01 TS      0001 RW,AUXW
          -THRU 0004
TPIQUQ01 TS
IAIQUQ01 IX          RW,ICOPY
IAIQUQ02 IX          RW,ICOPY
IAIQUQ03 IX          RW,ICOPY
IAIQUQ04 IX          RW,ICOPY
IPIQUQ01 IX      0001 RW,ICOPY
          -THRU 0004
IPIQUQ01 IX
IUIQUQ03 IX          RW,ICOPY
IXIQUQ02 IX          RW,ICOPY
***** DISPLAY OF DATABASE DBIQUQ01 ENDED *****
DSN9022I - DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION

```

Example: Displaying information about all objects in a database

The following command displays a list of all objects in database DB486A. This example shows five objects in the database. TS486A is a table space with four parts and TS486C is a nonpartitioned table space. IX486A is a nonpartitioned index for table space TS486A, IX486B is a partitioned index with four parts, and IX486C is a nonpartitioned index.

-DISPLAY DATABASE(DB486A) SPACE(*) OVERVIEW

The output is similar to this output:

```

DSNT360I - *****
DSNT361I - * DISPLAY DATABASE SUMMARY
          * GLOBAL OVERVIEW
DSNT360I - *****
DSNT362I - DATABASE = DB486A STATUS = RW
          DBD LENGTH = 4028
DSNT397I -
NAME      TYPE PART
-----
TS486A    TS      0004
TS486A    TS
IX486A    IX      L0004
IX486A    IX      L*
IX486A    IX
IX486B    IX      0004
IX486B    IX
TS486C    TS
IX486C    IX
***** DISPLAY OF DATABASE DB486A ENDED *****
DSN9022I - DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION

```

Example: Displaying table space information when table spaces are stopped

The following command displays information about all table spaces in database DB486B. In table space TS486X, partitions 1 through 6 are stopped. Partition 7 is in UT and COPY status, partition 8 is in STOP status, and partitions 9 and 10 are in RW status.

-DISPLAY DATABASE(DB486B) SPACE(*)

The output is similar to this output:

```

DSNT360I - *****
DSNT361I - *   DISPLAY DATABASE SUMMARY
          *   GLOBAL OVERVIEW
DSNT360I - *****
DSNT362I -   DATABASE = DB486B  STATUS = RW
          DBD LENGTH = 4028
DSNT397I -
NAME      TYPE PART  STATUS          PHYERRLO PHYERRHI CATALOG  PIECE
-----
TS486X   TS    0001 STOP
- THRU   0006
TS486X   TS    0007 UT,COPY
TS486X   TS    0008 STOP
TS486X   TS    0009 RW
- THRU   0010
TS486X   TS
IX486X   IX    L0001 RW
IX486X   IX    L0002 LSTOP
- THRU   L0003
IX486X   IX    L0004 LSTOP
- THRU   L0010
IX486X   IX    L*
IX486Y   IX    0001 RW
- THRU   0010
IX486Y   IX
IX486Z   IX          RW
***** DISPLAY OF DATABASE DB486B  ENDED *****
DSN9022I - DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION

```

Example: Displaying information about all indexes in a database

The following command displays information about all indexes in the DBKD0101 database. INDEX2 contains information to be displayed at a logical level. Partitions 0001 and 0002 of INDEX3 are data-partitioned secondary indexes, as indicated by 'D' in the partition number.

```
-DISPLAY DATABASE(DBKD0101) SPACENAM(INDEX*)
```

The output is similar to this output:

```

DSNT360I - *****
DSNT361I - *   DISPLAY DATABASE SUMMARY
          *   RESTRICTED
DSNT360I - *****
DSNT362I -   DATABASE = DBKD0101 STATUS = RW
          DBD LENGTH = 4028
DSNT397I -
NAME      TYPE PART  STATUS          PHYERRLO PHYERRHI CATALOG  PIECE
-----
INDEX1   IX    0001 RW
- THRU   0002
INDEX1   IX
INDEX2   IX    L*    RW
INDEX3   IX    D0001 RW
- THRU   D0002
INDEX3   IX
INDEX4   IX    L0001 RECP
INDEX4   IX    L0002 RW
INDEX4   IX

```

Example: Displaying information about all table spaces in a database that are in an advisory status

The following command displays information about all table spaces in the DBKD0103 database that are in the advisory REBUILD-pending status (ARBDP) and the advisory REORG-status (AREO*). Limit the number of messages that are displayed to the available space. Assume that you specify the following command:

```
-DISPLAY DATABASE(DBKD0103) SPACENAM(*) LIMIT(*) ADVISORY(ARBDP,AREO*)
```

The output is similar to the following output:

```
DSNT360I - *****
DSNT361I - * DISPLAY DATABASE SUMMARY
          * ADVISORY
DSNT360I - *****
DSNT362I - DATABASE = DBKD0103 STATUS = RW
          DBD LENGTH = 16142
DSNT397I -
NAME     TYPE PART STATUS          PHYERRLO PHYERRHI CATALOG PIECE
-----
PIX      IX  0001 RW,ARBDP,AREO*
          -THRU 0007
PIX      IX
```

Example: Displaying information about table space partitions

The following command displays information about table space DB2TSP in database DB2. The PART option includes both lists and ranges to display a very specific set of partitions. The table space underwent a single ROTATE operation before the final partitions were added.

```
-DISPLAY DATABASE(DB2) SPACENAM(DB2TSP) PART(1,2,4:6,9,10:12)
```

The output is similar to the following output:

```
DSNT360I - *****
DSNT361I - * DISPLAY DATABASE SUMMARY
          * GLOBAL
DSNT360I - *****
DSNT362I - DATABASE = DB2 STATUS = RW
          DBD LENGTH = 4028
DSNT397I -
NAME     TYPE PART STATUS          PHYERRLO PHYERRHI CATALOG PIECE
-----
DB2TSP   TS  0001 RW,AREO*
DB2TSP   TS  0002 RW,AREO*
DB2TSP   TS  0004 RW
          -THRU 0006
DB2TSP   TS  0009 RW
          -THRU 0012
***** DISPLAY OF DATABASE DB2 ENDED *****
DSN9022I - DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
```

Example: Displaying information about a table space that is in informational COPY-pending status

The following command displays information about table spaces that are in an advisory status. The output includes any table spaces that are in the informational COPY-pending status.

```
-DISPLAY DATABASE(DBIQUQ01) SPACENAM(*) ADVISORY
```

The output is similar to this output:

```
DSNT360I - *****
DSNT362I - DATABASE = DBIQUQ01 STATUS = RW
          DBD LENGTH = 8066
DSNT397I -
NAME     TYPE PART STATUS          HYERRLO PHYERRHI CATALOG PIECE
-----
TPIQUQ01 TS  001 RW,AUXW
          -THRU 003
TPIQUQ01 TS  004 RW,ICOPY
TPIQUQ01 TS
```

Example: Displaying information about clone tables

Suppose that some databases whose names begin with MYDB have clone

objects. The following command displays information about base table and clone table objects, as well as objects that are not involved with cloning.

```
-DISPLAY DATABASE(MYDB*) SPACENAM(*) LIMIT(*)
```

The output is similar to this output:

```
DSNT360I - *****
DSNT361I - * DISPLAY DATABASE SUMMARY
          * GLOBAL
DSNT360I - *****
DSNT362I - DATABASE = MYDBX STATUS = STOP
          DBD LENGTH = 4028
DSNT397I -
NAME     TYPE PART  STATUS          PHYERRLO PHYERRHI CATALOG  PIECE
-----
CHKOUT2  TSB1  001    RW
        -THRU  003
CHKOUT2  TSB1
XDEPT1   IXB1  001    RECP
XDEPT1   IXB1  002    RW
        -THRU  003
XDEPT1   IXB1
CHKOUT2  TSC2  001    RO
CHKOUT2  TSC2  002    RW
        -THRU  003
CHKOUT2  TSC2
XDEPT1   IXC2  001    ICOPY
XDEPT1   IXC2  002    RW
        -THRU  003
XDEPT1   IXC2
CHKOUT3  TS     001    RO
        -THRU  003
CHKOUT3  TS
XDEPT3   IX     001    RW
        -THRU  002
XDEPT3   IX     003    RO
XDEPT3   IX
```

Example: Displaying information about all objects that are in restricted status, when table spaces are in the DBETE status

The following command displays all objects that are in a restricted status.

```
-DISPLAY DATABASE(*) SPACENAM(*) RESTRICT
```

When you do not specify specific database names, for objects that are in DBETE status, DISPLAY DATABASE displays the PSID values instead of database names and space names, as shown in the following output:

```
DSNT360I @ *****
DSNT361I @ * DISPLAY DATABASE SUMMARY 371
          * RESTRICTED
DSNT360I @ *****
DSNT362I @ DATABASE = DB65309 STATUS = RW 373
          DBD LENGTH = 8066
DSNT397I @ 374
NAME     TYPE PART  STATUS          PHYERRLO PHYERRHI CATALOG  PIECE
-----
****0002 UN          RW,DBETE,RECP
****0005 UN          RW,DBETE,RECP
****0007 UN          RW,DBETE,RECP
***** DISPLAY OF DATABASE DB65309 ENDED *****
```

Example: Displaying information about specific objects that are in restricted status, when table spaces or index spaces are in the DBETE status

The following command displays objects in database DB65309 that are in a restricted status.


```
-DISPLAY DATABASE(DB65309) SPACENAM(*) RESTRICT
```

When you specify specific database names, for objects that are in DBETE status, DISPLAY DATABASE displays the object names, as shown in the following output:

```
DSNT360I @ *****
DSNT361I @ * DISPLAY DATABASE SUMMARY 395
          * RESTRICTED
DSNT360I @ *****
DSNT362I @ DATABASE = DB65309 STATUS = RW 397
          DBD LENGTH = 8066
DSNT397I @ 398
NAME     TYPE PART  STATUS          PHYERRLO PHYERRHI CATALOG  PIECE
-----
PS65309G UN          RW,DBETE,RECP
IP65309G UN          RW,DBETE,RECP
IX65309G UN    L*    RW,DBETE,RECP
***** DISPLAY OF DATABASE DB65309 ENDED *****
```

Chapter 28. -DISPLAY DDF (DB2)

The DISPLAY DDF command displays information regarding the status and configuration of DDF, as well as statistical information regarding connections or threads controlled by DDF.

Abbreviation: -DIS DDF

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

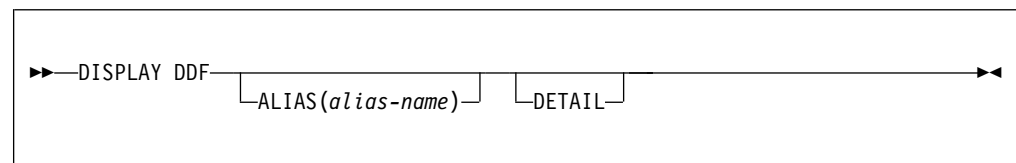
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- DISPLAY privilege
- System DBADM authority
- SYSOPR authority
- SYSTRM authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

ALIAS(alias-name)

Displays information specific to the DDF location alias specified by *alias-name*.

DETAIL

Displays additional statistics and configuration information.

Output

Message DSNL080I (DB2 Messages) indicates the beginning of the output of the command.

Examples

Example: Displaying a detailed DDF report when DDF has not been started

Suppose that DDF has not been started. The following command displays a DDF detail report:

```
-DISPLAY DDF DETAIL
```

The output is similar to this output:

```
DSNL081I STATUS=STOPDQ
DSNL082I LOCATION          LUNAME          GENERICLU
DSNL083I STLEC1            -NONE.SYEC1DB2  -NONE
DSNL084I TCPSPORT=446     SECPORT=0      RESPOR=5001  IPNAME=-NONE
DSNL085I IPADDR=NONE
DSNL086I SQL      DOMAIN=-NONE
DSNL090I DT=A  CONDBAT=    64  MDBAT=    64
DSNL092I ADBAT=    0  QUEDBAT=    0  INADBAT=    0  CONQUED=    0
DSNL093I DSCDBAT=    0  INACONN=    0
DSNL105I DSNLTDDF CURRENT DDF OPTIONS ARE:
DSNL106I PKGREL = COMMIT
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Example: Displaying a summary DDF report when DDF has been started

Suppose that DDF has been started. The following command displays a DDF summary report:

```
-DISPLAY DDF
```

The output is similar to this output:

```
DSNL080I ) DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION          LUNAME          GENERICLU
DSNL083I STLEC1            -NONE          -NONE
DSNL084I TCPSPORT=446     SECPORT=447  RESPOR=5001  IPNAME=XYZ_A
DSNL085I IPADDR=:9.30.178.50
DSNL085I IPADDR=ABCD::91E:B232
DSNL086I SQL      DOMAIN=xyz_ahost.ibm.com
DSNL086I RESYNC  DOMAIN=xyz_ahost.ibm.com
DSNL087I ALIAS          PORT      SECPORT  STATUS
DSNL088I XYZ_S          448      449     STATIC
DSNL089I MEMBER IPADDR=:9.30.178.112
DSNL089I MEMBER IPADDR=ABCD::91E:B270
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Example: Displaying a detailed DDF report when DDF has been started

Suppose that DDF has been started. The following command displays a detailed DDF report:

```
-DISPLAY DDF DETAIL
```

The output is similar to this output:

```
DSNL080I ) DSNLTDDF DISPLAY DDF REPORT FOLLOWS: 211
DSNL081I STATUS=STARTD
DSNL082I LOCATION          LUNAME          GENERICLU
DSNL083I STLEC1            USIBMSY.SYEC1DB2  -NONE
DSNL084I TCPSPORT=446     SECPORT=447  RESPOR=5001  IPNAME=XYZ_A
DSNL085I IPADDR=:9.30.178.50
DSNL085I IPADDR=ABCD::91E:B232
DSNL086I SQL      DOMAIN=xyz_ahost.ibm.com
DSNL086I RESYNC  DOMAIN=xyz_ahost.ibm.com
DSNL087I ALIAS          PORT      SECPORT  STATUS
DSNL088I XYZ_S          448      449     STATIC
DSNL089I MEMBER IPADDR=:9.30.178.112
DSNL089I MEMBER IPADDR=ABCD::91E:B270
DSNL090I DT=A  CONDBAT=    64  MDBAT=    64
DSNL092I ADBAT=    0  QUEDBAT=    0  INADBAT=    0  CONQUED=    0
```

```

DSNL093I DSCDBAT=      0 INACONN=      0
DSNL100I LOCATION SERVER LIST:
DSNL101I WT IPADDR      IPADDR
DSNL102I 64 ::9.30.178.111 ABCD::91E:B26F
DSNL102I  ::9.30.178.112 ABCD::91E:B270
DSNL105I DSNLTDDF CURRENT DDF OPTIONS ARE:
DSNL106I PKGREL = COMMIT
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE

```

Example: Displaying a summary DDF report when location aliases are defined

The following command displays a DDF summary report.

```
-DISPLAY DDF
```

Suppose that the DSNJU003 utility has been used to define location aliases. The output includes information about location aliases, as shown in the following output:

```

DSNL080I - DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION      LUNAME      GENERICLU
DSNL083I STL717A      USIBMSY.SYEC717A  -NONE
DSNL084I TCPPOPT=446  SECPOR=0      RESPOR=5001 IPNAME=-NONE
DSNL085I IPADDR=::9.30.115.135
DSNL085I IPADDR=2002:91E:610:1::5
DSNL086I SQL      DOMAIN=v7ec135.sv1.ibm.com
DSNL087I ALIAS      PORT  SECPOR STATUS
DSNL088I STL717A1      551  0  STATIC
DSNL088I STL717A2      552  0  STATIC
DSNL088I STL717A3      553  0  STATIC
DSNL088I STL717A4      554  0  STATIC
DSNL088I STL717A5      555  0  STATIC
DSNL088I STL717A6      556  0  STATIC
DSNL088I STL717A7      557  0  STATIC
DSNL088I STL717A8      558  0  STATIC
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE

```

Example: Displaying information about a specific location alias when DDF has not been started

The following command displays information about location alias ALIAS01 when DDF has not been started.

```
-DISPLAY DDF ALIAS(ALIAS01)
```

The output is similar to the following output:

```

-DISPLAY DDF ALIAS(ALIAS01)
DSNL080I @ DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL087I ALIAS      PORT  SECPOR STATUS
DSNL088I ALIAS01      5004 5005  STOPD
DSNL089I MEMBER IPADDR=::9.30.114.22
DSNL089I MEMBER IPADDR=2002:91E:610::1
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE

```

Example: Displaying detailed information about a specific location alias when DDF has been started

The following command displays information about location alias alias01 when DDF has been started.

```
-DISPLAY DDF ALIAS(ALIAS01) DETAIL
```

The output is similar to the following output:


```

DSNL080I @ DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL087I ALIAS      PORT  SECPOR STATUS
DSNL088I ALIAS01      5004 5005  STARTD
DSNL089I MEMBER IPADDR=::9.30.114.22
DSNL089I MEMBER IPADDR=2002:91E:610::1

```

```
DSNL096I ADBAT= 100 CONQUED= 1000 TCONS= 1000
DSNL100I LOCATION SERVER LIST:
DSNL101I WT IPADDR IPADDR
DSNL102I 32 ::9.30.114.22 2002:91E:610::1
DSNL102I 32 ::1.2.3.4
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

Related information:

 [DSNL080I \(DB2 Messages\)](#)

Chapter 29. -DISPLAY FUNCTION SPECIFIC (DB2)

The DB2 command DISPLAY FUNCTION SPECIFIC displays statistics about external user-defined functions that DB2 applications access.

Abbreviation: -DIS FUNC SPEC

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the value of the SCOPE option.

Authorization

To run this command, you must use a privilege set of the process that includes one of the following authorities for each function:

- DISPLAY privilege
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

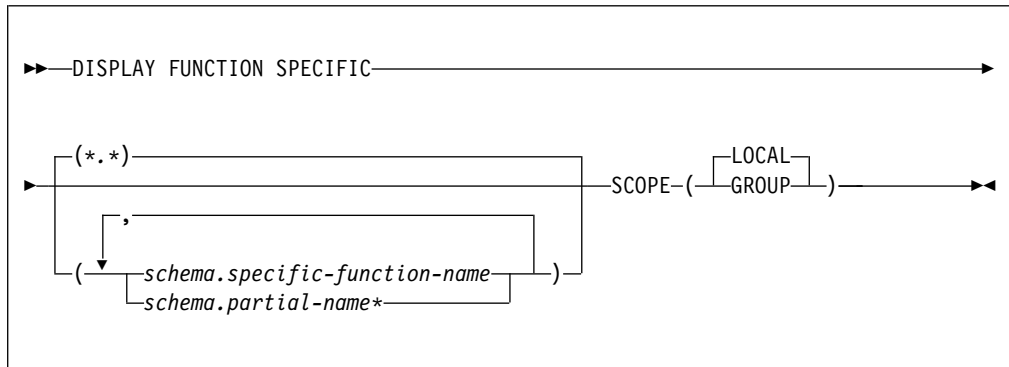
If you specify DISPLAY FUNCTION SPECIFIC *.* or *schema.partial-name **, you must use a privilege set of the process that includes one of the following authorities:

- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

If you are using an external security product to authorize usage of DISPLAY FUNCTION SPECIFIC, define SYSOPR as a user to the external security product for those cases in which DISPLAY FUNCTION SPECIFIC SCOPE(GROUP) runs on a remote system and SYSOPR is used as the authorization ID.

Syntax



Option descriptions

schema.specific-function-name

Displays information for the specific named function in the specified schema. You cannot specify a function name as you can in SQL; you must use the specific name. If a specific name was not specified on the CREATE FUNCTION statement, query SYSIBM.SYSROUTINES for the correct specific name:

```
SELECT SPECIFICNAME, PARM_COUNT  
FROM SYSIBM.SYSROUTINES  
WHERE NAME='function_name'  
AND SCHEMA='schema_name';
```

For overloaded functions, this query can return multiple rows.

*schema.partial-name**

Displays information for a set of functions in the specified schema.

The specific names of all functions in the set begin with *partial-name* and can end with any string, including the empty string. For example, `schema1.ABC*` displays information for all functions with specific names that begin with ABC in `schema1`.

(*.*)

Displays information for all functions that DB2 applications have accessed since the DB2 subsystem was started.

SCOPE

Specifies the scope of the command.

(LOCAL)

Specifies that the display includes information from only the local member.

(GROUP)

Specifies that the display includes information from all members of the data sharing group.

Usage notes

Displaying information for all functions: If you do not specify a partial or specific function name, DB2 displays information for all functions that DB2 applications have accessed since the DB2 subsystem was started.

Built-in functions or user-defined functions that are sourced on another function: This command does not apply to built-in functions or user-defined functions that are sourced on another function.

Displaying SQL functions: SQL functions are displayed in the DISPLAY FUNCTION SPECIFIC output only if you invoke the function in debug mode. In that case, the WLM environment column in the output contains the WLM environment that you specified for debugging when you created the SQL function.

The DISPLAY FUNCTION output shows the statistics on an SQL function as '0' if the function is under the effect of a STOP FUNCTION command.

Output

Message DSNX975I (DB2 Messages) indicates the beginning of the output of the command.

Examples

Example: Displaying information about all user-defined functions in a schema

The following command displays information about all functions in the PAYROLL schema and the HRPROD schema.

```
-DISPLAY FUNCTION SPECIFIC(PAYROLL.*, HRPROD.*)
```

The output is similar to the following output:

```
DSNX975I = DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT FOLLOWS-
```

```
----- SCHEMA=PAYROLL
FUNCTION      STATUS      ACTIVE  QUED  MAXQ  TIMEOUT  FAIL  WLM_ENV
PAYRFNC1     STARTED     0       0     1     0        0    WLMENV1
PAYRFNC2     STOPQUE     0       5     5     3        0    WLMENV1
PAYRFNC3     STARTED     2       0     6     0        0    WLMENV1
USERFNC4     STOPREJ     0       0     1     0        0    WLMENV3

----- SCHEMA=HRPROD
FUNCTION      STATUS      ACTIVE  QUED  MAXQ  TIMEOUT  FAIL  WLM_ENV
HRFNC1       STARTED     0       0     1     0        0    WLMENV2
HRFNC2       STOPREJ     0       0     1     0        0    WLMENV2
DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT COMPLETE
DSN9022I = DSNX9COM '-DISPLAY FUNC' NORMAL COMPLETION
```

Example: Displaying information about selected user-defined functions in a schema

The following command displays information about functions USERFNC2 and USERFNC4 in the PAYROLL schema.

```
-DISPLAY FUNCTION SPECIFIC(PAYROLL.USERFNC2,PAYROLL.USERFNC4)
```

The output is similar to the following output:

```
DSNX975I = DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT FOLLOWS-
```

```
----- SCHEMA=PAYROLL
FUNCTION      STATUS      ACTIVE  QUED  MAXQ  TIMEOUT  FAIL  WLM_ENV
USERFNC2     STOPQUE     0       5     5     3        0    WLMENV3
USERFNC4     STOPREJ     0       0     1     0        0    WLMENV3
DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT COMPLETE
DSN9022I = DSNX9COM '-DISPLAY FUNC' NORMAL COMPLETION
```

Example: Displaying information about stopped user-defined functions

Suppose that you issue the following commands:

```
-STOP FUNCTION SPECIFIC(SYSADM.FN*) ACTION(Queue)
-DISPLAY FUNCTION SPECIFIC(SYSADM.*)
```

The output looks similar to the following output:

DSNX975I = DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT FOLLOWS-

----- SCHEMA=SYSADM

FUNCTION	STATUS	ACTIVE	QUED	MAXQ	TIMEOUT	FAIL	WLM_ENV
FNC1	STOPQUE	0	0	0	0	0	WLMENV1
FNC2	STOPQUE	0	0	0	0	0	WLMENV3

DSNX9DIS FUNCTIONS FN - FN* STOP QUEUE

DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT COMPLETE

DSN9022I = DSNX9COM '-DISPLAY FUNC' NORMAL COMPLETION

Chapter 30. -DISPLAY GROUP (DB2)

The DB2 command DISPLAY GROUP displays information about the data sharing group to which a DB2 subsystem belongs.

DISPLAY GROUP DETAIL displays the DB2 subsystem and group mode (conversion mode, enabling new function mode, or DB2 Version 8 new-function mode or later).

Abbreviation: -DIS GROUP

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group

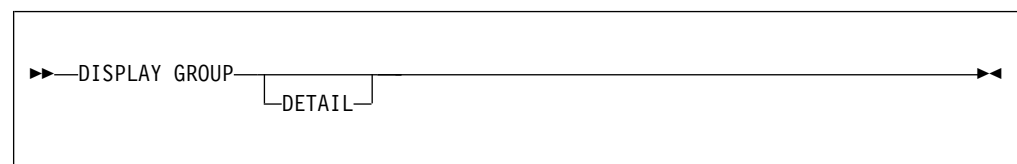
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- DISPLAY privilege
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

DETAIL

Displays information about the parallelism coordinator, parallelism assistant, and the current inline length of the LOB that contains bound package information.

Usage notes

Member status: Message DSN7100I includes information about the XCF status of the members (STATUS in the display output). The status can be ACTIVE, QUIESCED, FAILED, or DEACT.

ACTIVE

Indicates that the DB2 subsystem is active.

FAILED

Indicates that the DB2 subsystem is failed.

DEACT

Indicates that the DB2 subsystem is deactivated. Deactivation is the first step in the process of deleting a data sharing member.

QUIESCED

Indicates a normal quiesced state, as the result of a normal STOP DB2 command.

Q Q (quiesced) can be paired with one or more of the following letters:

I Indoubt or postponed abort units of recovery (URs) are outstanding. This means that retained locks are held.

C The member was shut down without completing castout processing. This event might have occurred because DB2 was started with the LIGHT(YES) option or stopped with the CASTOUT(NO) option, or because an error was encountered while attempting to cast out data from the coupling facility to the disk.

If DB2 encounters errors during castout processing, ensure that no connectivity problems exist between the coupling facility and the processor before restarting DB2.

R Retained information is needed for DB2 to perform resynchronization with one or more remote locations.

When DB2 is restarted, this resynchronization occurs.

ACTIVE

Indicates a normal active state without conditions.

A The member is active, but with the additional conditions. A (active) can be paired with the following letter:

I Indoubt or postponed abort units of recovery (URs) are outstanding. This indicates that retained locks are held.

Using this command in a non-data-sharing environment: DB2 issues the same response, except for information which does not exist: group name, member name, and member ID.

Examples

Example: Displaying information about a data sharing group when the group is in conversion mode

Suppose that you issue the following command against a data sharing group, when all members are in conversion mode, and are all members are on DB2 10:

```
-DB1A DISPLAY GROUP
```

The output is similar to the following output:

```

DSN7100I -DB1A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DSNDB10 ) CATALOG LEVEL(101) MODE(C )
          PROTOCOL LEVEL(2)  GROUP ATTACH NAME(DB10)
-----
DB2
MEMBER  ID  SUBSYS  CMDPREF  STATUS  DB2 SYSTEM  IRLM
-----  -  -  -  -  -  -  -
DB1A    1  DB1A   -DB1A   ACTIVE  101 MVSA   DJ1A  DB1AIRLM
DB1B    2  DB1B   -DB1B   ACTIVE  101 MVSB   DJ1B  DB1BIRLM
DB1C    3  DB1C   -DB1C   ACTIVE  101 MVSC   DJ1C  DB1CIRLM
DB1D    4  DB1D   -DB1D   FAILED  101 MVSD   DJ1D  DB1DIRLM
DB1E    5  DB1E   -DB1E   QUIESCED 101 MVSE   DJ1E  DB1EIRLM
DB1F    6  DB1F   -DB1F   ACTIVE  101 MVSF   DJ1F  DB1FIRLM
DB1G    7  DB1G   -DB1G   ACTIVE  101 MVSG   DJ1G  DB1GIRLM
-----
SCA  STRUCTURE SIZE:    1024 KB, STATUS= AC,   SCA IN USE:    11 %
LOCK1 STRUCTURE SIZE:    1536 KB
NUMBER LOCK ENTRIES:    262144
NUMBER LIST ENTRIES:    7353, LIST ENTRIES IN USE:    0
*** END DISPLAY OF GROUP(DSNDB10 )
DSN9022I -DB1A DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION

```

Example: Displaying information about a data sharing group when the group is in coexistence mode

Suppose that you issue the following command against a data sharing group, when not all members are on DB2 10:

```
-DB1A DISPLAY GROUP
```

The output is similar to the following output:

```

DSN7100I -DB1A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DSNDB10) CATALOG LEVEL(101) MODE(X )
          PROTOCOL LEVEL(2)  GROUP ATTACH NAME(DB10)
-----
DB2
MEMBER  ID  SUBSYS  CMDPREF  STATUS  DB2 SYSTEM  IRLM
-----  -  -  -  -  -  -  -
DB1A    1  DB1A   -DB1A   ACTIVE  101 MVSA   DJ1A  DB1AIRLM
DB1B    2  DB1B   -DB1B   ACTIVE  101 MVSB   DJ1B  DB1BIRLM
DB1C    3  DB1C   -DB1C   ACTIVE  101 MVSC   DJ1C  DB1CIRLM
DB1D    4  DB1D   -DB1D   FAILED  910 MVSD   DJ1D  DB1DIRLM
DB1E    5  DB1E   -DB1E   QUIESCED 910 MVSE   DJ1E  DB1EIRLM
DB1F    6  DB1F   -DB1F   ACTIVE  101 MVSF   DJ1F  DB1FIRLM
DB1G    7  DB1G   -DB1G   ACTIVE  101 MVSG   DJ1G  DB1GIRLM
-----
SCA  STRUCTURE SIZE:    1024 KB, STATUS= AC,   SCA IN USE:    11 %
LOCK1 STRUCTURE SIZE:    1536 KB
NUMBER LOCK ENTRIES:    262144
NUMBER LIST ENTRIES:    7353, LIST ENTRIES IN USE:    0
*** END DISPLAY OF GROUP(DSNDB10 )
DSN9022I -DB1A DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION

```

Example: DISPLAY GROUP in a non-data-sharing environment

Suppose that you issue the following command against a single subsystem on which new function is activated:

```
-DB1A DISPLAY GROUP
```

The output is similar to the following output:

```

DSN7100I -DB1A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(.....) CATALOG LEVEL(1010) MODE(N )
          PROTOCOL LEVEL(...)  GROUP ATTACH NAME(....)
-----
DB2
MEMBER  ID  SUBSYS  CMDPREF  STATUS  DB2 SYSTEM  IRLM
-----  -  -  -  -  -  -  -

```

```

..... 0 DB1A -DB1A ACTIVE 101 MVSA DJ1A DB1AIRLM
-----
*** END DISPLAY OF GROUP(DSNDB10)
DSN9022I -DB1A DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION

```

Example: Displaying detailed information about a data sharing group

You can obtain more information about the data sharing group by using the DISPLAY GROUP command with the DETAIL option. Issue the following command:

```
-DB1A DISPLAY GROUP DETAIL
```

The output is similar to the following output:

```


DSN7100I @ DSN7GCMD 488
*** BEGIN DISPLAY OF GROUP(DSNCAT ) CATALOG LEVEL(101) MODE(NFM )
          PROTOCOL LEVEL(2)  GROUP ATTACH NAME(DSNG)
-----
DB2          DB2 SYSTEM      IRLM
MEMBER  ID  SUBSYS  CMDPREF  STATUS  LVL  NAME      SUBSYS  IRLMPROC
-----
VA1A      1  VA1A    @          ACTIVE  101  DTEC745    PR21    PRLM21
-----
DB2          PARALLEL    PARALLEL MEMBER  COORDINATOR ASSISTANT
-----
DISPLAY SUBGROUP ATTACH INFORMATION FOR GROUP ATTACH DSNG
-----
MEMBERS OF SUBGROUP ATTACH SUBA
  VA1A  VA1B
MEMBERS OF SUBGROUP ATTACH SUBB
  VA1C
-----
SCA  STRUCTURE SIZE:      4096 KB, STATUS= AC,   SCA IN USE:      5 %
LOCK1 STRUCTURE SIZE:      4096 KB
NUMBER LOCK ENTRIES:      524288
NUMBER LIST ENTRIES:      3833, LIST ENTRIES IN USE:      0
*** END DISPLAY OF GROUP(DSNCAT )
DSN9022I @ DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION

```

Output

Message DSN7100I (DB2 Messages) indicates the beginning of the output of the command.

Related reference:

 Obtaining information about the group (DB2 Data Sharing Planning and Administration)

Chapter 31. -DISPLAY GROUPBUFFERPOOL (DB2)

The DB2 command DISPLAY GROUPBUFFERPOOL displays information about the status of DB2 group buffer pools. It can also display related statistics.

Abbreviation

-DIS GBPOOL

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group

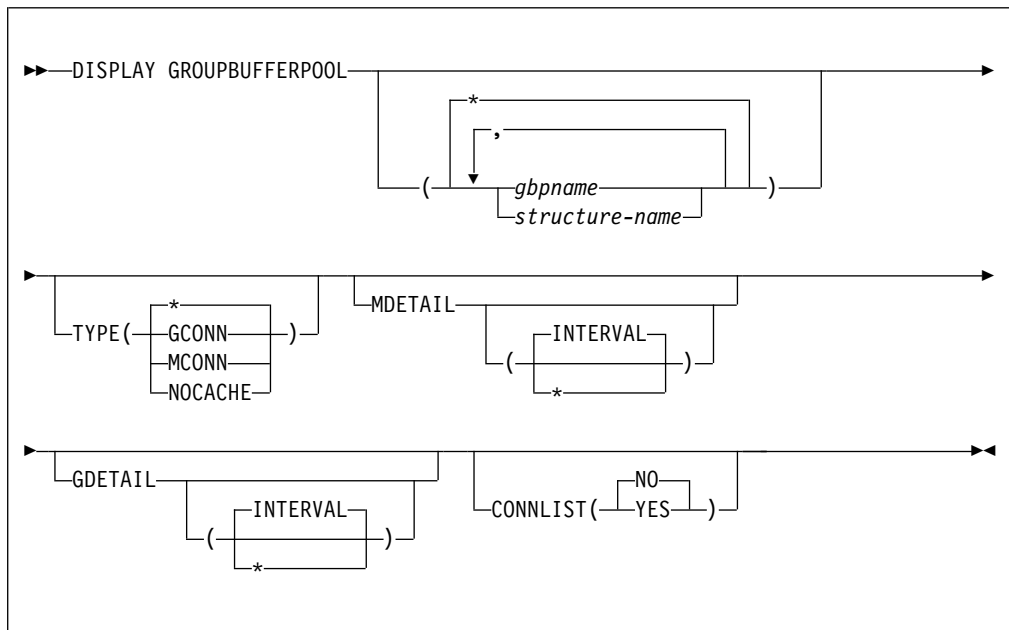
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- DISPLAY system privilege
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option description

(*)

Displays the group buffer pool status for all group buffer pools.

(gbpname)

Names the group buffer pool for which status is to be displayed.

- 4-KB group buffer pools are named GBP0 through GBP49
- 8-KB group buffer pools are named GBP8K0 through GBP8K9
- 16-KB group buffer pools are named GBP16K0 through GBP16K9
- 32-KB group buffer pools are named GBP32K through GBP32K9

(structure-name)

Names the backing coupling facility structure for the group buffer pool. The coupling facility structure name has the following format:

groupname_gbpname

where *groupname* is the DB2 data sharing group name and the underscore (`_`) separates *groupname* and *gbpname* .

TYPE

Indicates the type of group buffer pools (among those that are specified) for which information is displayed.

(*)

All group buffer pools are specified. This is the default.

(GCONN)

Group buffer pools that are currently connected to any member of the data sharing group. The connection can be “active” or “failed-persistent”.

(MCONN)

Group buffer pools that are currently connected to the member to which the command is directed.

(NOCACHE)

Group buffer pools that have the GBPCACHE attribute set to NO.

MDETAIL

Shows a detailed statistical report for the specified group buffer pools, reflecting the member's activity for each group buffer pool. If the member to which the command is directed has never been actively connected to the group buffer pool, no detail report is shown.

(INTERVAL)

Shows incremental statistics. The values displayed are accumulated since the last MDETAIL(INTERVAL) report for this member, if there was one. This is the default.

(*)

Shows cumulative statistics. The values displayed are accumulated since this member first connected to the group buffer pool.

GDETAIL

Shows a detailed statistical report for the specified group buffer pools, reflecting the activity of the entire group for each group buffer pool. If the member to which the command is directed is not actively connected to the group buffer pool, no detail report is shown.

(INTERVAL)

Shows incremental statistics. The values displayed are accumulated since the last GDETAIL(INTERVAL) report, if there was one. This is the default.

(*)

Shows cumulative statistics. The values displayed are accumulated since the group buffer pool was most recently allocated or reallocated.

CONNLIST

Specifies whether a connection list report is shown for the specified group buffer pools, listing the connection names of the subsystems that are currently connected to the group buffer pools and their connection status.

(NO)

Do not show the connection list report.

(YES)

Show the connection list report.

Output

Message DSNB750I (DB2 Messages) indicates the beginning of the output of the command.

Examples

Example: Displaying a group buffer pool summary report

Issue the following command to issue a summary report for group buffer pool GBP29:

```
-DISPLAY GROUPBUFFERPOOL(GBP29)
```

The output is similar to this output. Message DSNB799I is displayed if the group buffer pool is duplexed and the secondary group buffer pool is currently allocated. If a secondary group buffer pool is not allocated, message DSNB799I is not included in the output.

```

DSNB750I - DISPLAY FOR GROUP BUFFER POOL GBP29 FOLLOWS
DSNB755I - DB2 GROUP BUFFER POOL STATUS
            CONNECTED = YES
            CURRENT DIRECTORY TO DATA RATIO = 5
            PENDING DIRECTORY TO DATA RATIO = 5
            CURRENT GBPCACHE ATTRIBUTE = YES
            PENDING GBPCACHE ATTRIBUTE = YES
DSNB756I - CLASS CASTOUT THRESHOLD = 10%
            GROUP BUFFER POOL CASTOUT THRESHOLD = 50%
            GROUP BUFFER POOL CHECKPOINT INTERVAL = 8 MINUTES
            RECOVERY STATUS = NORMAL
            AUTOMATIC RECOVERY = Y
DSNB757I - MVS CFRM POLICY STATUS FOR DSNCAT_GBP29 = NORMAL
            MAX SIZE INDICATED IN POLICY = 2048 KB
            DUPLEX INDICATOR IN POLICY = ENABLED
            CURRENT DUPLEXING MODE = DUPLEX
            ALLOCATED = YES
DSNB758I - ALLOCATED SIZE = 2048 KB
            VOLATILITY STATUS = VOLATILE
            REBUILD STATUS = DUPLEXED
            CFNAME = CACHE01
            CFLEVEL - OPERATIONAL = 5
            CFLEVEL - ACTUAL = 7
DSNB759I - NUMBER OF DIRECTORY ENTRIES = 1950
            NUMBER OF DATA PAGES = 389
            NUMBER OF CONNECTIONS = 2
DSNB798I - LAST GROUP BUFFER POOL CHECKPOINT
            17:08:41 OCT 16, 2011
            GBP CHECKPOINT RECOVERY LRSN = AF6BBAEF3307
            STRUCTURE OWNER = V61B
DSNB799I - SECONDARY GBP ATTRIBUTES
            ALLOCATED SIZE = 2048 KB
            VOLATILITY STATUS = VOLATILE
            CFNAME = LF01
            CFLEVEL - OPERATIONAL = 5
            CFLEVEL - ACTUAL = 7
            NUMBER OF DIRECTORY ENTRIES = 1950
            NUMBER OF DATA PAGES = 389
DSNB790I - DISPLAY FOR GROUP BUFFER POOL GBP29 IS COMPLETE
DSN9022I - DSNB1CMD '-DISPLAY GBPOOL' NORMAL COMPLETION

```

Example: Displaying all connections to a group buffer pool

The following command displays a summary report about group buffer pool 29 (GBP29), including all connections to that group buffer pool:

```
-DISPLAY GROUPBUFFERPOOL(GBP29) CONNLIST(YES)
```

The output is similar to this output:

```

DSNB750I - DISPLAY FOR GROUP BUFFER POOL GBP29 FOLLOWS
DSNB755I - DB2 GROUP BUFFER POOL STATUS
            CONNECTED = YES
            CURRENT DIRECTORY TO DATA RATIO = 5
            PENDING DIRECTORY TO DATA RATIO = 5
            CURRENT GBPCACHE ATTRIBUTE = YES
            PENDING GBPCACHE ATTRIBUTE = YES
DSNB756I - CLASS CASTOUT THRESHOLD = 10%
            GROUP BUFFER POOL CASTOUT THRESHOLD = 50%
            GROUP BUFFER POOL CHECKPOINT INTERVAL = 8 MINUTES
            RECOVERY STATUS = NORMAL
            AUTOMATIC RECOVERY = Y
DSNB757I - MVS CFRM POLICY STATUS FOR DSNCAT_GBP29 = NORMAL
            MAX SIZE INDICATED IN POLICY = 2048 KB
            DUPLEX INDICATOR IN POLICY = ENABLED
            CURRENT DUPLEXING MODE = SIMPLEX
            ALLOCATED = YES
DSNB758I - ALLOCATED SIZE = 2048 KB
            VOLATILITY STATUS = VOLATILE

```

```

REBUILD STATUS = DUPLEXED
CFNAME = CACHE01
CFLEVEL - OPERATIONAL = 5
CFLEVEL - ACTUAL = 7
DSNB759I - NUMBER OF DIRECTORY ENTRIES = 1950
NUMBER OF DATA PAGES = 389
NUMBER OF CONNECTIONS = 2
DSNB798I - LAST GROUP BUFFER POOL CHECKPOINT
17:08:41 OCT 16, 2011
GBP CHECKPOINT RECOVERY LRSN = AF6BBAEF3307
STRUCTURE OWNER = V61B
DSNB799I - SECONDARY GBP ATTRIBUTES
ALLOCATED SIZE = 2048 KB
VOLATILITY STATUS = VOLATILE
CFNAME = LF01
OPERATIONAL CFLEVEL = 5
ACTUAL CFLEVEL = 7
NUMBER OF DIRECTORY ENTRIES = 1950
NUMBER OF DATA PAGES = 389
DSNB766I - THE CONNLIST REPORT FOLLOWS
DSNB767I - CONNECTION NAME = DB2_V61B , CONNECTION STATUS = D
CONNECTOR'S RELEASE = 6100
DSNB767I - CONNECTION NAME = DB2_V61A , CONNECTION STATUS = D
CONNECTOR'S RELEASE = 6100
DSNB769I - THE CONNLIST REPORT IS COMPLETE
DSNB790I - DISPLAY FOR GROUP BUFFER POOL GBP29 IS COMPLETE
DSN9022I - DSNB1CMD '-DISPLAY GBPOOL' NORMAL COMPLETION

```

Example: Displaying a detailed report for a group buffer pool

Issue the following command to display detailed information about group buffer pool GBP29:

```
-DISPLAY GROUPBUFFERPOOL(GBP29) GDETAIL(*)
```

The output is similar to this output. Message DSNB762I is displayed in the output only if the secondary group buffer pool is allocated.

```

DSNB750I - DISPLAY FOR GROUP BUFFER POOL GBP29 FOLLOWS
DSNB755I - DB2 GROUP BUFFER POOL STATUS
CONNECTED = YES
CURRENT DIRECTORY TO DATA RATIO = 5
PENDING DIRECTORY TO DATA RATIO = 5
CURRENT GBPCACHE ATTRIBUTE = YES
PENDING GBPCACHE ATTRIBUTE = YES
DSNB756I - CLASS CASTOUT THRESHOLD = 10%
GROUP BUFFER POOL CASTOUT THRESHOLD = 50%
GROUP BUFFER POOL CHECKPOINT INTERVAL = 8 MINUTES
RECOVERY STATUS = NORMAL
AUTOMATIC RECOVERY = Y
DSNB757I - MVS CFRM POLICY STATUS FOR DSNCAT_GBP29 = NORMAL
MAX SIZE INDICATED IN POLICY = 2048 KB
DUPLEX INDICATOR IN POLICY = ENABLED
CURRENT DUPLEXING MODE = DUPLEX
ALLOCATED = YES
DSNB758I - ALLOCATED SIZE = 2048 KB
VOLATILITY STATUS = VOLATILE
REBUILD STATUS = DUPLEXED
CFNAME = CACHE01
CFLEVEL - OPERATIONAL = 5
CFLEVEL - ACTUAL = 7
DSNB759I - NUMBER OF DIRECTORY ENTRIES = 1950
NUMBER OF DATA PAGES = 389
NUMBER OF CONNECTIONS = 2
DSNB798I - LAST GROUP BUFFER POOL CHECKPOINT
17:08:41 OCT 16, 2011
GBP CHECKPOINT RECOVERY LRSN = AF6BBAEF3307
STRUCTURE OWNER = V61B
DSNB799I - SECONDARY GBP ATTRIBUTES

```

```

        ALLOCATED SIZE                = 2048 KB
        VOLATILITY STATUS              = VOLATILE
        CFNAME                        = LF01
        OPERATIONAL CFLEVEL           = 5
        ACTUAL CFLEVEL                = 7
        NUMBER OF DIRECTORY ENTRIES   = 1950
        NUMBER OF DATA PAGES         = 389
DSNB783I - CUMULATIVE GROUP DETAIL STATISTICS SINCE 17:08:35 OCT 16,
2011
DSNB784I - GROUP DETAIL STATISTICS
        READS
            DATA RETURNED            = 4
DSNB785I -     DATA NOT RETURNED
            DIRECTORY ENTRY EXISTED   = 0
            DIRECTORY ENTRY CREATED   = 45
            DIRECTORY ENTRY NOT CREATED = 0, 0
DSNB786I -     WRITES
            CHANGED PAGES             = 5
            CLEAN PAGES               = 0
            FAILED DUE TO LACK OF STORAGE = 0
            CHANGED PAGES SNAPSHOT VALUE = 5
DSNB787I -     RECLAIMS
            FOR DIRECTORY ENTRIES     = 0
            FOR DATA ENTRIES         = 0
            CASTOUTS                  = 0
DSNB788I -     CROSS INVALIDATIONS
            DUE TO DIRECTORY RECLAIMS = 0
            DUE TO WRITES             = 0
            EXPLICIT                  = 0
DSNB762I - DUPLEXING STATISTICS FOR GBP29-SEC
        WRITES
            CHANGED PAGES             = 5
            FAILED DUE TO LACK OF STORAGE = 0
            CHANGED PAGES SNAPSHOT VALUE = 5
DSNB790I - DISPLAY FOR GROUP BUFFER POOL GBP29 IS COMPLETE
DSN9022I - DSNB1CMD '-DISPLAY GBPOOL' NORMAL COMPLETION

```

Example: Displaying member detail information for a group buffer pool

The following command displays member detail information for group buffer pool GBP29:

```
-DISPLAY GROUPBUFFERPOOL(GBP29) MDETAIL(*)
```

The output is similar to this output. Messages DSNB764I and DSNB793I are displayed in the output only if the secondary group buffer pool is allocated.

```

DSNB750I - DISPLAY FOR GROUP BUFFER POOL GBP29 FOLLOWS
DSNB755I - DB2 GROUP BUFFER POOL STATUS
        CONNECTED                    = YES
        CURRENT DIRECTORY TO DATA RATIO = 5
        PENDING DIRECTORY TO DATA RATIO = 5
        CURRENT GBPCACHE ATTRIBUTE     = YES
        PENDING GBPCACHE ATTRIBUTE     = YES
DSNB756I - CLASS CASTOUT THRESHOLD   = 10%
        GROUP BUFFER POOL CASTOUT THRESHOLD = 50%
        GROUP BUFFER POOL CHECKPOINT INTERVAL = 8 MINUTES
        RECOVERY STATUS                 = NORMAL
        AUTOMATIC RECOVERY              = Y
DSNB757I - MVS CFRM POLICY STATUS FOR DSNCAT_GBP29 = NORMAL
        MAX SIZE INDICATED IN POLICY    = 2048 KB
        DUPLEX INDICATOR IN POLICY     = ENABLED
        CURRENT DUPLEXING MODE         = DUPLEX
        ALLOCATED                      = YES
DSNB758I -     ALLOCATED SIZE          = 2048 KB
            VOLATILITY STATUS          = VOLATILE
            REBUILD STATUS             = DUPLEXED

```

```

          CFNAME = CACHE01
          CFLEVEL - OPERATIONAL = 5
          CFLEVEL - ACTUAL = 7
DSNB759I - NUMBER OF DIRECTORY ENTRIES = 1950
          NUMBER OF DATA PAGES = 389
          NUMBER OF CONNECTIONS = 2
DSNB798I - LAST GROUP BUFFER POOL CHECKPOINT
          17:08:41 OCT 16, 2011
          GBP CHECKPOINT RECOVERY LRSN = AF6BBAEF3307
          STRUCTURE OWNER = V61B
DSNB799I - SECONDARY GBP ATTRIBUTES
          ALLOCATED SIZE = 2048 KB
          VOLATILITY STATUS = VOLATILE
          CFNAME = LF01
          OPERATIONAL CFLEVEL = 5
          ACTUAL CFLEVEL = 7
          NUMBER OF DIRECTORY ENTRIES = 1950
          NUMBER OF DATA PAGES = 389
DSNB772I - CUMULATIVE MEMBER DETAIL STATISTICS SINCE 17:08:41 OCT 16,
2011
DSNB773I - MEMBER DETAIL STATISTICS
          SYNCHRONOUS READS
            DUE TO BUFFER INVALIDATION
              DATA RETURNED = 0
              DATA NOT RETURNED = 0
DSNB774I - DUE TO DATA PAGE NOT IN BUFFER POOL
              DATA RETURNED = 0
              DATA NOT RETURNED = 0
DSNB775I - PREFETCH READS
              DATA NOT RETURNED = 0
DSNB789I - REGISTER PAGE LIST = 0
          PAGES RETRIEVED = 0
DSNB776I - SYNCHRONOUS WRITES
          CHANGED PAGES = 5
          CLEAN PAGES = 0
DSNB777I - ASYNCHRONOUS WRITES
          CHANGED PAGES = 0
          CLEAN PAGES = 0
          FAILED WRITES DUE TO LACK OF STORAGE = 0
DSNB778I - CASTOUT THRESHOLDS DETECTED
          FOR CLASSES = 0
          FOR GROUP BUFFER POOL = 0
          GBP CHECKPOINTS TRIGGERED = 0
          PARTICIPATION IN REBUILD = 1
DSNB796I - CASTOUTS
          PAGES CASTOUT = 0
          UNLOCK CASTOUT = 0
          READ CASTOUT CLASS = 0
          READ CASTOUT STATISTICS = 0
          READ DIRECTORY INFO = 0
DSNB797I - OTHER INTERACTIONS
          REGISTER PAGE = 0
          UNREGISTER PAGE = 0
          DELETE NAME = 0
          READ STORAGE STATISTICS = 0
          EXPLICIT CROSS INVALIDATIONS = 0
          ASYNCHRONOUS GBP REQUESTS = 0
DSNB764I - DUPLEXING STATISTICS FOR GBP29-SEC
          WRITES
            FAILED DUE TO LACK OF STORAGE = 0
            ASYNC COMPLETION CHECKS = 0
DSNB793I - DELETE NAME LIST = 0
          READ CASTOUT STATISTICS = 0
          DELETE NAME = 0

```

```
OTHER ASYNCHRONOUS GBP REQUESTS          = 0
DSNB790I - DISPLAY FOR GROUP BUFFER POOL GBP29 IS COMPLETE
DSN9022I - DSNB1CMD '-DISPLAY GBPOOL' NORMAL COMPLETION
```

Chapter 32. -DISPLAY LOCATION (DB2)

The DB2 command DISPLAY LOCATION displays various information about the specified remote locations.

When you specify the DETAIL keyword, information about the numbers of connections with partner locations that have particular attributes are shown, and detailed information about connections owned by DB2 system threads that are communicating with the location might also be shown for each partner location.

The information returned by the DISPLAY LOCATION command reflects a dynamic status. By the time the information is displayed, it is possible that the status has changed.

Abbreviation: -DIS LOC

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

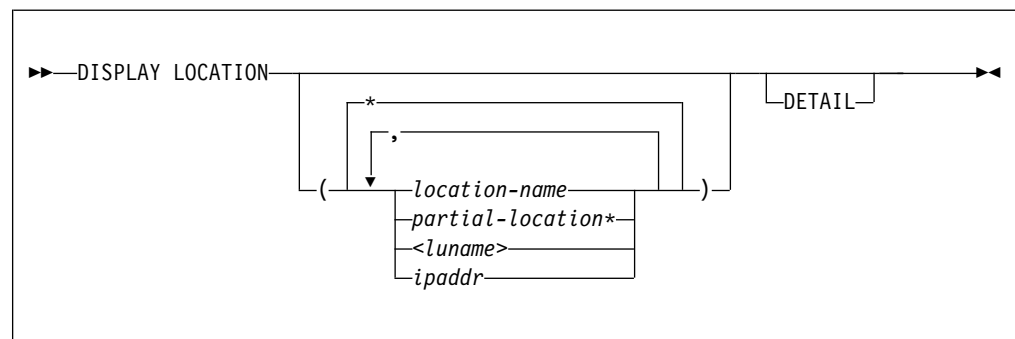
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- DISPLAY system privilege
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

(*)

Displays information for all remote locations.

(location-name)

Lists one or more location names, separated by commas.

Because DB2 does not receive a location name from requesters that are not DB2 for z/OS subsystems, you can enter the LUNAME or IP address of such a requester. Refer to the option descriptions for the < *luname* > and (*ipaddr*) options for more information about using the LUNAME or IP address to specify a requester that is not a DB2 for z/OS subsystem.

(partial-location *)

Selects all location names that begin with the string *partial-location* and can end with any string, including the empty string. For example, LOCATION(ABC*) selects all location names that begin with the string 'ABC'.

< luname >

Requests information about the remote clients that are connected to DDF through the remote SNA LU that is specified. Enclose the LU name in the less-than (<) and greater-than (>) symbols. For example, DISPLAY LOCATION(<LULA>) displays information about a remote location (that is not DB2 for z/OS) with the LU name of LULA.

You can use an asterisk (*) when specifying an LU name in the same manner as previously described for specifying a partial-location name. For example, DISPLAY LOCATION(<LULA*) selects all remote locations (that are not DB2 for z/OS) with an LU name that begins with the string 'LULA'.

(ipaddr)

Requests information about the clients that are connected to DDF through the remote TCP/IP host. Enter the IP address. For example, DISPLAY LOCATION(::FFFF:124.63.51.17) displays information about clients at the remote TCP/IP host whose dotted decimal IP address is 124.63.51.17.

DETAIL

Displays additional information about conversation activity for DB2 system threads.

Output

Message DSNL200I (DB2 Messages) indicates the beginning of the output of the command.

Examples

Example: Displaying information about threads and conversations with specific remote locations

The following command displays information about threads and conversations with remote locations LUND1, LUND2, and LUND3.

```
DISPLAY LOCATION(LUND1,LUND2,LUND3)
```

The output looks similar to this output:

```
| DSNL200I @ DISPLAY LOCATION REPORT FOLLOWS-  
| LOCATION PRDID T ATT CONNS  
| LUND1 DSN10015 R 1  
| LUND2 DSN10015 R 1
```



```

LUND2                DSN10015 S      1
LUND3                DSN10015 R      1
LUND3                DSN10015 S      3
DISPLAY LOCATION REPORT COMPLETE

```

Example: Displaying information about conversations and threads with all remote locations

The following command displays detailed information about conversations with all remote locations, and detail conversation information about DB2 system threads that communicate with other locations.

```
-DISPLAY LOCATION DETAIL
```

The output looks similar to this output:

```

DSNL200I @ DISPLAY LOCATION REPORT FOLLOWS-
DSNL200I @ DISPLAY LOCATION REPORT FOLLOWS-
LOCATION                                PRDID    T ATT CONNS
LUND1                                DSN10015 R      1
LUND2                                DSN10015 R      1
LUND2                                DSN10015 S      1
L203-SYSTASK  SESSID          A ST TIME
L204-RESYNC   E15FDE02D310FB84 N R  0820914411184
LUND3                                DSN10015 R      1
LUND3                                DSN10015 S      3
L203-SYSTASK  SESSID          A ST TIME
L204-RESYNC   E15FDE02D310FB87 W R  0820914411187
L204-RESYNC   E15FDE02D310FB88 W R  0820914411188
L204-RESYNC   E15FDE02D310FB89 W R  0820914411189

DISPLAY LOCATION REPORT COMPLETE
DISPLAY LOCATION REPORT COMPLETE

```

Example: Displaying information about a connection with DRDA partner sites:

Suppose that a DB2 system is connected with the following DRDA partners:

- A non-z/OS server via TCP/IP IPv4.
- Several TCP/IP clients from the same TCP/IP host as the server.
- A DB2 for z/OS server via SNA.
- A DB2 for z/OS client via TCP/IP IPv6.

The output from the following command shows information about those DRDA partners:

```
-DISPLAY LOCATION(*)
```

The output is similar to the following output:

```

DSNL200I @ DISPLAY LOCATION REPORT FOLLOWS-
LOCATION                                PRDID    T ATT CONNS
::FFFF:124.63.51.17..50000           SQL09073 R      3
::FFFF:124.63.51.17                 SQL09073 S     15
LULA                                  DSN10015 S      1
2002:91E:610:1::5                   DSN10015 S      1

```

Example: Display information about a trusted connection:

Suppose that there is trusted connection to location ::FFFF:9.30.115.135. You issue the following command:

```
-DIS LOC DET
```

A value of TRS in the ATT field indicates that a connection is trusted, as shown in the following example:

Chapter 33. -DISPLAY LOG (DB2)

The DB2 command DISPLAY LOG displays log information about, and the status of, the offload task.

Abbreviation: DIS LOG

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- DISPLAY system privilege
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



```
▶▶—DISPLAY LOG————▶▶
```

Usage notes

Information provided by the DISPLAY LOG command : You can use the DISPLAY LOG command to view the current checkpoint scheduling parameters and information about the current active log data sets and status of the offload task. You can obtain additional information about log data sets and checkpoint information by using the Print Log Map utility (DSNJU004).

Output

Message DSNJ370I (DB2 Messages) indicates the beginning of the output of the command.

Examples

Example: Displaying log information

The following command displays information about the log and the status of any offload tasks.

```
-DISPLAY LOG
```

The output is similar to the following output:

```
DSNJ370I - DSNJC00A LOG DISPLAY
CURRENT COPY1 LOG = DSN100.LOGCOPY1.DS03 IS 22% FULL
CURRENT COPY2 LOG = DSN100.LOGCOPY2.DS03 IS 22% FULL
          H/W RBA = 0000039A9F24,
          LOGLOAD = 150000
          FULL LOGS TO OFFLOAD = 2 OF 6,
          OFFLOAD TASK IS (BUSY,ALLC)
          SOFTWARE ACCELERATION IS ENABLED
DSNJ371I - DB2 RESTARTED 14:06:23 MAY 22, 2011
RESTART RBA 0000039A8000
DSN9022I - DSNJC001 'DIS LOG' NORMAL COMPLETION
```

The output provides the following information:

- The active log data sets are 22% full. If you are running dual logs and the percentages are different, the log data sets are of different sizes. DB2 switches both active logs when one reaches the end of the file. This can result in unused active log space if one log data set is larger than the other.
- The current LOGLOAD setting is 150000 log records between system checkpoints. You can modify this value using the SET LOG command.
- Two of the six active log data sets require archiving. The status of the offload task includes the indicator that it is busy, allocating an archive log data set. This might be an indication of an outstanding tape mount on the system console. If the status remains busy and no longer seems to be functioning, you can terminate the task and then restart it using the ARCHIVE LOG CANCEL OFFLOAD command.
- DB2 was started at 14:06:23 on MAY 22, 2011, and began logging at RBA 0000039A8000.
- Software acceleration is enabled for peer-to-peer remote copy.

Chapter 34. -DISPLAY PROCEDURE (DB2)

The DB2 command DISPLAY PROCEDURE displays statistics about stored procedures that are accessed by DB2 applications.

This command displays one line of output for each stored procedure that a DB2 application has accessed. You can qualify stored procedure names with a schema name.

The information returned by the DISPLAY PROCEDURE command reflects a dynamic status. By the time the information is displayed, it is possible that the status could have changed.

Abbreviation: -DIS PROC

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or a CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the value of the SCOPE option.

Authorization

To run this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- DISPLAY system privilege
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

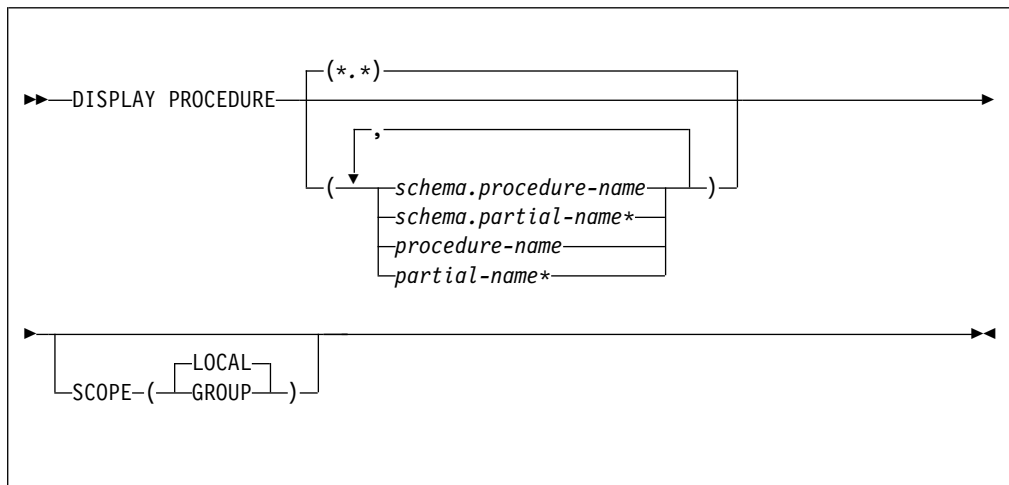
If you specify DISPLAY PROCEDURE *.* or *schema.partial-name **, the privilege set of the process must include one of the following authorities:

- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

If you are using an external security product to authorize usage of DISPLAY PROCEDURE, define SYSOPR as a user to the external security product for those cases in which DISPLAY PROCEDURE SCOPE(GROUP) runs on a remote system and SYSOPR is used as the authorization ID.

Syntax



Option descriptions

(*,*)

Displays information for all stored procedures in all schemas that DB2 applications have accessed since DB2 was started.

(schema.procedure-name)

Displays the specified stored procedure in the specified schema.

(schema.partial-name *)

Displays a set of stored procedures in the specified schema that DB2 applications have accessed since DB2 was started. The names of all procedures in the set begin with *partial-name* and can end with any string, including the empty string. For example, PAYROLL.ABC* displays information for all stored procedure names beginning with ABC in the PAYROLL schema.

(procedure-name)

Displays one or more specific stored procedure names in the SYSPROC schema. If no procedures are named, DB2 displays information for all stored procedures that have been accessed by DB2 applications.

(partial-name *)

Displays information for a set of stored procedures in the SYSPROC schema that DB2 applications have accessed since DB2 was started. The names of all procedures in the set begin with *partial-name* and can end with any string, including the empty string. For example, ABC* displays information for all stored procedures in the SYSPROC schema with names that begin with ABC.

SCOPE

Specifies the scope of the command.

(LOCAL)

Specify to display information about procedures on the local member only.

(GROUP)

Specify to display information about procedures on all members of the data sharing group.

Output

Message DSNX940I (DB2 Messages) indicates the beginning of the output of the command.

Displaying native SQL procedures: Native SQL procedures are not displayed in the DISPLAY PROCEDURE output unless you run the procedures in DEBUG mode. If you run a procedure in DEBUG mode, the WLM environment column in the output contains the WLM ENVIRONMENT FOR DEBUG that you specified when you created the native SQL procedure.

The DISPLAY PROCEDURE output shows the statistics of a native SQL procedure as 0 if the STOP PROCEDURE command is processing the native SQL procedure.

Examples

Example: Displaying information about all stored procedures that have been accessed

The following command displays information about all stored procedures that applications have accessed in a DB2 subsystem.

```
-DISPLAY PROCEDURE
```

The output is similar to the following output:

```
DSNX940I = DSNX9DIS DISPLAY PROCEDURE REPORT FOLLOWS-
```

```
----- SCHEMA=SYSPROC
PROCEDURE      STATUS    ACTIVE    QUED    MAXQ    TIMEOUT  FAIL    WLM_ENV
USERPRC1       STARTED   0         0       1       0        0      SANDBOX
USERPRC2       STOPQUE   0         5       5       3        0      SANDBOX
USERPRC3       STARTED   2         0       6       0        0      SANDBOX
USERPRC4       STOPREJ   0         0       1       0        0      SANDBOX
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I = DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

Example: Displaying information about a specific stored procedure

The following command displays information about stored procedures USERPRC2 and USERPRC4 in the SYSPROC schema. The SYSPROC schema is the default schema if the schema name is not explicitly specified.

```
-DISPLAY PROCEDURE(USERPRC2,USERPRC4)
```

The output is similar to the following output:

```
DSNX940I = DSNX9DIS DISPLAY PROCEDURE REPORT FOLLOWS-
```

```
----- SCHEMA=SYSPROC
PROCEDURE      STATUS    ACTIVE    QUED    MAXQ    TIMEOUT  FAIL    WLM_ENV
USERPRC2       STOPQUE   0         5       5       3        0      SANDBOX
USERPRC4       STOPREJ   0         0       1       0        0      SANDBOX
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I = DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

Example: Displaying information about all stored procedures in specified schemas

The following command displays information about all stored procedures in the PAYROLL schema and in the HRPROD schema:

```
-DISPLAY PROCEDURE(PAYROLL.*,HRPROD.*)
```

The output is similar to the following output.

DSNX940I = DSNX9DIS DISPLAY PROCEDURE REPORT FOLLOWS-

```
----- SCHEMA=PAYROLL
PROCEDURE      STATUS    ACTIVE    QUED    MAXQ    TIMEOUT  FAIL    WLM_ENV
PAYPRC1        STARTED    0         0       1       0        0      PAYROLL
PAYPRC2        STOPQUE    0         5       5       3        0      PAYROLL
PAYPRC3        STARTED    2         0       6       0        0      PAYROLL
USERPRC4       STOPREJ    0         0       1       0        0      SANDBOX
```

```
----- SCHEMA=HRPROD
PROCEDURE      STATUS    ACTIVE    QUED    MAXQ    TIMEOUT  FAIL    WLM_ENV
HRPRC1         STARTED    0         0       1       0        0      HRPROCS
HRPRC2         STOPREJ    0         0       1       0        0      HRPROCS
```

DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I = DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION

Example: Displaying information about stopped procedures

Suppose that all stored procedures in schema SYSADM that begin with the characters "SP" have been stopped. The following command displays information about those stored procedures.

```
-DISPLAY PROCEDURE(SYSADM.SP*)
```

The output is similar to the following output.

DSNX940I = DSNX9DIS DISPLAY PROCEDURE REPORT FOLLOWS-

```
----- SCHEMA=SYSADM
PROCEDURE      STATUS    ACTIVE    QUED    MAXQ    TIMEOUT  FAIL    WLM_ENV
SPC1           STOPQUE    0         0       0       0        0      WLMENV1
SPC2           STOPQUE    0         0       0       0        0      WLMENV3
```

DSNX9DIS PROCEDURES SP - SP* STOP QUEUE
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I = DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION

Chapter 35. -DISPLAY PROFILE (DB2)

The DISPLAY PROFILE command allows you to determine if profiling is active or inactive

Abbreviation: -DIS PROFILE

Environment

This command can be issued from the z/OS console, through a batch job or the instrumentation facility interface (IFI). However, in general, this command is intended for use by tools.

Data sharing scope: Member

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SQLADM authority
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

Syntax

```
▶▶—DISPLAY PROFILE—◀◀
```

Examples

The following command displays the status of active and inactive profiling activities.

```
-DISPLAY PROFILE
```

Output

Message DSNT753I (DB2 Messages) indicates the beginning of the output of the command.

Chapter 36. -DISPLAY RLIMIT (DB2)

The DB2 command DISPLAY RLIMIT displays the current status of the resource limit facility (governor).

If the facility has already been started, DISPLAY RLIMIT also displays the ID of the resource limit specification table or the resource limit middleware table that is being used.

Abbreviation: -DIS RLIM

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Output

Message DSNT700I (DB2 Messages) indicates the beginning of the output of the command.

Syntax

```
▶▶—DISPLAY RLIMIT—▶▶
```

Examples

Example: Displaying the current status of the resource limit facility

The following command displays the current status of the resource limit facility.

```
-DISPLAY RLIMIT
```



When the resource limit facility is inactive, the output is similar to the following output.

```
DSNT701I - RESOURCE LIMIT FACILITY IS INACTIVE  
DSN9022I - DSNTCDIS 'DISPLAY RLIMIT' NORMAL COMPLETION
```


When the resource limit facility is active, the output is similar to the following output.

```
DSNT700I = SYSADM.DSNRLST03 IS THE ACTIVE RESOURCE LIMIT  
SPECIFICATION TABLE  
DSN9022I = DSNTCDIS 'DISPLAY RLIMIT' NORMAL COMPLETION
```

Related concepts:

-  [The resource limit facility \(Introduction to DB2 for z/OS\)](#)
-  [Resource limit facility controls \(DB2 Performance\)](#)


Related tasks:

-  [Setting limits for system resource usage by using the resource limit facility \(DB2 Performance\)](#)

Related reference:

- START RLIMIT (DB2)
- STOP RLIMIT (DB2)

Related information:

-  [DSNT700I \(DB2 Messages\)](#)

Chapter 37. -DISPLAY THREAD (DB2)

The DB2 command DISPLAY THREAD displays current status information about DB2 threads.

A DB2 thread can be an allied thread, a database access thread, or a parallel task thread. Threads can be active, inactive, indoubt, or postponed.

Distributed threads are those threads that have a connection with a remote location (active or inactive) or that had a connection with a remote location (indoubt). An allied thread and a parallel task thread can be distributed or non-distributed; a database access thread is always distributed.

The DISPLAY THREAD command allows you to select the type of information you want to display by using one or more of the following criteria:

- Active threads, inactive threads, indoubt threads, postponed threads, procedure threads, system threads, or the set of active, indoubt, postponed, and system threads (see the descriptions under the TYPE option for more information)
- Allied threads, including those threads that are associated with the address spaces whose connection names are specified
- Distributed threads, including those threads that are associated with a specific remote location
- Detailed information about connections with remote locations
- A specific logical unit of work ID (LUWID)

The information that is returned by the DISPLAY THREAD command reflects a dynamic status. When the information is displayed, it is possible that the status has changed. Moreover, the information is consistent only within one address space and is *not necessarily* consistent across all address spaces displayed.

Abbreviation: -DIS THD

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the SCOPE option.

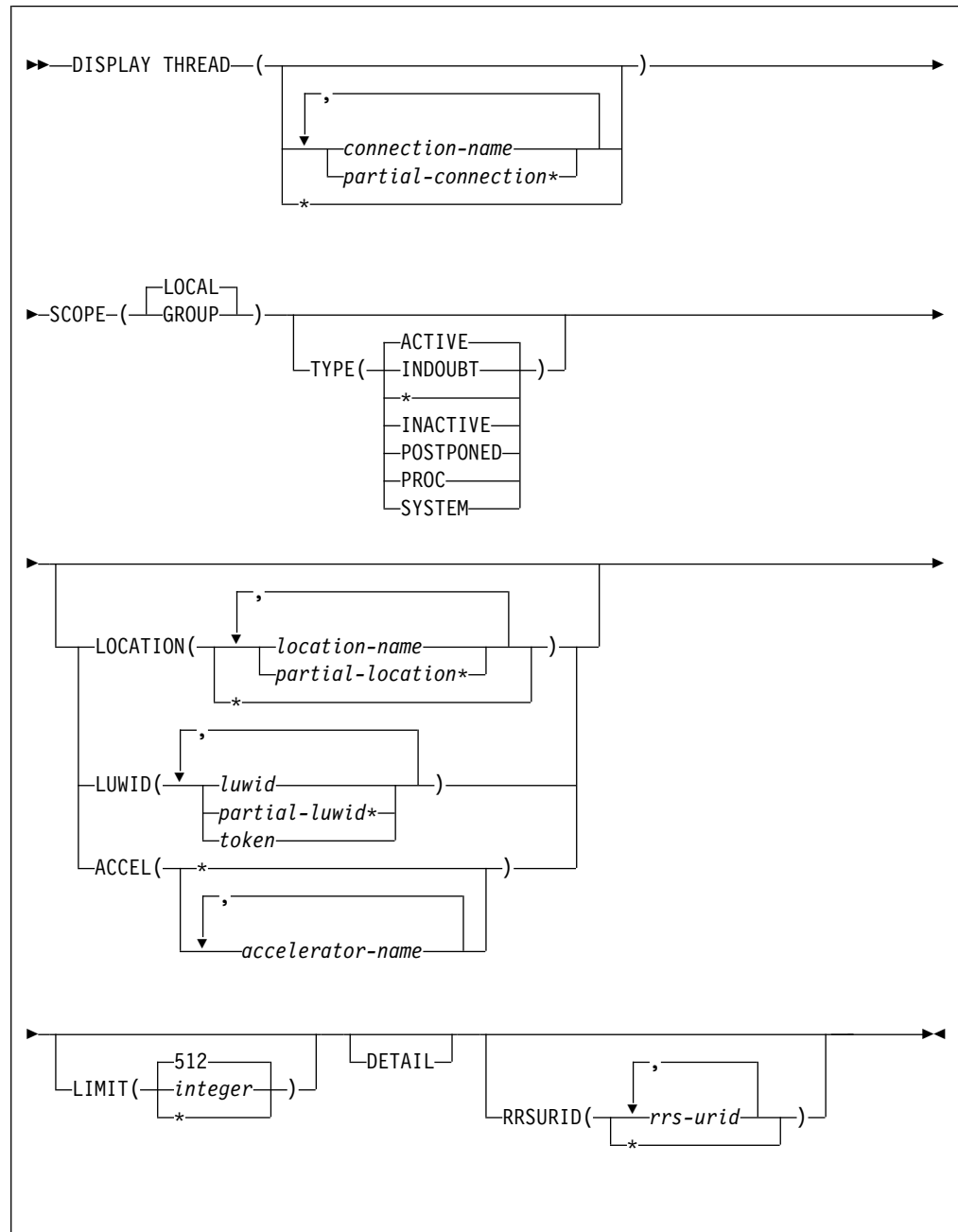
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- DISPLAY system privilege
- System DBADM authority
- SYSOPR authority
- SYCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

Only under certain conditions, as described in the following lists, are any of the following options required.

If you do not specify either (*connection-name*) or (*), the following rules apply:

- If the command is issued from a DSN session under TSO, a DB2I panel (DB2 COMMANDS), or an IMS or CICS terminal, the connection name is inherited from the associated address space.
- If the command is not issued from one of those environments, the following rules apply:
 - If you do not specify either LOCATION or LUWID, processing terminates with a DSNV413I message.
 - If you do specify LOCATION or LUWID, only distributed threads of the type selected by the TYPE option are displayed.
 - When you explicitly specify *location-name*, only distributed threads of the type selected by the TYPE option that either have (active or inactive threads) or had (indoubt threads) a connection with the specified location are displayed.

(*connection-name* , ...)

Lists one or more connection names (of 1 to 8 characters each). Allied threads are selected only from the address spaces associated with those connection names. The LOCATION option can restrict what is displayed:

- If you specify LOCATION(*), only distributed threads of the type specified in the TYPE option are displayed.
- When you explicitly specify *location-name*, only distributed threads of the specified type that either have or had a connection with the specified location are displayed.

(*partial-connection , ...)**

Selects the connections that begin with the string *partial-connection* and can end with any string, including the empty string. For example, DISPLAY THREAD(CICS*,IMS*) selects all connection names that begin with the string 'CICS' or 'IMS'. The LOCATION option can restrict the display exactly the same way as previously described for *location-name*.

(*)

Displays all threads in all address spaces attached to DB2 and all database access threads of the types specified in the TYPE option. The LOCATION option can restrict what is displayed:

- If you specify LOCATION(*), only distributed threads are displayed.
- When you explicitly specify *location-name*, only distributed threads that either have (active or inactive threads) or had (indoubt threads) a connection with the specified location are displayed.

The **default** is to display only the connections that are associated with the transaction manager from which the command was entered.

SCOPE

Specifies the scope of the command.

(LOCAL)

Displays threads on only the current member.

(GROUP)

Displays all threads on the data sharing group.

TYPE

Tells the type of thread to display.

Abbreviation: T

(ACTIVE)

Displays only active threads. An active allied thread is connected to DB2

via TSO, BATCH, IMS, CICS or CAF. An active database access thread is connected by a network connection to another system and is performing work on behalf of that system. If, during command processing, an active thread becomes indoubt, it can appear twice—once as active and once as indoubt.

Abbreviation: A

The information that is produced by ACTIVE can be useful for debugging purposes, especially message DSNV402I.

(INDOUBT)

Displays only indoubt threads.

An indoubt thread is a participant in a two-phase commit protocol that has completed the first phase of commit, and has then lost communication with the commit coordinator and does not know whether to commit or roll back the updates that have been made.

The indoubt thread information that is displayed includes threads for which DB2 has a coordinator role, a participant role, or both coordinator and participant roles.

The commit coordinator for an allied thread is either a transaction manager (for example, IMS or CICS) or z/OS RRS for threads that use RRSAF. The commit coordinator for a database access thread is a requester at a remote system.

Indoubt threads hold locks on all resources that were updated.

Abbreviation: I

(*)

Displays active, indoubt, postponed, and system threads.

(INACTIVE)

Displays only inactive database access threads and connections that are connected by a network connection to another system. An inactive thread is idle and waits for a new unit of work to begin from that system.

Abbreviation: INA

Use qualifiers such as complete location names or LUWIDs with this option. When there are large numbers of inactive database access threads, unqualified display requests can temporarily change the DB2 working set, which can temporarily affect the performance of active threads.

(POSTPONED)

Displays information about units of work whose back-out processing has been postponed.

Abbreviation: P

After you have identified postponed threads, use the RECOVER POSTPONED command, described, to complete backout processing for the postponed units of work.

(PROC)

Displays information about threads that are executing stored procedures and user-defined functions. All -DISPLAY THREAD keywords are valid when TYPE(PROC) is specified.

(SYSTEM)

Displays a subset of system agent threads with message DSNV402I. The

system agent threads displayed are deemed useful for serviceability purposes. Not all system threads are displayed.

Abbreviation SYS

If the command or system agent can be canceled, then the associated token is displayed in the output. You can specify this option from early in the restart to late in the shutdown process. If you issue this command before or after release work, then the restart or shutdown status is displayed in the output.

When the token is identified for the database command or systems agent, you can use the CANCEL THREAD(token) command to cancel it.

LOCATION(*location-name* , ...)

Limits the display to distributed threads as described.

Abbreviation: LOC

location-name

Displays only distributed threads of the specified type that either have (active or inactive threads) or had (indoubt threads) a remote connection with the specified *location-name* .

DB2 does not receive a location name from requesters that are not DB2 for z/OS subsystems. To display information about a requester that is not a DB2 for z/OS subsystem, enter its LU name or IP address. Enclose the LU name in the less-than (<) and greater-than (>) symbols. For example, the following command displays information about a remote location (that is not DB2 for z/OS) with the LU name of LULA:

```
-DISPLAY THREAD (*) LOCATION (<LULA>)
```

The following command displays information about a remote location (that is not DB2 for z/OS) with an IP address of ::FFFF:123.34.101.98:

```
-DISPLAY THREAD (*) LOCATION (::FFFF:123.34.101.98)
```

DB2 uses the <LU name> notation or IP address in messages displaying information about requesters other than DB2.

*partial-location **

Selects all location names that begin with the string *partial-location* and can end with any string, including the empty string. For example, LOCATION(SAN*) selects all location names that begin with the string 'SAN'.

You can use an asterisk (*) when specifying an LU name in the same manner as previously described for other location names that are not DB2 for z/OS subsystems. For example, LOCATION(<LULA*) selects all remote locations (that are not DB2 for z/OS) with an LU name that begins with the string 'LULA'.

You cannot use an asterisk when you specify an IP address.

(*)

Display all distributed threads of the specified type.

LUID(*luwid* , ...)

Displays information about the distributed threads that have the specified LUWID. It is possible for more than one thread to have the same LUWID.

luwid

Consists of a fully qualified LU network name followed by a period and an LUW instance number.

The LU network name consists of a one- to eight-character network ID, a period, and a one- to eight-character network LU name. The LUW instance number consists of 12 hexadecimal characters that uniquely identify the unit of work.

*partial-luwid **

Selects all LUWIDs that begin with the string *partial-luwid* and can end with any string, including the empty string. For example, LUWID(NET1.*) selects all LUWIDs with a network name of 'NET1'.

token

Identifies a specific thread in an alternate way. DB2 assigns a token to each distributed thread it creates. A token is a one- to six-digit decimal number that appears after the equal sign in all DB2 messages that display a LUWID.

If you do not include any periods nor a '*' in the LUWID specification, DB2 assumes that you are supplying a token. The token that DB2 assigns to a specific LUWID is unique for that DB2 subsystem, but not necessarily unique across subsystems.

ACCEL(*accelerator-name* , ...)

Limits the list to threads with active accelerator processes executing within the specified accelerator server.

accelerator-name

The specific accelerator server name. If ACCEL(*accelerator-name*) is specified, only threads active in that specific ACCEL will be displayed.

Supplying an asterisk (*) as the *accelerator-name* indicates that the display must include all threads with any accelerator server. If ACCEL(*) is specified, only threads currently active in an accelerator will be displayed.

DETAIL

Displays additional information about active, inactive, and indoubt threads.

LIMIT

Accepts numeric input that specifies the number of lines of output that you want. The default limit is 512, so you must use the limit keyword if you want to set a different limit.

If you select LIMIT(*), all output that can be properly formatted within internal storage constraints will be displayed.

When SCOPE(GROUP) is specified, the line limit is enforced per member displayed.

RRSURID(*rrs-urid*)

Specifies that only threads that match the specified RRSURID selection criteria are to be displayed.

- If RRSURID(*rrs-urid*) is specified, any thread involved in the RRSURID that has the value *rrs-urid*, and that meets any other specified selection criteria, will be displayed.
- If RRSURID(*) is specified, any thread involved in any RRSURID, and that meets any other specified selection criteria, will be displayed.

Usage notes

Formatted report for distributed threads

The series of messages, DSNV444I through DSNV446I, augment the formatted report for DISPLAY THREAD TYPE (ACTIVE or INACTIVE) for distributed threads.

Threads for connections to remote servers

A database access thread that is connected to a requester can also be connected to another database server location using DRDA access. In that case, DB2 issues message DSNV445I for the requester, and message DSNV444I and zero or more DSNV446I messages for the remote server connections. In addition, the database access thread acts as an intermediate database server.

Participant threads waiting for the commit or abort decision

A DSNV465I message is issued for an active participant thread that has completed phase 1 of commit processing and has been waiting for the commit or abort decision from the coordinator for more than 60 seconds.

DISPLAY THREAD output limit

If a DISPLAY THREAD command is issued from the z/OS console, the maximum number of lines of output for a single invocation of the command is 512 lines (at which time a DSNV513I, DSNV514I, or DSNV515I message is printed). If you do not receive the required information in the first 255 lines of output, issue the command again, specifying the TYPE option and a specific connection name, location, LUWID, or a combination of these, as appropriate, to reduce the output.

Showing parallel tasks

The DISPLAY THREAD command shows parallel tasks by using a status type of PT. The parallel tasks are displayed immediately after the originating task. If the thread has a status of PT, the connection name contains blanks if the thread of the originating task is running on the same DB2 subsystem. This shows that these parallel tasks are related to the originating task. If the parallel task is running on a DB2 subsystem that is different from the subsystem that runs the originating task, the connection name is shown and the entry is followed by message DSNV443I.

Displaying the XID

If the DISPLAY THREAD command is issued with the TYPE ACTIVE and DETAIL options, or with the TYPE INDOUBT option, message DSNV440I displays the contents of the XID. The contents of the XID are displayed as a hexadecimal value.

The XID is displayed in the DISPLAY THREAD TYPE INDOUBT report if the indoubt transaction is XID related.

Threads associated with trusted context

If a thread is associated with a trusted context, message DSNV485I displays the trusted context name, system authorization ID, and role.

Displaying threads while DB2 is active

All -DISPLAY and -CANCEL commands are allowed to run concurrent to other command issued through the console. This ensures thread tokens can be identified and used for soft thread cancels, even while long running console commands are in progress. The majority of system threads displayed cannot be canceled. For example, a 0 thread token is provided for DSNV402I messages.

The command preprocessor (GCPC) console commands tasks can be canceled, and a token will be provided for these threads in a DSNV402I message in the -DISPLAY THREAD(*) TYPE(SYSTEM) report.

Displaying threads during shutdown

All commands will be quiesced at the beginning of shutdown. Only -DISPLAY THREAD(*) commands will be accepted after this point, and these will be valid until the last stages of shutdown. Much like -DISPLAY THD(*) during restart, only -DISPLAY THREAD(*) TYPE(SYSTEM) will produce meaningful thread information. All other variations of -DISPLAY THREAD(*) will terminate without thread data. If SCOPE(GROUP) is specified during shutdown, then a DSNV495I message will be displayed and the command will only execute locally.

Output

Message DSNV401I (DB2 Messages) indicates the beginning of the output of the command.

Examples

Example: Displaying information about a local thread

The output of the command DISPLAY THREAD shows a token for every thread, distributed or not. This example shows the token for an allied thread that is not distributed. The token is 123. You can use the thread's token as the parameter in the CANCEL THREAD command.

```
-DISPLAY THREAD(*)
```

This command produces output similar to the following output:

```
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV402I - ACTIVE THREADS -
NAME      ST A   REQ ID           AUTHID  PLAN      ASID  TOKEN
BATCH    T  *     5 BKH2C           SYSADM  BKH2     000D  123
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

Example: Displaying information about threads at all locations

This example shows information about conversation activity when distribution information is displayed for active threads. DB2 returns the following output, indicating that the local site application is waiting for a conversation to be allocated in DB2, and that a DB2 server is accessed by a DRDA client using TCP/IP.

```
-DISPLAY THREAD(*) LOCATION(*) DETAIL
```

This command produces output similar to the following output:

```
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV402I - ACTIVE THREADS -
NAME      ST A   REQ ID           AUTHID  PLAN      ASID  TOKEN
TSO       TR *     9 SYSADM           SYSADM  DSNESPRR 0029   30
V441-ACCOUNTING=D1001
V436-PGM=DSNESPRR.DSNESM68, SEC=1, STMT=137, THREAD-INFO=SYSADM:***:
***:***:***
V444-USIBMSY.SYEC717A.BD6E55DCF589=30 ACCESSING DATA AT
( 1)STL716A-SYEC716A
V447--INDEX  SESSID           A ST   TIME
V448-- ( 1)  0000000000000000 N A4   052200855443
TSO       RA *     2 SYSADM           SYSADM  DSNESPRR 0033   28
V441-ACCOUNTING=D1001
V445-USIBMSY.SYEC716A.BD6E543A7BDF=28 ACCESSING DATA FOR
( 1)::FFFF:9.30.115.136
```

```

V447--INDEX    SESSID          A ST    TIME
V448--( 1) 446:1027          W R2    0522008483849
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION

```

Example: Displaying information about indoubt threads

In this example, a system at Site 1 has a TSO application and an IMS application. The system at Site 1 fails after DB2 commits the TSO application, but before the commit decision has been communicated to the participant subsystems at Site 2. The failure occurs before IMS has communicated the commit or rollback decision to the Site 1 DB2 subsystem. The DISPLAY THREAD commands that are issued at Site 1 and 2 show report output similar to the following:

The following DISPLAY THREAD command is issued at Site 1:

```
-DISPLAY THREAD(*) TYPE(INDOUBT)
```

This command produces output similar to the following output:

```

DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV406I - INDOUBT THREADS -
COORDINATOR          STATUS      RESET URID          AUTHID
TEST0001             INDOUBT          00000EA8BD60 SYSADM
V449-HAS NID= DONSQLXX.F100000000 AND ID= CTHDCORID001
V467-HAS LUWID USIBMSY.SYEC717A.BD6E5A52FFd1.0001=49
V466-THREAD HAS BEEN INDOUBT FOR    00:06:03
V450-HAS PARTICIPANT INDOUBT AT
STL717B-SYEC717B
STL717A             COMMITTED          00000EA8C6CE SYSADM
V467-HAS LUWID USIBMSY.SYEC717A.BD6E5B0A1F94.0001=52
V450-HAS PARTICIPANT INDOUBT AT
STL717B-SYEC717B
STL716A-SYEC716A
DISPLAY INDOUBT REPORT COMPLETE
DSN9022I - DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION

```

The following DISPLAY THREAD command is issued at Site 2:

```
-DISPLAY THREAD(*) TYPE(INDOUBT)
```

This command produces output similar to the following output:

```

DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV406I - INDOUBT THREADS -
COORDINATOR          STATUS      RESET URID          AUTHID
STL717A-SYEC717A     INDOUBT          00000DAC6538 SYSADM
V467-HAS LUWID USIBMSY.SYEC717A.BD6E5A52FFD1.0001=26
V466-THREAD HAS BEEN INDOUBT FOR    00:14:29
STL717A-SYEC717A     INDOUBT          00000DAC792C SYSADM
V467-HAS LUWID USIBMSY.SYEC717A.BD6E5B0A1F94.0001=28
V466-THREAD HAS BEEN INDOUBT FOR    00:09:45
DISPLAY INDOUBT REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THREAD' NORMAL COMPLETION

```

Example: Displaying information about a stored procedure thread that is waiting

This example shows a thread executing within a stored procedure and a thread waiting for a stored procedure to be scheduled. Assume that an application makes a call to stored procedure PROC1 and then to stored procedure PROC2. PROC2 is in a STOP QUEUE state.

The output for PROC1 while it is executing shows a status of SP in the ST column, which indicates that a thread is executing within a stored procedure:

```
-DISPLAY THREAD(*)
```

This command produces output similar to the following output:

```
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV402I - ACTIVE THREADS - 176
NAME      ST A  REQ ID      AUTHID  PLAN      ASID  TOKEN
BATCH    SP      3 RUNAPPL    SYSADM  PL01AP01 001D  43
V429 CALLING STORED PROCEDURE PROC1, LOAD MODULE LMPROC1
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION
```

The output for PROC2, while it is queued, shows a status of SW in the ST column, which indicates that a thread is waiting for a stored procedure to be scheduled:

```
-DISPLAY THREAD(*)
```

This command produces output similar to the following output:

```
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV402I - ACTIVE THREADS - 198
NAME      ST A  REQ ID      AUTHID  PLAN      ASID  TOKEN
BATCH    SW *  13 RUNAPPL    SYSADM  PL01AP01 001D  43
V429 CALLING STORED PROCEDURE PROC2, LOAD MODULE
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION
```

Example: Displaying information about an allied, non-distributed originating thread and its parallel tasks

This example shows an allied, non-distributed originating thread (TOKEN=30) that is established (allocated according to plan) in addition to all of its parallel tasks (PT), which are running on the same DB2 system. All parallel tasks are displayed immediately following their corresponding originating thread.

```
16.32.57          DB1G DISPLAY THREAD(*)
16.32.57 STC00090 DSNV401I DB1G DISPLAY THREAD REPORT FOLLOWS -
16.32.57 STC00090 DSNV402I DB1G ACTIVE THREADS -
NAME      ST A  REQ ID      AUTHID  PLAN      ASID  TOKEN
BATCH    T *   1 PUPPYDML    USER001  MYPLAN    0025  30
PT *    641 PUPPYDML    USER001  MYPLAN    002A  40
PT *    72 PUPPYDML    USER001  MYPLAN    002A  39
PT *    549 PUPPYDML    USER001  MYPLAN    002A  38
PT *    892 PUPPYDML    USER001  MYPLAN    002A  37
PT *    47 PUPPYDML    USER001  MYPLAN    002A  36
PT *    612 PUPPYDML    USER001  MYPLAN    002A  35
PT *    545 PUPPYDML    USER001  MYPLAN    002A  34
PT *    432 PUPPYDML    USER001  MYPLAN    002A  33
PT *    443 PUPPYDML    USER001  MYPLAN    002A  32
PT *    252 PUPPYDML    USER001  MYPLAN    002A  31
DISPLAY ACTIVE REPORT COMPLETE
16.32.58 STC00090 DSN9022I DB1G DSNVDT '-DISPLAY THREAD' NORMAL
COMPLETION
```

Example: Displaying detailed information TCP/IP threads from remote locations

This example shows the detail report for a DB2 client that uses TCP/IP to access a remote DRDA server.

```
-DISPLAY THREAD(*) LOCATION(*) DETAIL
```

This command produces output similar to the following output:

```
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV402I - ACTIVE THREADS -
NAME      ST A  REQ ID      AUTHID  PLAN      ASID  TOKEN
BATCH    TR *   4 AAAAAAA  ADMF001  AAAAAAA  0024  61
V441-ACCOUNTING=KEITH:AAAAAAA
V436-PGM=MYPLAN.AAAAAAA, SEC=1, STMT=1973, THREAD-INFO=SYSADM:***:
***:***
```

```

V444-USIBMSY.SYEC717A.BD6E60CE2BD3=61 ACCESSING DATA AT
( 1)STL717B-::FFFF:9.30.115.135..447
V447--INDEX SESSID          A ST TIME
V448--( 1) 1027:447          N R2 0522009445180
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION

```

Example: Displaying information about TCP/IP threads that use IPv6 addressing

This example shows the detail report for a client that supports IPv6 addressing and provides a 58-character extended correlation token to the DB2 server. That token is displayed in message V442.

```
-DISPLAY THREAD(*)
```

This command produces output similar to the following output:

```

DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV402I - ACTIVE THREADS -
NAME      ST A  REQ ID          AUTHID  PLAN      ASID TOKEN
SERVER    RA *   1 db2bp          ADMF001 DISTSERV 003A   3
V437-WORKSTATION=hornet, USERID=admf001,
APPLICATION NAME=db2bp
V442-CRTKN=1111:2222:3333:4444:5555:6666:7777:8888.65535.123456789ABC
V445-G91A0D32.EBCF.C0C44BCCFF4=3 ACCESSING DATA FOR
1111:2222:3333:4444:5555:6666:7777:8888
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION

```

Example: Displaying information about units of work with postponed back-out processing

This example shows information about units of work whose back-out processing has been postponed.

```
-DISPLAY THREAD (*) TYPE (POSTPONED)
```

This command produces output similar to the following output:

```

DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV431I - POSTPONED ABORT THREADS -
COORDINATOR          STATUS      RESET URID          AUTHID
BATCH                ABORT-P    000002FF98EA ADMF001
BATCH                ABORT-P    000002FF9000 ADMF001
DISPLAY POSTPONED ABORT REPORT COMPLETE
DSN9022I - DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION

```

Example: Displaying detailed information about threads that are processing under a user-defined function

This example shows the token for a thread that is processing a user-defined function. The token is 18.

```
-DISPLAY THREAD(*) DETAIL
```

This command produces output similar to the following output:

```

DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV402I - ACTIVE THREADS -
NAME      ST A  REQ ID          AUTHID  PLAN      ASID TOKEN
BATCH     T  *   231 DISTHD     ADMF001  0021    95
BATCH     SW *   38 INSERT     ADMF001  MYPLAN  0025    18
V436-PGM=CLIP74C1.UFIP74C1, SEC=0, STMT=0, THREAD-INFO=SYSADM:***:
*:*:*:*
V429 CALLING FUNCTION =SCIP7401.SP_UFIP74C1 ,
PROC=V61AWLM3, ASID=0030, WLM_ENV=WLMENV3
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION

```

Example: Displaying information about threads that are involved in an RRS unit of recovery

This example shows information about a thread that is involved in an RRS unit of recovery.

```
-DISPLAY THREAD(*) RRSURID(*)
```

This command produces output similar to the following output:

```
- 08.23.58 STC00149 DSNV401I ( DISPLAY THREAD REPORT FOLLOWS -  
- 08.23.58 STC00149 DSNV402I ( ACTIVE THREADS -  
- NAME ST A REQ ID AUTHID PLAN ASID TOKEN  
- RRSAF T 8 TGXID-111 ADMF001 TGXIDR 0023 35  
- V481-DB2 IS PARTICIPANT FOR RRS URID B4D0FC267EB02000000001101010000  
- DISPLAY ACTIVE REPORT COMPLETE  
- 08.23.58 STC00149 DSN9022I ( DSNVDT '-DIS THD' NORMAL COMPLETION
```

Example: Displaying information about threads when DB2 is the coordinator for an indoubt RRS unit of recovery

This example shows information about a thread where DB2 is the coordinator for an indoubt RRS unit of recovery. DB2 has committed the thread but has not been able to resolve the RRS UR with RRS.

```
-DISPLAY THREAD(*) TYPE(I) RRSURID(*)
```

This command produces output similar to the following output:

```
- 09.27.21 STC00185 DSNV406I ( INDOUBT THREADS -  
- COORDINATOR STATUS RESET URID AUTHID  
- UNKNOWN COMMITTED 123456789ABC UNKNOWN  
- V480-DB2 IS COORDINATOR FOR RRS URID C4D4FA267EB04000000001201020000  
00- DISPLAY INDOUBT REPORT COMPLETE  
- 09.27.21 STC00185 DSNV434I ( DSNVDT NO POSTPONED ABORT THREADS FOUND  
- 09.27.21 STC00185 DSN9022I ( DSNVDT '-DIS THD' NORMAL COMPLETION
```

Example: Displaying detailed information about threads in a distributed transaction

This example shows the XID for an active thread that is associated with an XA transaction manager:

```
-DISPLAY THREAD(*) DETAIL
```

This command produces output similar the following output:

```
#dis thd(*) det  
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -  
DSNV402I - ACTIVE THREADS -  
NAME ST A REQ ID AUTHID PLAN ASID TOKEN  
SERVERRX * 3 xidappl AdMF001 DISTSERV 0036 50  
V440-XID=53514C20 00000017 00000000 544D4442  
00000000 002F93DD A92F8C4F F3000000  
0000BD  
V445-G91E1F24.BAC1.01E098172410=50 ACCESSING DATA FOR  
( 1)::FFFF:9.30.31.36  
V447--INDEX SESSID A ST TIME  
V448-- ( 1) 447:47809 W R2 0522010250267  
DISPLAY ACTIVE REPORT COMPLETE  
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

Example: Displaying information about threads that are associated with a trusted context

This example shows a DISPLAY report of a thread associated with a trusted context.

```
-DIS THD(*)  
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -  
DSNV402I - ACTIVE THREADS - 133  
NAME ST A REQ ID AUTHID PLAN ASID TOKEN
```



```

BATCH      T * 10 JOB01      ADMF001  APPL01  0027  12
          V485-TRUSTED CONTEXT=DOMINOCONTEXT, SYSTEM AUTHID=SYSADM,
          ROLE=USRROLE
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I  - DSNVDT '-DIS THD' NORMAL COMPLETION

```

Example: Displaying information about threads during DB2 restart

This example shows the command during restart. A DSNV506I message indicates the state that DB2 is in, such as STARTING, ACTIVE, STOPPING. The message will also indicate the phase of restart or shutdown, if available.

```

-DIS THD(*) TYPE(SYSTEM)
DSNV401I  - DISPLAY THREAD REPORT FOLLOWS -
DSNV497I  - SYSTEM THREADS -
DB2 STARTING PHASE=Subsystem Startup Recovery Log Manager
NAME      ST A REQ ID      AUTHID  PLAN  ASID  TOKEN
V91A      N *  0 031.G1mTsk00  SYSOPR          002D  0
V490-SUSPENDED 05136-13:53:12.73  DSN7LSTK +00000D0C 05.45
V91A      N *  0 023.GCSCNM03  SYSOPR          002D  0
DISPLAY SYSTEM THREAD REPORT COMPLETE
DSN9022I  ) DSNVDT '-DIS THD' NORMAL COMPLETION

```

Example: Displaying information about suspended threads

This example shows the command executing while DB2 is active. Many system threads are suspended as shown in the DSNV490I messages. In addition, system thread 023.GSCN6 03 is currently executing the -DISPLAY THREAD(*) TYPE(SYSTEM) command.

```

-DIS THD(*) TYPE(SYSTEM)
DSNV401I  ) DISPLAY THREAD REPORT FOLLOWS -
DSNV497I  ) SYSTEM THREADS -
DB2 ACTIVE
NAME      ST A REQ ID      AUTHID  PLAN  ASID  TOKEN
V91A      N *  0 020.PEXCTL00  SYSOPR          002E  0
V490-SUSPENDED 05136-13:53:29.55  DSNTLCTL +000004F0 16.31
V91A      N *  0 010.PMICMS01  SYSOPR          002E  0
V490-SUSPENDED 05136-13:53:47.29  DSNB1CMS +000005E8 12.21
V91A      N *  0.010.PMITMR02  SYSOPR          002E  0
...
...
V91A      N *  0.004.JW007 01  SYSOPR          002D  0
V490-SUSPENDED 05136-13:54:03.60  DSNJW107 +000002B2 12.27
V91A      N *  0.004.JTIMR 00  SYSOPR          002D  0
V91A      N *  0.016.WVSMG 00  SYSOPR          002D  0
V490-SUSPENDED 05136-13:53:23.83  DSNWVSMG +00000534 01.54
V91A      N *  0.026.WVZXT 01  SYSOPR          002D  0
V91A      N *  0.016.WVSMT 01  SYSOPR          002D  0
V490-SUSPENDED 05136-13:53:48.10  DSNWVSMT +00000D7C 01.54
V91A      N *  0.023.GSCN6 03  SYSOPR          002D  0
V501-COMMAND EXECUTING: -DIS THD(*) TYPE(SYSTEM)
DISPLAY SYSTEM THREAD REPORT COMPLETE
DSN9022I  ) DSNVDT '-DIS THD' NORMAL COMPLETION
DISPLAY SYSTEM THREAD REPORT COMPLETE
DSN9022I  ) DSNVDT '-DIS THD' NORMAL COMPLETION

```

Example: Displaying information about threads when DB2 is stopping

This display shows the command executing while DB2 is stopping. The DSNV506I message identifies the phase of shutdown.

```

-DIS THD(*) TYPE(SYSTEM)
DSNV401I  - DISPLAY THREAD REPORT FOLLOWS -
DSNV497I  - SYSTEM THREADS -
NAME      ST A REQ ID      AUTHID  PLAN  ASID  TOKEN
V91A      N *  0 020.PEXCTL00  SYSOPR          002E  0
V490-SUSPENDED 05136-13:53:29.55  DSNTLCTL +000004F0 16.31
V91A      N *  0 010.PMICMS01  SYSOPR          002E  0
V490-SUSPENDED 05136-13:53:47.29  DSNB1CMS +000005E8 12.21

```

```

V91A      N * 0.010.PMITMR02    SYSOPR          002E  0
V91A      N * 0.022.SPQMON01    SYSOPR          002E  0
V91A      N * 0.014.RTSTST00    SYSOPR          002E  0
V490-SUSPENDED 05136-13:53:45.79 DSNB1TMR +00000B74 16.27
V91A      N * 0.010.PM2PCP01    SYSOPR          002E  0
V490-SUSPENDED 05136-13:53:47.33 DSNB1TMR +00000B74 16.27
V91A      N * 0 010.PM2CSX01    SYSOPR          002E  0
V490-SUSPENDED 05136-13:53:47.30 DSNB1TMR +00000B74 16.27
V91A      N * 0 010.PM2PCK02    SYSOPR          002E  0
V490-SUSPENDED 05136-13:54:17.31 DSNB1TMR +00000B74 16.27
V91A      N * 0 031.G1mTsk00    SYSOPR          002D  0
V490-SUSPENDED 05136-13:53:12.73 DSN7LSTK +00000D0C 05.45
V91A      N * 0 023.GCSCNM03    SYSOPR          002D  0
V91A      N * 0 004.JW007 01    SYSOPR          002D  0
V490-SUSPENDED 05136-13:54:31.62 DSNJW107 +000002B2 12.27
V91A      N * 0 023.GSCN6 03    SYSOPR          002D  0

```

```

V501-COMMAND EXECUTING: -DIS THD(*) TYPE(SYSTEM)
DISPLAY SYSTEM THREAD REPORT COMPLETE
DSN9022I ) DSNVDT '-DIS THD' NORMAL COMPLETION

```

Example: Displaying information about disconnected database access threads

This example shows information about an active database access thread that is disconnected.

```
-DISPLAY THREAD(*) DETAIL
```

This command produces output that is similar to the following output:

```

13.23.46          =dis thread(*) detail
13.23.46 STC00067 DSNV401I = DISPLAY THREAD REPORT FOLLOWS -
13.23.46 STC00067 DSNV402I = ACTIVE THREADS -
NAME      ST A  REQ ID          AUTHID  PLAN      ASID TOKEN
DISCONN  DA *   8 NONE          NONE    DISTSERV 003B   11
V471-USIBMSY.SYEC1DB2.C1C38BDE809D=11
DISPLAY ACTIVE REPORT COMPLETE
13.23.46 STC00067 DSN9022I = DSNVDT '-DIS THREAD' NORMAL COMPLETION

```

Example: Displaying information about inactive threads

This example shows information about an inactive connection.

```
-DISPLAY THREAD(*) TYPE(INACTIVE)
```

This command produces output that is similar to the following output:

```

- 13.27.47          =dis thread(*) type(inactive)
- 13.27.47 STC00067 DSNV401I = DISPLAY THREAD REPORT FOLLOWS -
- 13.27.47 STC00067 DSNV424I = INACTIVE THREADS -
- NAME      ST A  REQ ID          AUTHID  PLAN      ASID TOKEN
- SERVER   R2    0 javaw.exe    ADMF001 DISTSERV 003B   12
- V437-WORKSTATION=KFUKUSH, USERID=admf001,
- APPLICATION NAME=javaw.exe
- V445-G91E1686.EF04.080107212323=12 ACCESSING DATA FOR ::FFFF:9.30.22.134
- SERVER   R2    0 db2bp.exe    ADMF001 DISTSERV 003B   14
- V437-WORKSTATION=KFUKUSH, USERID=admf001,
- APPLICATION NAME=db2bp.exe
- V445-G91E1686.F104.080107212343=14 ACCESSING DATA FOR ::FFFF:9.30.22.134
- DISPLAY INACTIVE REPORT COMPLETE
- 13.27.47 STC00067 DSN9022I = DSNVDT '-DIS THREAD' NORMAL COMPLETION

```

Example: Displaying information about threads that use an accelerator server


This example shows the details for an active thread that is using an accelerator server.

```
-DISPLAY THREAD(*) ACCEL(ACCEL1) DETAIL
```

This command produces output similar to the following output:

```
-DIS THD(*) ACC(ACCEL1) DETAIL
DSNV401I # DISPLAY THREAD REPORT FOLLOWS -
DSNV402I # ACTIVE THREADS -
NAME      ST A   REQ ID          AUTHID   PLAN      ASID TOKEN
BATCH    AC *   231 BI           ADMF001  DSNTEP2   0053    55
V666 ACC=ACCEL1,ADDR=::FFFF:9.30.30.133..446:1076
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION
```

Related information:

 [DSNV401I \(DB2 Messages\)](#)

Chapter 38. -DISPLAY TRACE (DB2)

The DB2 command DISPLAY TRACE displays a list of active traces.

An additional option to this command and additional values for a few options of this command are not described here. They are intended for service and use under the direction of IBM support personnel.

Abbreviation: -DIS TRACE

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the value of the SCOPE option.

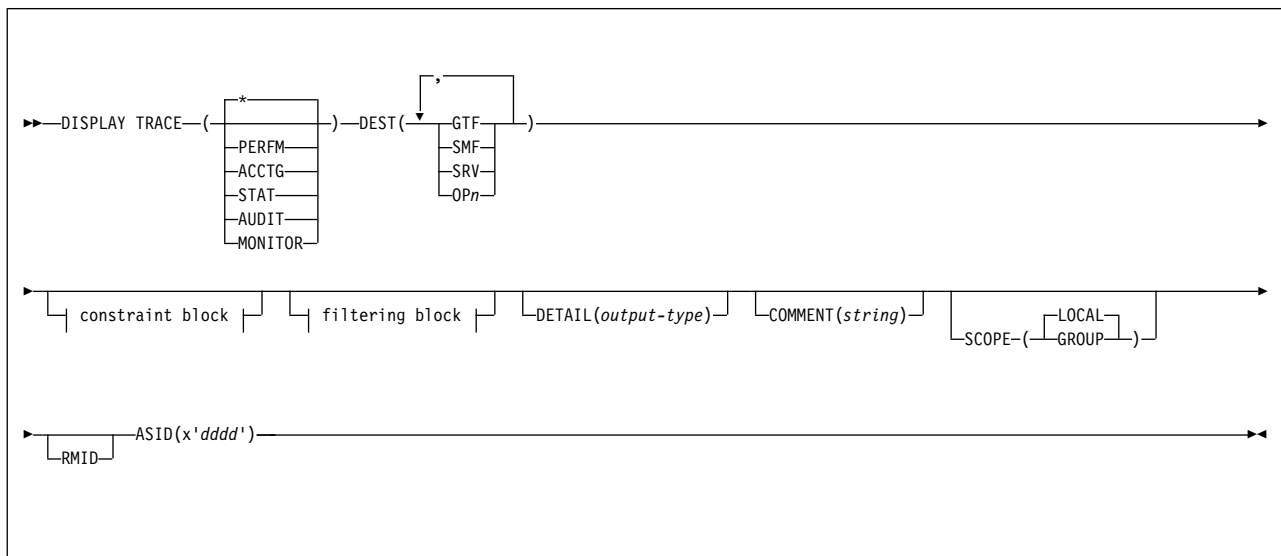
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

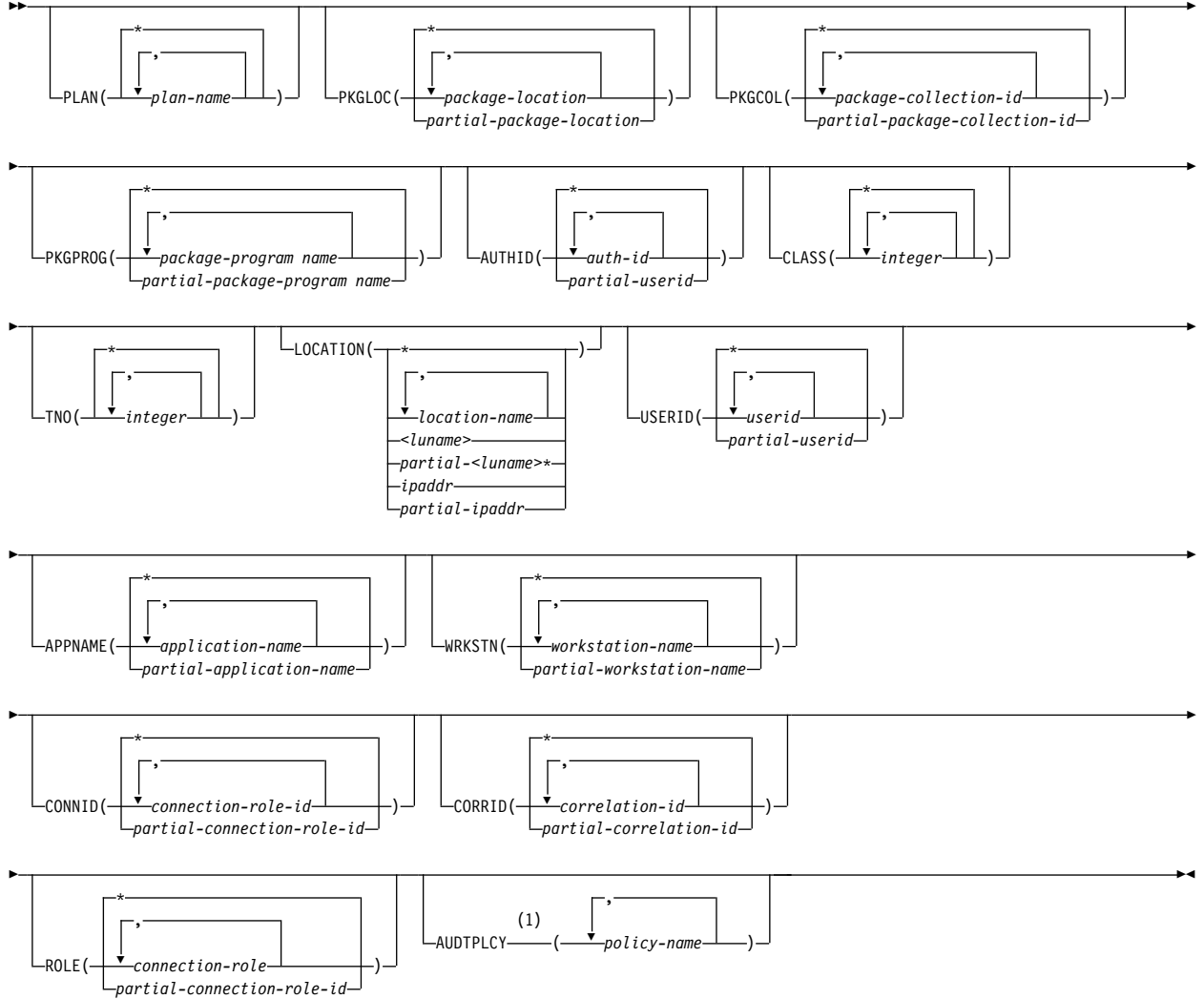
- DISPLAY privilege
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax

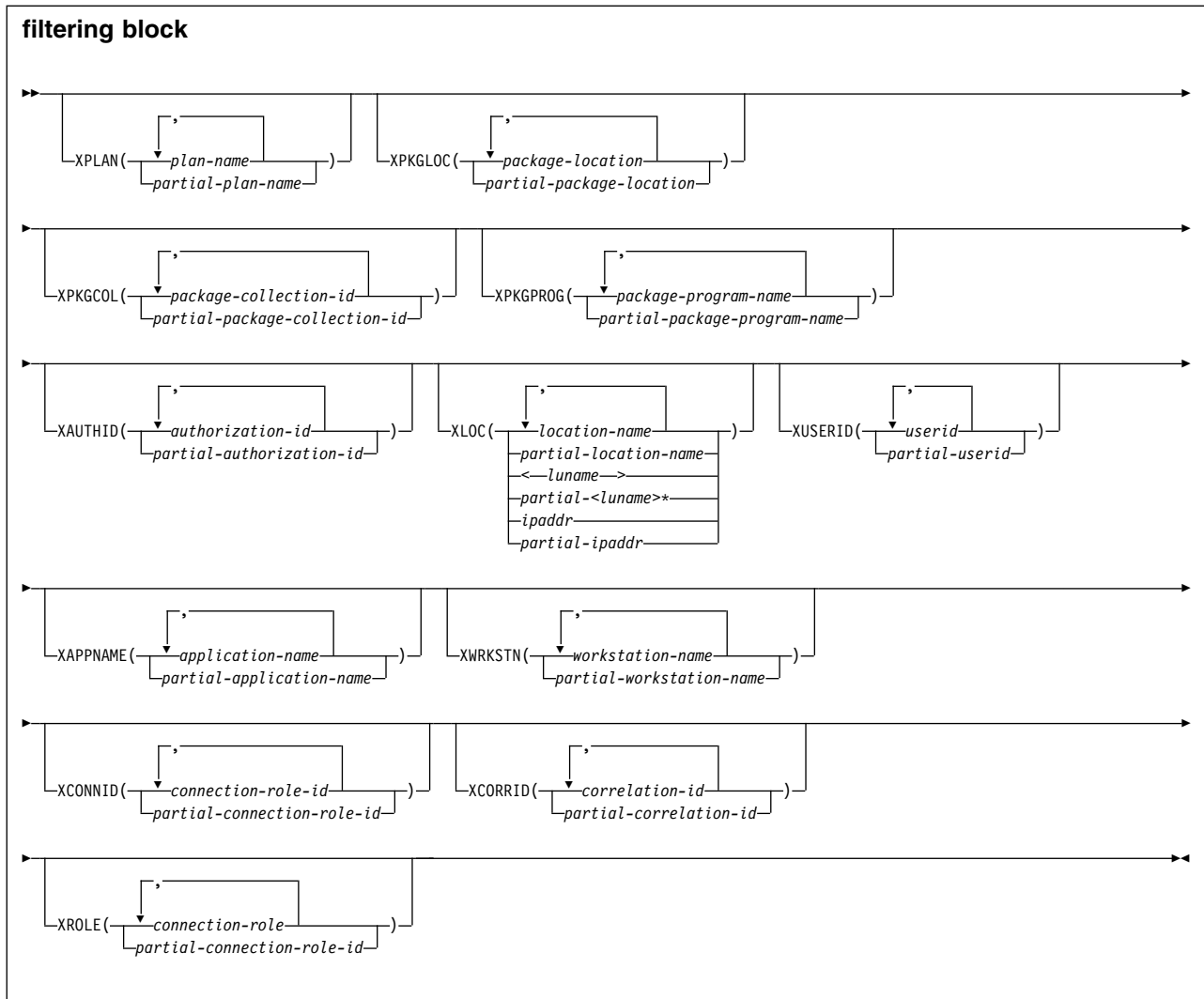


constraint block



Notes:

- 1 AUDTPLCY applies to trace type AUDIT. You cannot specify CLASS or IFCID with AUDTPLCY.



Option descriptions

None of the options are required. The command `DISPLAY TRACE` lists all active traces. Each option that is used, except `TNO`, limits the effect of the command to active traces that were started using the same option, either explicitly or by default, with exactly the same parameter values. For example, the following command lists only the active traces that were started using the options `PERFM` and `CLASS (1,2)`; it does *not* list, for example, any trace started using `CLASS(1)`.

```
-DISPLAY TRACE (PERFM) CLASS (1,2)
```

(*)

Does not limit the list of traces. The **default** is (*).

The `CLASS` option cannot be used with `DISPLAY TRACE (*)`.

Each of the following keywords limits the list to traces of the corresponding type.

Type (Abbrev)

Description

PERFM (P)

Performance records of specific events

ACCTG (A)

Accounting records for each transaction

STAT (S)

Statistical data

AUDIT (AU)

Audit data

MONITOR (MON)

Monitor data

DETAIL (*output-type*)

Limits the information that a trace displays based on the output type specified within parentheses.

The possible values for *output-type* are:

- 1 Display summary trace information: TRACE NUMBER, TYPE, CLASS, DEST
- 2 Display qualification trace information: TRACE NUMBER, AUTHID, LOCATION, PACKAGE, PLAN, TXACT, USERID, WORKSTATION
- 1,2 Display both summary and qualification information
- * Display both summary and qualification information

If no parameter follows DETAIL, type 1 trace information is displayed.

An additional column, QUAL, is also displayed, indicating whether the trace is qualified. Part of the summary trace information, the QUAL column can be used to determine if further qualification information for the trace is available. This information can be obtained by specifying DETAIL (2) or DETAIL (*). A QUAL column value of YES indicates that additional information for this particular trace exists in the qualification trace information; a value of NO indicates that no additional information for this trace exists.

COMMENT (*string*)

Specifies that comment *string* appears in the trace output, except for the output in the resident trace tables.

string is any character string; it must be enclosed between apostrophes if it includes a blank, comma, or special character. The comment does not appear in the display; it can be recorded in trace output, but only if commands are being traced.

SCOPE

Specifies the scope of the command.

(LOCAL)

Displays the trace for the local member only.

(GROUP)

Displays the trace for all members in the data sharing group.

DEST

Limits the list to traces that were started for particular destinations. More than one value can be specified, but do not use the same value twice. If you do not specify a value for DEST, DB2 does not use the destination of where trace output is recorded to limit the list of traces displayed.

Abbreviation: D

Possible values and their meanings are:

Value Trace destination

- GTF** The generalized trace facility
- SMF** The system management facility
- SRV** An exit to a user-written routine
- OP *n*** A specific destination. *n* can be a value from 1 to 8.

RMID

Specifies resource manager identifier. You can specify up to 8 valid RMIDs, which are one or two digit identifiers. You cannot specify RMID for accounting or statistic traces.

ASID(x'*dddd*') |

Specifies that information about the trace for an address space is to be displayed.

dddd is a four-byte hexadecimal address space ID (ASID).

CLASS(*integer* , ...)

Limits the list to traces that were started for particular classes.

The **default** is **CLASS(*_*)**, which does not limit the list.

TNO(*integer* , ...)

Limits the list to particular traces, identified by their trace numbers (1 to 32, 01 to 09). Up to eight trace numbers can be used. If more than one number is used, only one value each for PLAN, AUTHID, and LOCATION can be used.

The **default** is **TNO(*_*)**, which does not limit the list.

PLAN(*plan-name* , ...) or XPLAN(*plan-name* , ...)

Introduces a list of specific plans for which trace information is gathered. Use PLAN to constrain the trace to the specified plans or XPLAN to exclude the specified plans. You cannot use this option for a STAT trace.

The **default** is **PLAN(*_*)**.

(*)**

Displays a trace for all plans.

plan-name

The name of an application plan. You can use up to eight names. If you use more than one name, you can use only one value for AUTHID and LOCATION.

PKGLOC or XPKGLOC

Specifies the location name where the package is bound. Use PKGLOC to constrain the trace to the specified locations or XPKGLOC to exclude the specified locations.

PKGCOL or XPKGCOL

Specifies the package collection name. Use PKGCOL to constrain the trace to the specified collections or XPKGCOL to exclude the specified collections.

PKGPROG or XPKGPROG

Specifies the DBRM or program name. Use PKGPROG to constrain the trace to the specified programs or XPKGPROG to exclude the specified programs.

AUTHID(*authorization-id* , ...) or XAUTHID(*authorization-id* , ...)

Introduces a list of specific authorization IDs for which trace information is gathered. Use AUTHID to constrain the trace to the specified authorization IDs or XAUTHID to exclude the specified authorization IDs. The authorization IDs specified must be the primary authorization IDs. You cannot use this option for a STAT trace.

The **default** is AUTHID().

()

Displays a trace for all authorization IDs.

authorization-id

Specifies an authorization ID. You can use up to eight identifiers. If you use more than one identifier, you can use only one value for PLAN and LOCATION.

LOCATION(*location-name* , ...) or XLOC(*location-name* , ...)

Specifies a list of location names for which trace information is gathered. Use LOCATION to constrain the trace to the specified locations or XLOC to exclude the specified locations. The use of the LOCATION or XLOC option precludes tracing threads that have no distributed data relationship.

location-name

Identifies the DB2 subsystems whose distributed threads you want to trace. Activates the DB2 trace for the remote TCP/IP or SNA location that you specify by *location-name*.

You can specify up to eight locations. You can specify only one location if you use more than one plan name or authorization ID.

<*luname*>

Activates the DB2 trace for the remote clients that are connected to DDF through the remote SNA LU name that you specified in *luname*.

ipaddr

Activates the DB2 trace for the remote clients that are connected to DDF through the remote TCP/IP host. *ipaddr* is the IP address.

(*)

Indicates that you want to display trace events that occur under distributed threads regardless of which location they are connected to. Specifying the local location name is equivalent to specifying LOCATION(*).

Clients other than DB2 for z/OS: DB2 for z/OS does not receive a location name from clients that are not DB2 for z/OS subsystems. To display a trace for a client that is not a DB2 for z/OS subsystem, enter its LUNAME or IP address. Enclose the LUNAME by the less-than (<) and greater-than (>) symbols. Enter the IP address in the form *nnn.nnn.nnn.nnn*. For example, to display a trace for a client with the LUNAME of LULA, enter the following command:

```
-START TRACE (*) LOCATION (<LULA>)
```

To display a trace for a client with the IP address of 123.34.101.98, enter the following command:

```
-DISPLAY TRACE (*) LOCATION (::FFFF:123.34.101.98)
```

USERID or XUSERID

Specifies the user ID. Use **USERID** to constrain the trace to the specified user IDs or **XUSERID** to exclude the specified user IDs. You can specify multiple values and wildcard values as described in Usage notes.

APPNAME or XAPPNAME

Specifies the application name. Use **APPNAME** to constrain the trace to the specified applications or **XAPPNAME** to exclude the specified applications. You can specify multiple values and wildcard values as described in Usage notes.

| **WRKSTN or XWRKSTN**

| Specifies the workstation name. Use **WRKSTN** to constrain the trace to the
| specified workstations or **XWRKSTN** to exclude the specified workstations.
| You can specify multiple values and wildcard values as described in Usage
| notes.

| **CONNID or XCONNID**

| Specifies the connection ID. Use **CONNID** to constrain the trace to the
| specified connections or **XCONNID** to exclude the specified connections.

| **CORRID or XCORRID**

| Specifies the correlation ID. Use **CORRID** to constrain the trace to the specified
| correlation IDs or **XCORRID** to exclude the specified correlation IDs.

| **ROLE or XROLE**

| Specifies the connection roles. Use **ROLE** to constrain the trace to the specified
| roles or **XROLE** to exclude the specified roles.

| **AUDTPLCY**

| A 128-byte area that contains the policy name. You can specify up to eight
| audit policy names. **AUDTPLCY** applies to trace type **AUDIT**. You cannot
| specify **CLASS** or **IFCID** with **AUDTPLCY**.

Usage notes

*Displaying traced threads using the * wildcard:* You can use the wildcard suffix, **"**"** to filter threads that you want to display. For example, if you specify **"-DISPLAY TRACE PLAN (A,B,C*)"**, DB2 will display traces A, B, CDE, CDEFG, CDEFGH, and so on. It will display traced threads **"A"**, **"B"** and all threads starting with **"C"**.

Displaying traced threads using the positional, () wildcard: You can use the positional wildcard, which is represented by the, **"_"** character, to display traced threads when you want the wildcard in the middle, or when you want to trace threads of a specific length. For example, if you specify **"-DISPLAY TRACE PLAN (A_B)"**, all display all traced threads that have **"A"** as the first character, any character for the **"_"** and **"C"** as the plan filter.

Displaying multiple traced threads at once using wildcards: You also have the option of displaying multiple traced threads based on multiple trace qualifications. For example, you can specify, **"-DISPLAY TRACE PLAN (A*, B*, C*)"** to simultaneously display all traced threads for plan that start with **"A"**, **"B"**, and **"C"**. The wildcard character, **"**"** will display all traced threads. You can specify more complex combinations such as, **"-DISPLAY TRACE PLAN (A_B*, C*, AND C/_D*)"**, which will return all threads that:

- begin with **"A"**, have a one character wild card as the second character in the thread, have a **"B"** as the third character in the thread, and end with any type or number of characters (**ADBIO P, AOBTYJDP**)
- begin with **"C"**, and end with any combination of characters (**CDE, CGHKO**)
- begin with **"C_D"** and end with any type of character combination (**C_DEFGH, C_DIMNOP**)

All of the possible thread combinations listed above will be returned with the command above.

You have the ability to filter multiple threads at the same time, setting specific criteria for the trace. For example, you can specify, **"-DISPLAY TRACE PLAN (A) USERID (B)"**. This will display all traced threads where the plan thread is **"A,"** and

the user ID is "B." **Note:** When displaying traced threads, you can only specify one thread criteria for each filter for the display trace command. For example, you can specify `"-DISPLAY TRACE PLAN(A,B) USERID (B) WRKSTN (E),"` but you cannot specify `"-DISPLAY TRACE PLAN(A, B) USERID(A, B) WRKSTN(E)"` because, in this example, two of the filter qualifications have two elements defined to be traced, and DB2 only allows for one attribute to have more than one trace element to be defined per trace.

Filtering traced threads that you want to display using exclude functionality:

When you specify an "X" with any constraint keyword, such as "XPLAN", when you are filtering threads, you are using the exclude functionality for the display trace command. You have the option of excluding specific types of threads you want to display when you are running trace commands. You can use the "X" character to exclude specific combinations of characters when you are running a display trace command. For example, you can specify `"-DISPLAY TRACE XPLAN(A),"` to display all traced threads EXCEPT "A". In this instance B, BCD, BCDE, or CD could possibly be returned.

You also have the option of excluding multiple types of threads from your trace. For example, you can specify, `"-DISPLAY TRACE XPLAN (A*, B*)"` to display all traced threads EXCEPT those starting with "A", with any combination of characters following "A", and all those characters starting with "B", with any combination of characters following "B". Note: Specifying XPLAN (*) will exclude all threads from your search, and is not allowed. You also cannot use the wildcard in the middle of a display trace command with exclude functionality, such as, `"-DISPLAY TRACE XPLAN (A*C)."` You can, however, specify `"-DISPLAY TRACE XPLAN (A_ _ C *)"`, which will return all threads EXCEPT those starting with "A", a variety of TWO characters next, a "C" in the fourth space, and a variety of characters at the end. The wildcard symbol cannot be placed in the middle of trace criteria.

You have the ability to display two traces at once, in order to help you optimize your tracing capabilities. For example, you can specify `(-DISPLAY TRACE XPLAN (A, B, C) USERID (D))"`. This tells DB2 to display all threads that are being traced for "plan" EXCEPT threads "A", "B", or "C", only where the user ID = "D".

Combining trace qualifiers: You can customize the threads you trace by commanding DB2 to trace specific threads, while excluding other specific threads. For example, you can specify, `"-DISPLAY TRACE USERID(A,B) XPLAN (C)"`. This criteria only traces threads where the user ID is equal to "A" or "B", and plan is NOT equal to "C". In this example, a thread where the user ID is "A" and the plan is equal to "D" would pass, and be traced, but a thread where the user ID is "A" and plan is "C" would not pass, and would not be traced.

You can introduce wildcards into your display trace commands to add more customization to your traces. For example, you can specify `"-DISPLAY TRACE PLAN (C*) USERID (Z, X) XPLAN (C, D, E)"`. In this example, for the thread to be traced, the plan must begin with "C," the user ID must be equal to "Z" or to "X," and the plan cannot be "C," "D," or "E." So a plan of "CB," with a user ID of "Z" would pass, and the thread being traced would be displayed, but plan C with a user ID of "X" would fail because the command specifies not to trace threads where the plan is "C", without additional characters in the thread.

DISPLAY TRACE return code when there are no active traces: If you run DISPLAY TRACE in a batch environment, and no traces are active, the return code for the job step is 12.

Output

Message DSNW127I (DB2 Messages) indicates the beginning of the output of the command.

Examples

Example: Listing all traces that have a specific destination

The following command lists all traces that have the generalized trace facility as their only destination.

```
-DISPLAY TRACE (*) DEST (GTF)
```

The output is similar to this output:

```
- 15.05.15          @DISPLAY TRACE (*) DEST (GTF)
- 15.05.15 STC00042 DSNW127I @ CURRENT TRACE ACTIVITY IS -
- TNO TYPE  CLASS      DEST QUAL IFCID
- 04 PERFM  01,02,03,23 GTF NO
- *****END OF DISPLAY TRACE SUMMARY DATA*****
- 15.05.15 STC00042 DSN9022I @ DSNWVCM1 '-DISPLAY TRACE' NORMAL COMPLETION
```

Example: Listing traces of a specific trace class

The following command lists all active performance traces.

```
-DISPLAY TRACE=P
```

The output is similar to this output:

```
- 15.10.12          @DISPLAY TRACE=P
- 15.10.12 STC00042 DSNW127I @ CURRENT TRACE ACTIVITY IS -
- TNO TYPE  CLASS      DEST QUAL IFCID
- 04 PERFM  01,02,03    GTF NO
- 05 PERFM  08          GTF NO
- *****END OF DISPLAY TRACE SUMMARY DATA*****
- 15.10.12 STC00042 DSN9022I @ DSNWVCM1 '-DISPLAY TRACE' NORMAL COMPLETION
```

Example: Listing all traces for threads that are connected to a specific location

The following command lists all active audit traces for threads that are connection to DB2 subsystem with location name USIBMSTODB23.

```
-START TRACE(AUDIT) LOCATION(USIBMSTODB23)
```

The output is similar to this output:

```
- 15.21.03 STC00042 DSNW127I @ CURRENT TRACE ACTIVITY IS -
- TNO TYPE  CLASS      DEST QUAL IFCID
- 02 AUDIT  01          GTF YES
- 03 AUDIT  01          SMF YES
- *****END OF DISPLAY TRACE SUMMARY DATA*****
- 15.21.03 STC00042 DSN9022I @ DSNWVCM1 '-DISPLAY TRACE' NORMAL COMPLETION
```

Example: Listing all traces for a specific audit policy

The following command lists trace information for all audit traces that have GTF as their only destination and audit policy AUDITADMIN.

```
-DISPLAY TRACE (AUDIT) DETAIL (2) DEST (GTF) AUDTPLCY (AUDITADMIN)
```

The output is similar to this output:

```
- 15.46.36 STC00042 DSNW143I @ CURRENT TRACE QUALIFICATIONS ARE -
- 15.46.36 STC00042 DSNW152I @ BEGIN TNO 04 QUALIFICATIONS:
- NO QUALIFICATIONS
- END TNO 04 QUALIFICATIONS
- 15.46.36 STC00042 DSNW185I @ BEGIN TNO 04 AUDIT POLICIES:
- ACTIVE AUDIT POLICY: AUDITADMIN
- END TNO 04 AUDIT POLICIES
```

```
- 15.46.36 STC00042 DSNW148I @ *****END OF DISPLAY TRACE QUALIFICATION
- DATA*****
- 15.46.36 STC00042 DSN9022I @ DSNWVCM1 '-DISPLAY TRACE' NORMAL COMPLETION
```

Example: Using an filtering clause to display traces that do not have a specific criterion

The following command lists all active performance traces that are being written to SMF, and are not tracing activity for plan DSNREXX.

```
-DISPLAY TRACE(PERFM) DEST(SMF) XPLAN(DSNREXX)
```

The output is similar to this output:

```
- 16.02.20 STC00042 DSNW127I @ CURRENT TRACE ACTIVITY IS -
- TNO TYPE CLASS DEST QUAL IFCID
- 05 PERFM 01,02,03,23 SMF YES
- *****END OF DISPLAY TRACE SUMMARY DATA*****
- 16.02.20 STC00042 DSN9022I @ DSNWVCM1 '-DISPLAY TRACE' NORMAL COMPLETION
```

Related reference:

```
-STOP TRACE (DB2)
-START TRACE (DB2)
-MODIFY TRACE (DB2)
```

Chapter 39. -DISPLAY UTILITY (DB2)

The DB2 command DISPLAY UTILITY displays the status of utility jobs, including utility jobs in a data sharing group.

The output from the command consists of informational messages only. One set of messages is returned for each job identified by the command. For utility jobs in a data sharing group, the output shows the member name of the system on which each utility job is running.

The status from the display represents the current status, except in a data sharing group when the utility is running on a member other than the one from which the command is issued. In that case, the status is current as of the last checkpoint.

Abbreviation: -DIS UTIL

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

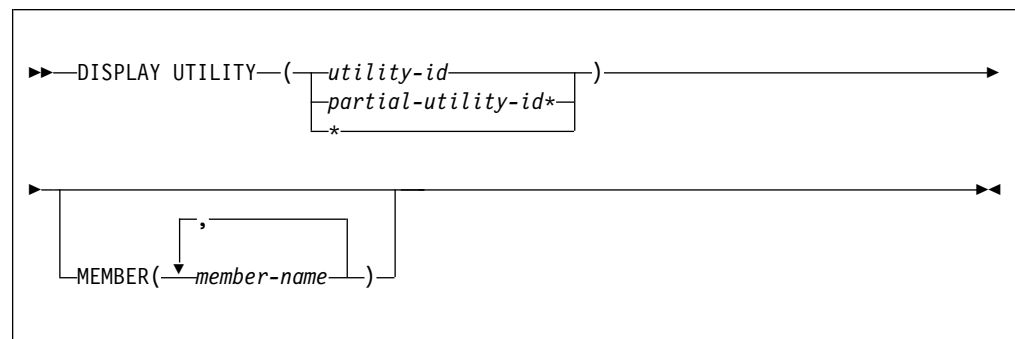
Data sharing scope: Group or member, depending on which option you choose.

To use the DISPLAY UTILITY command for BACKUP SYSTEM on a data sharing group, issue the command from the member on which the BACKUP SYSTEM utility is invoked. Otherwise, the current utility information is not displayed.

Authorization

None is required.

Syntax



Option descriptions

Use at least one of the following options but do not use the same option more than once.

(utility-id)

Identifies a single job by its utility identifier, the value given for the UID parameter when the job was created.

If *utility-id* was created by the DSNU CLIST by default, it has the form of *tso-userid.control-file-name* .

If *utility-id* was omitted when the utility job was created, *utility-id* has the form *userid.jobname*.

If *utility-id* contains lowercase letters or special characters, it must be enclosed in single quotation marks (').

(partial-utility-id*)

Identifies a set of utility jobs. A status message is shown for each utility identifier that begins with the characters of *partial-utility-id* .

For example, `-DISPLAY UTILITY(ABCD*)` shows the status of every utility job known to DB2 whose identifier begins with the characters ABCD.

(*)

Shows the status of all utility jobs known to DB2, including jobs currently running in a data sharing group.

MEMBER (member-name , ...)

Restricts the display for the identified utility jobs to specific members of the data sharing group. The default is to display utility jobs running on any member. In a non-data-sharing environment, the option is ignored.

One set of messages is returned for each job identified by the command.

Usage notes

DISPLAY status

The status displayed in the returned message is the status at the time the DB2 utility function received the command. Execution has proceeded, therefore the current state of the utility can be different from the state reported. For instance, the DISPLAY UTILITY command can indicate that a particular utility identifier is active, but, when the message is received by the requester, the utility job step could have terminated so that the utility identifier is no longer known to DB2.

Command response

In a data sharing environment, messages DSNU100I, DSNU105I, DSNU106I show the name of the member on which the utility job is running. If you specify a single member name in the MEMBER option and that member does not belong to the group, or if you specify a list of member names in the MEMBER option and none of those members belong to the group, the command fails and a message is issued.

Output

Message DSNU100I (DB2 Messages) indicates the beginning of the output of the command.

Examples

Example: Displaying the status of all utility jobs in a single DB2 subsystem

The following command displays status information for all utility jobs that are currently known to a DB2 subsystem.

```
-DISPLAY UTILITY(*)
```


The output is similar to the following output.

```
DSNU100I - DSNUGDIS - USERID = SAMPID
          MEMBER =
          UTILID = RUNTS
          PROCESSING UTILITY STATEMENT 1
          UTILITY = RUNSTATS
          PHASE = RUNSTATS   COUNT = 0
          STATUS = STOPPED
DSN9022I - DSNUGCC '-DISPLAY UTILITY' NORMAL COMPLETION
```

Example: Displaying the status of all utility jobs in a DB2 data sharing group

The following command displays status information for all utility jobs that are currently known to a DB2 data sharing group.

```
-DB1G DISPLAY UTILITY (*)
```

The output is similar to the following output.

```
DSNU100I -DB1G DSNUGDIS USER = SAMPID
          MEMBER = DB1G
          UTILID = RUNTS
          PROCESSING UTILITY STATEMENT 1
          UTILITY = RUNSTATS
          PHASE = RUNSTATS   COUNT = 0
          STATUS = STOPPED
DSNU100I -DB1G DSNUGDIS USER = SAMPID
          MEMBER = DB2G
          UTILID = CHKIX1
          PROCESSING UTILITY STATEMENT 8
          UTILITY = CHECK
          PHASE = UNLOAD   COUNT = 0
          STATUS = STOPPED
DSN9022I -DB1G DSNUGCC '-DB1G DISPLAY UTILITY' NORMAL COMPLETION
```

Example: Displaying the status of utilities on a specific data sharing member

The following command displays the status of utilities on data sharing member DB1G.

```
-DB1G DISPLAY UTILITY (*) MEMBER (DB1G)
```

The output is similar to the following output.

```
DSNU105I -DB1G DSNUGDIS - USERID = SYSADM 973
          MEMBER = DB1G
          UTILID = REORGCP
          PROCESSING UTILITY STATEMENT 1
          UTILITY = REORG
          PHASE = LOG   COUNT = 0
          STATUS = ACTIVE
DSNU347I -DB1G DSNUGDIS - 974
          DEADLINE = NONE
DSNU384I -DB1G DSNUGDIS - 975
          MAXRO = DEFER
          LONGLOG = CONTINUE
          DELAY = 1200 SECONDS
DSNU383I -DB1G DSNUGDIS - CURRENT ITERATION NUMBER = 4 976
WRITE ACCESS ALLOWED IN THIS ITERATION = YES
ITERATION BEFORE PREVIOUS ITERATION:
  ELAPSED TIME = 00:00:00
  NUMBER OF LOG RECORDS PROCESSED = 0
PREVIOUS ITERATION:
  ELAPSED TIME = 00:00:00
  NUMBER OF LOG RECORDS PROCESSED = 0
CURRENT ITERATION:
  ESTIMATED ELAPSED TIME = 00:00:00
  ACTUAL ELAPSED TIME SO FAR = 00:00:00
  ACTUAL NUMBER OF LOG RECORDS BEING PROCESSED = 0
CURRENT ESTIMATE FOR NEXT ITERATION:
```

ELAPSED TIME = 00:00:00
NUMBER OF LOG RECORDS TO BE PROCESSED = 0
DSN9022I -DB1G DSNUGCC '-DIS UTIL' NORMAL COMPLETION

Chapter 40. DSN (TSO)

The TSO command DSN starts a DSN session.

For a list of commands that you can issue under the TSO command DSN, see The DSN command and its subcommands.

During a DSN session, you can also enter DB2 commands, except for START DB2. DB2 commands must start with a hyphen (-). For detailed descriptions of the DB2 commands, see the commands with names that are preceded by the recognition character “-” and followed by“(DB2)” under DB2 and related commands.

You can also enter comments by starting them with an asterisk (*).

During a DSN session, you can also issue TSO commands, except for FREE, RUN, TEST, and TIME. To use TSO TEST to debug an application program, run it with the DSN command; for example:

```
TEST 'prefix.SDSNLOAD(DSN)' CP
```

Important: The ABEND subcommand is used for diagnostic purposes only, and is intended to be used only under the direction of IBM Software Support. Use it only when diagnosing a problem with DSN or DB2.

Percent commands are not recognized during a DSN session, they are only supported by the TSO command processor.

Environment

A DSN session runs under TSO in either foreground or background mode. When you run it in background mode, you are not prompted for corrections or additional required information.

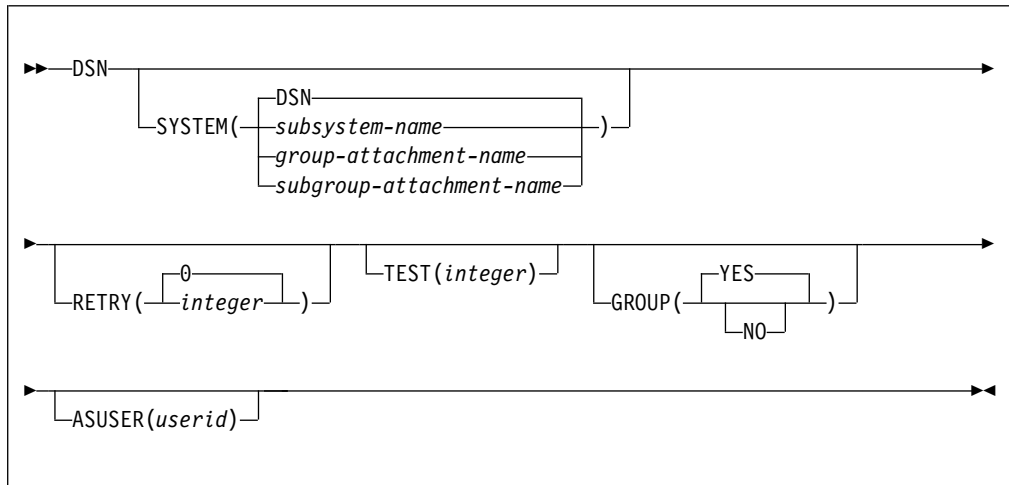
You can also start a DSN session from a CLIST running in either foreground or background mode.

Data sharing scope: Member

Authorization

None is required for the DSN command, but authorization is required for most subcommands.

Syntax



Option descriptions

None of the following options are required.

SYSTEM

(*subsystem-name*)

Specifies the name of the DB2 subsystem.

(*group-attachment-name*)

Specifies the group attachment name of the data sharing group.

(*subgroup-attachment-name*)

Specifies the subgroup attachment name of the data sharing group.

The **default** is **SYSTEM(DSN)**. This value can be modified during DB2 installation.

RETRY(*integer*)

Specifies the number (*integer*) of additional times connection to the DB2 subsystem should be attempted if DB2 is not up or the maximum number of batch connections has been reached when DSN is issued. Retries occur at 30-second intervals.

The **default** is **RETRY(0)**. The maximum number of retries is 120.

TEST(*integer*)

Specifies the last two digits (*integer*) of the module name in order to trace a single DSN module. Specify a number greater than 100 to trace all DSN modules. DSN trace information messages are written to the TSO SYSTSPRT DD statement, and optionally, to the DSNTRACE DD statement.

GROUP

(YES)

Specifies that group attachment processing is considered when the system is not active.

(**NO**) Specifies that group attach processing is not considered.

ASUSER(*userid*)

Specifies a user ID to associate with the trusted connection for the current DSN session.

A trusted connection is established if the primary authorization ID and jobname matches a trusted context defined in DB2. The user ID that you specify as ASUSER goes through the standard authorization and connection exit processing to pick up the primary and secondary IDs. If the primary authorization ID is allowed to use the trusted connection without authentication, DB2 establishes the trusted connection for the ASUSER user ID. The primary authorization ID, any secondary authorization IDs, and any role associated with the ASUSER user ID are now in effect for the trusted connection.

If the primary authorization ID associated with the user ID that you specify in the ASUSER option is not allowed to use the trusted connection or requires authentication information the connection request fails.

DB2 retains the ASUSER value only for the life of the DSN session.

Usage notes

Beginning a DSN session: Issue the DSN command to begin a DSN session, which allows you to enter DSN subcommands. The following rules govern the session:

- In foreground operation, you are prompted for input by the prompt string DSN at the terminal. In background mode, your input is read from the SYSTSIN data set.
- Except for delimited table names in the DCLGEN command, input in lowercase letters is changed to uppercase.
- If duplicate keywords of any subcommand are specified, only the last of these keywords is processed. For example, if both MEMBER(*dbrm-member-name1*) and MEMBER(*dbrm-member-name2*) are specified with BIND PLAN, DB2 receives only the latter, MEMBER(*dbrm-member-name2*).
- If ATTENTION (PA1) is pressed during a DSN session, and PROMPT is specified in the TSO user profile, message DSNE005 appears: EXECUTION IS INTERRUPTED, ENTER C TO CANCEL, OR ANY OTHER REPLY TO RESUME THE *subcommand* SUBCOMMAND.

If you enter C, the current subcommand is canceled and the current DB2 connection terminates; a new one is established, and another DSN prompt appears. Any other reply, except ATTENTION, causes the current subcommand to continue from the point at which it was interrupted.

If a DSN session is started from a CLIST, or a CLIST is executed under DSN, CONTROL PROMPT must be specified in the CLIST in order to receive message DSNE005.

- After a command is processed during a DSN session, you are prompted for input. That cycle continues until you end the session.
- You can end the session by taking one of the following actions:
 - Issue the END subcommand. Control is passed to TSO.
 - Press ATTENTION and respond to the message by pressing ATTENTION again.
 - Issue another DSN command. The old session ends and a new one begins.

DSN return code processing: At the end of a DSN session, register 15 contains the highest value used by any DSN subcommand in the session or by any program run using the RUN subcommand. Your run time environment might format that value as a return code. The value does not, however, originate in DSN.

Establishing trusted context using TSO and DB2I: DB2I runs under TSO using ISPF services. The DB2I facility provides an ISPF front end for tools such as SPUFI and DCLGEN and tasks such as preparing DB2 programs and binding plans and packages. The DB2I Defaults Panel includes the field AS USER. Use the AS USER field to specify an authorization name to use for the current session that is associated with the trusted connection. The trusted connection is established when the TSO logon ID matches the system authorization ID and jobname defined for a trusted context. The ASUSER field is always blank on entry to DB2I. If you enter a value in the AS USER field it is passed to all TSO attach (DSN) calls by using the ASUSER option of the DSN (TSO) command.

Examples

Example: Starting a DSN session with multiple retry attempts

The following command starts a DSN session. If the attempt to connect to DB2 fails, up to five retries, at 30 second intervals, are made.

```
DSN SYSTEM (DB2) RETRY (5)
```

Example: Starting a DSN session, running a program, and ending a DSN session

The following example shows the commands to start a DSN session, run a program, and end the DSN session.

```
TSO prompt : READY
USER enters: DSN SYS(SSSTR)
DSN prompt : DSN
USER enters: RUN PROGRAM(MYPROG)
DSN prompt : DSN
USER enters: END
TSO prompt : READY
```

Related concepts:

[↗](#) DSN command processor (DB2 Application programming and SQL)

Related tasks:

[↗](#) Issuing commands from TSO terminals (DB2 Administration Guide)

Related reference:

[↗](#) ADMIN_COMMAND_DSN stored procedure (DB2 SQL)

Chapter 41. DSNH (TSO CLIST)

The DSNH command procedure (a TSO CLIST) is a powerful yet easy method of preparing an application program for execution.

The DSNH command procedure (a TSO CLIST) prepares a program for execution, and executes it if it runs under TSO. By issuing a single command, you can select numerous options required for the preparation of an application and execute it under TSO.

DSNH processing is a sequential process that can include any of the actions listed in the following table.

Individual steps or a sequence of steps can be performed, and you can end the process at any point you choose. Any steps in the process that are skipped must have previously been completed successfully by DSNH.

Table 15. DSNH actions and the corresponding step names

For invoking the...	Use step name
PL/I macro processor	MP
DB2 precompiler	PC
CICS command language translator	TR
DSN BIND PLAN subcommand for binding a plan	BI
DSN BIND PACKAGE subcommand for binding a package	BP
Compiler or assembler for your program	CO
A C compiler prelink utility for including compile-time parameters	PL
Link-editor to produce an executable load module	LE
DSN RUN subcommand to execute the program	RU

Note: The step names are used in the heading of Table 17 on page 291.

Subsections:

- DSNH (TSO CLIST)
- DSNH (TSO CLIST)
- DSNH (TSO CLIST)
- DSNH (TSO CLIST)
- DSNH (TSO CLIST)

Environment

The DSNH CLIST can run in TSO foreground or in batch under the TSO terminal monitor program. DB2I uses the DSNH CLIST on the precompiler panel to control program preparation. You can pass DSNH parameters from DB2I panels on the "Other options" lines.

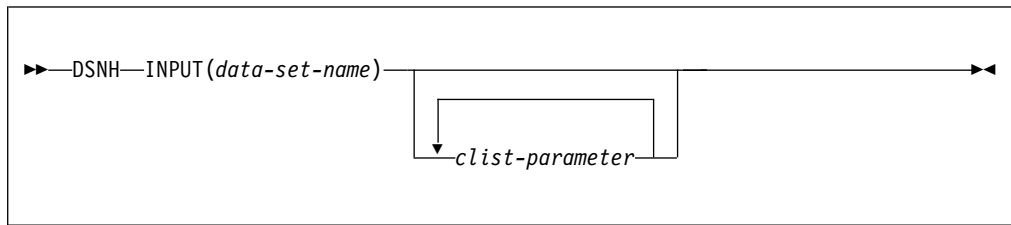
Data sharing scope: Member

Authorization

DSNH requires no special authorization. However, if DSNH binds a package or plan, or runs a plan, authorization is required to perform those actions:

- See the BIND PACKAGE authorization information for the privileges that are necessary to bind a package.
- See the BIND PLAN authorization information for the privileges that are necessary to bind a plan.
- See the RUN authorization information for the privileges that are necessary to run a plan.

Syntax



Summary of DSNH CLIST parameters

The CLIST parameters provide the processing options for each step; specify them when you execute DSNH. Some parameters are used for more than one step, as indicated in Table 17 on page 291. This table shows where each parameter is used, using the following notation:

- Y in any cell shows that the option listed at the beginning of the row is used in the step whose name appears at the top of the column.
- * in any cell indicates that the option listed at the beginning of the row is used in *another* step which affects the step whose name appears at the top of the column.

Notation of CLIST parameters for the BIND PLAN and BIND PACKAGE steps:

Many parameters of BIND PLAN and of BIND PACKAGE provide the same function and are spelled alike. CLIST parameters for BIND PLAN and BIND PACKAGE are differentiated from general parameters and from each other by prefixes. A parameter name prefixed by the letter B applies to the BIND PLAN subcommand; a parameter name prefixed by the letter P applies to BIND PACKAGE. The following table shows the possible variations for a single parameter name.

Table 16. DSNH CLIST prefixing rules

Parameter value	Function or subcommand	Example
<i>parameter</i>	If no prefix is specified, the parameter applies to a single function or subcommand.	DBRMLIB
B/ <i>parameter</i>	The prefix B is used to indicate that this variation of the parameter applies only to the BIND PLAN step.	B/DBRMLIB
P/ <i>parameter</i>	The prefix P is used to indicate that this variation of the parameter applies only to the BIND PACKAGE step.	P/DBRMLIB

In the following table, a prefix is separated from the DB2 parameter name by a slash (/). Refer to Table 15 on page 289 for an explanation of the two-letter step names.

Table 17. Summary of DSNH CLIST parameters

OPTIONS	MP	PC	TR	BI	BP	CO	PL	LE	RU
ACQUIRE				Y					*
P/ACTION				Y	Y				
ASMLIB						Y			
ASMLOAD						Y			
P/BDMEM				Y	Y				
P/BIND				Y	Y				
P/BLIB				Y					
P/BnLIB				Y					
P/BMEM				Y	Y				
CACHESIZE				Y					
CCLINK							Y		
CCLLIB								Y	
CCLOAD						Y			
CCMSGs						Y	Y		
CCOLIB							Y		
CCPLIB								Y	
CCPMSGs							Y		
CCSID		Y							
CCSLIB						Y			
P/CICS				Y	Y				
CICSCOB			Y					Y	
CICSLLIB			Y					Y	
CICSOPT			Y						
CICSPRE			Y					Y	
CICSPLIB			Y					Y	
CICSVER			Y					Y	
CICSXLAT			Y						
CLIB	Y					Y			
CnLIB	Y					Y			
COBICOMP								Y	
COBILINK								Y	
COBIPLNK							Y		
COBIPMSG							Y		
COBLIB								Y	
COBLOAD						Y			
COBSOM							Y		
COB2CICS								Y	

Table 17. Summary of DSNH CLIST parameters (continued)

OPTIONS	MP	PC	TR	BI	BP	CO	PL	LE	RU
COB2LIB								Y	
COB2LOAD						Y			
COMPILE						Y			
CONNECT		Y							
CONTROL	Y		Y	*		Y		Y	Y
COPTION	Y					Y			
COPY					Y				
COPYVER					Y				
CPPCLASS							Y		
CPPCLINK							Y		
CPPCLLIB								Y	
CPPCSLIB						Y			
CPPLLIB							Y		
CPPPMMSG							Y		
CPPSLIB						Y			
CPPUTIL						Y			
CURRENTDATA				Y	Y				
CURRENTSERVER				Y					
DATE		Y							
P/DBPROTOCOL				Y	Y				
P/B/DBRMLIB		Y		Y	Y				
DECARTH		Y							
DECIMAL		Y				*			
P/DEFER				Y	Y				
P/DEGREE				Y	Y				
DELIMIT		Y	Y			Y			
P/DISABLE				Y	Y				
DISCONNECT				Y	-				
P/DLIBATCH				Y	Y				
P/DYNAMICRULES				Y	Y				
P/ENABLE				Y	Y				
ENTRY								Y	
EXPLAIN				Y	Y				
P/FLAG	Y	Y	Y	Y	Y	Y			
FORTLIB								Y	
FORTLOAD						Y			
HOST	Y	Y	Y	*		Y		Y	Y
P/IMSBMP				Y	Y				
P/IMSMPP				Y	Y				
IMSPRE								Y	

Table 17. Summary of DSNH CLIST parameters (continued)

OPTIONS	MP	PC	TR	BI	BP	CO	PL	LE	RU
INPUT	Y	Y	*	*		Y		Y	Y
P/ISOLATION				Y	Y				
P/KEEPDYNAMIC				Y	Y				
LINECOUNT	Y	Y	Y			Y			
LINK								Y	
LLIB								Y	
LnLIB								Y	
LOAD								Y	Y
LOPTION								Y	
MACRO	Y		Y						
NEWFUN		Y							
NOFOR	Y								
P/NODEFER				Y	Y				
P/OPTHINT				Y	Y				
OPTIONS		Y	Y					Y	Y
OUTNAME	Y	Y				Y		Y	Y
P/OWNER				Y	Y				
PACKAGE					Y				
PARMS									Y
PASS		Y							
P/PATH				Y	Y				
PCLOAD		Y							
PKLIST				Y					
PLAN				Y					Y
PLIB		Y							
PnLIB		Y							
PLI2LIB								Y	
PLILIB								Y	
PLILOAD	Y					Y			
PLIPLNK								Y	
PLIPMSG								Y	
POPTION							Y		
PRECOMP		Y							
PRELINK							Y		
PRINT	Y	Y	Y			Y		Y	
PSECSPAC	Y	Y	Y			Y			Y
PSPACE	Y	Y	Y			Y			Y
P/QUALIFIER				Y	Y				
RCTERM	Y	Y	Y	Y	Y	Y	Y		Y
P/RELEASE				Y	Y				

Table 17. Summary of DSNH CLIST parameters (continued)

OPTIONS	MP	PC	TR	BI	BP	CO	PL	LE	RU
REMOTE					Y				
P/REOPT				Y	Y				
REPLVER					Y				
RETAIN				Y					
RUN		Y	Y					Y	Y
RUNIN									Y
RUNOUT									Y
SOMDLLI						Y	Y		
SOURCE	Y	Y	Y			Y			
SPACEUN	Y	Y	Y			Y			Y
SQL		Y							
SQLDELIM		Y							
SQLERROR					Y				
SQLRULES				Y					
STDSQL		Y							
SUFFIX	Y	Y							
SYSTEM				*	*				Y
TERM	Y	Y				Y		Y	
TIME		Y							
P/VALIDATE				Y	Y				
VERSION		Y							
WORKUNIT	Y	Y	Y			Y			
WSECSPAC	Y	Y	Y			Y			
WSPACE	Y	Y	Y			Y			
XLIB								Y	
XREF	Y	Y				Y		Y	

General parameter descriptions

Due to similarities in name and function, the CLIST parameters for BIND PLAN and BIND PACKAGE are described separately from the parameters in Table 18 on page 295. For a summary of:

- BIND PLAN parameters, refer to Table 19 on page 308
- BIND PACKAGE parameters, refer to Table 20 on page 311

The only parameter that is required on the DSNH statement is INPUT; the others are optional. In Table 18 on page 295:

- Parameter values must be enclosed between parentheses.
- Parameter values need not be enclosed between apostrophes, except in either of the following cases:
 - If the value is a list of tokens with separators, the value must be enclosed between apostrophes.

- If the value is a data set name, your user identifier is added as a prefix. To avoid the prefix, enclose the data set name between sets of three apostrophes.
- Most parameter values that are data set names (*dsname*) cannot include member names. Exceptions are noted in the parameter descriptions.
- Underlined values are defaults. Default names can be changed to names specific to your site when DB2 is installed.

Table 18. General DSNH CLIST parameters

Parameter	Value	Comments
ASMLIB	<i>dsname</i>	Specifies a data set to be used as the standard MACLIB for High Level Assembler. The default is "'SYS1.MACLIB'".
ASMLOAD	<i>dsname</i>	Specifies a data set that contains the High Level Assembler load module. <i>dsname</i> can include a member name. The default is "'SYS1.LINKLIB(ASMA90)'".
CCLINK	<i>dsname</i>	Specifies a data set that contains the IBM Language Environment® prelink editor utility invocation load module that is to be used for preparing C programs. <i>dsname</i> can include a member name. The default is "'CEE.SCEERUN(EDCPRLK)'".
CCLLIB	<i>dsname</i>	Specifies a data set that contains the linkage editor include modules for the C compiler routines. The default is "'CEE.SCEELKED'".
CCLOAD	<i>dsname</i>	Specifies a data set that contains the C compiler invocation load module. <i>dsname</i> can include a member name. The default is "'CBC.SCCNCMP(CCNDVR)'".
CCMSG	<i>dsname</i>	Specifies a data set that contains the C compiler messages. This data set is required only for C/370. <i>dsname</i> can include a member name. The default is "'EDC.V1R2M0.SEDCDMSG(EDCMSGE)'".
CCSID	<i>integer</i>	Specifies the CCSID for source SQL statements.
CCOLIB	<u>NONE</u> <i>dsname</i>	Specifies that the data set that contains C object modules is included during the execution of the prelink utility step.
CCPLIB	<u>NONE</u> <i>dsname</i>	Specifies a data set that contains include modules for PL/I routines. This parameter is used only for IBM C/370 Version 2 or earlier.
CCPMSG	<i>dsname</i>	Specifies a data set that contains the message library that is to be used by the IBM prelink editor when preparing C programs. The default is "'CEE.SCEEMSGP(EDCPMSG)'".
CCSLIB	<i>dsname</i>	Specifies a data set that contains the C compiler headers. The default is "'CEE.SCEEH.H'".

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
CICSOPT	<u>NONE</u> <i>option-list</i>	Specifies a list of additional CICS translator options. See the appropriate CICS application programming reference for information about translator options. The default , NONE, specifies no additional options.
CICSPRE	<i>prefix</i>	Specifies the prefix for the CICS libraries. The library names are: <i>prefix</i> .LOADLIB for translators <i>prefix</i> .PL1LIBn for PL/I include <i>prefix</i> .COBLIB for COBOL include Leave this parameter blank to use CICSLLIB, CICSPLIB, CICSCOB. The default is blank.
CICSLLIB	<i>dsname</i>	Specifies the CICS load library. To use this library, leave the CICSPRE parameter blank. The default is set on installation panel DSNTIP3.
CICSPLIB	<i>dsname</i>	Specifies the CICS PL/I library. To use this library, leave the CICSPRE parameter blank. The default is set on installation panel DSNTIP3.
CICSCOB	<i>dsname</i>	Specifies the CICS COBOL library. To use this library, leave the CICSPRE parameter blank. The default is set on installation panel DSNTIP3.
CICSVER	21 31 <u>33</u> 41	Specifies the CICS release. This field is ignored because current releases of CICS do not require DSNH to handle release-specific considerations.
CICSXLAT	NO YES	Specifies whether to execute the CICS command translator. This parameter is effective only if you use RUN(CICS). You cannot use this parameter with the MARGINS option of the translator. The default is YES. The DB2I panel default is NO.
CLIB C <i>n</i> LIB	<u>NONE</u> <i>dsname</i>	Specifies a data set that contains host language source statements to be included by the compiler or assembler. The parameters C <i>n</i> LIB (where <i>n</i> can be 2, 3, or 4) are extensions of CLIB, which is used to simplify passing a list of data set names. Use the default , NONE, to specify no data set.
COBICOMP	<i>dsname</i>	Specifies the IBM COBOL data set that is required for compilation. The default is "IGY.SIGYCOMP".
COBILINK	<i>dsname</i>	Specifies the IBM COBOL data set that is required for link edit. The default is "CEE.SCEELKED".
COBIPLNK	<i>dsname</i>	Specifies a data set that contains the IBM Environment prelink editor utility invocation load module that is to be used for preparing COBOL programs. <i>dsname</i> can include a member name. The default is "CEE.SCEERUN(EDCPRLK)".

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
COBIPMSG	<i>dsname</i>	Specifies a data set that contains the message library that is to be used by the IBM prelink editor when preparing COBOL programs. <i>dsname</i> can include a member name. The default is "CEE.SCEEMSGP(EDCPMSGE)".
COBLIB	<i>dsname</i>	Specifies the linkage editor include library that is to be used for OS/VS COBOL routines. This parameter is obsolete.
COBLOAD	<i>dsname</i>	Specifies a data set that contains the OS/VS COBOL compiler load module. This parameter is obsolete.
COBSOM	<i>dsname</i>	Specifies the IBM System Object Model (SOM) data set that is required for access to SOM objects. This parameter is obsolete.
COB2CICS	<i>dsname</i>	Specifies the linkage editor include library that is to be used for VS COBOL II CICS routines. This parameter is obsolete.
COB2LIB	<i>dsname</i>	Specifies the linkage editor include library that is to be used for the VS COBOL II or COBOL/370 routines. This parameter is obsolete.
COB2LOAD	<i>dsname</i>	Specifies a data set that contains the VS COBOL II or COBOL/370 compiler load module. This parameter is obsolete.
COMPILE	<u>YES</u> NO	Specifies whether to execute the compiler or assembler if the precompile step is successful.
CONNECT	(1) <u>(2)</u>	Specifies whether a CONNECT SQL statement should be processed as a type 1 CONNECT or a type 2 CONNECT statement. The DSNH(TSO CLIST) command does not accept the CT(1) and CT(2) abbreviations for this precompiler option. The default is CONNECT(2).
CONTROL	<u>NONE</u> CONLIST LIST SYMLIST	Specify to help you trace the allocation of non-existent data sets. Use this parameter if you have a problem without an obvious cause. CONLIST displays CLIST commands after substitution for symbols and before command execution. LIST displays TSO commands after substitution for symbols and before command execution. SYMLIST displays all executable statements (TSO commands and CLIST statements) before substitution for symbols.
COPTION	<u>NONE</u> <i>string</i>	Specifies a list of compiler or assembler options. For more information, refer to the manual that describes the compiler or assembler options for the specific language you are using. For a list of restrictions on some options, see COBOL Options. NONE specifies no options.

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
CPPCLASS	<i>dsname</i>	Specifies the data set that contains C++ class libraries. The default is "CBC.SCLBCPP".
CPPCLINK	<i>dsname</i>	Specifies the data set that contains the IBM Language Environment prelink editor utility invocation load module that is to be used for preparing C programs. The default is "CEE.SCEERUN(EDCPRLK)".
CPPCLLIB	<i>dsname</i>	Specifies the data set for the C linkage editor automatic call library that is used by the C++ compiler. The default is "CEE.SCEELKED".
CPPCSLIB	<i>dsname</i>	Specifies a data set that contains the C compiler headers that are used by the C++ compiler. The default is "CEE.SCEEH.H".
CPPLLIB	<i>dsname</i>	Specifies a data set that contains the C++ prelink automatic call library. The default is "CEE.SCEECPP".
CPPPMMSG	<i>dsname</i>	Specifies the data set that contains the message library that is to be used by the IBM prelink editor when preparing C++ programs. <i>dsname</i> can include a member name. The default is "CEE.SCEEMSGP(EDCPMSGE)".
CPPSLIB	<i>dsname</i>	Specifies the data set that contains C++ header files for class libraries. The default is "CBC.SCLBH.HPP".
CPPUTIL	<i>dsname</i>	Specifies the data set that contains procedures to set up and execute the C++ compiler. The default is "CBC.SCCNUTL".
DATE	ISO JIS USA EUR LOCAL	Specifies the format of date values that are to be returned, which overrides the format that is specified as the location default. The default is the value that is supplied when DB2 is installed, and is written in the data-only application defaults load module.

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
DBRMLIB	<p><u>DEFAULT</u> <i>dsname(member)</i> NONE</p>	<p>Specifies the partitioned data set, and an optional member name, that contains the DBRM library and member name that is used during the DB2 precompile step. Because you can specify individual DBRM member and library names during each individual phase, you must use the DBRMLIB parameter and associated prefixes to identify a specific phase.</p> <p>DBRMLIB specifies the DBRM library and member that is defined on the DBRMLIB DD statement during DB2 precompiler processing.</p> <p>DEFAULT indicates that the same DBRM library data set that is defined for the DB2 precompiler process (DBRMLIB(<i>parameter</i>)) is also used on the LIBRARY(<i>dsname</i>) subcommand keyword. If the precompiler DBRMLIB is not specified, the default generated DBRMLIB library that is based on the INPUT data set name is used.</p> <p><i>dsname</i> is generated using the DSNH OUTNAME parameter value, or its default, TEMP, with the constant DBRM appended to the prefix; for example, <i>outname</i> .DBRM or TEMP.DBRM.</p> <p><i>member</i> is obtained from the data set member name that is specified on the DSNH INPUT parameter or from the data set name as follows:</p> <ul style="list-style-type: none"> • Given INPUT(<i>outname.DBRM(dbrmmem)</i>): <ul style="list-style-type: none"> – <i>outname.DBRM(dbrmmem)</i> - If the member name is specified – <i>outname.DBRM(dbrm)</i> - If no member name is specified <p>NONE indicates that no LIBRARY(<i>dsname</i>) subcommand keyword is specified on invocation.</p>
DECARTH	<p><u>DEFAULT</u> 15 31</p>	<p>Specifies the maximum precision of decimal numbers.</p> <p>DEFAULT designates the value chosen, during installation, for the DECIMAL ARITHMETIC field on the APPLICATION PROGRAMMING DEFAULTS panel.</p> <p>A value of 15 specifies that decimal arithmetic operations on decimal values with precision 15 or less are performed in accordance with the existing rules for determining the precision and scale of the result.</p> <p>A value of 31 specifies that decimal arithmetic operations on decimal values with precision 15 to 31 are performed in accordance with new rules for determining the precision and scale of the result.</p> <p>DECARTH is ignored for Fortran.</p>
DECIMAL	<p>COMMA PERIOD</p>	<p>Specifies the decimal point indicator for decimal and floating point literals. DECIMAL is valid only for COBOL programs; PERIOD is forced for all other programs.</p> <p>COMMA makes the indicator a comma.</p> <p>PERIOD makes the indicator a period.</p> <p>The default is the value of the DECIMAL POINT field, set on the DB2 APPLICATION PROGRAMMING DEFAULTS panel during installation.</p>

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
DELIMIT	<p>DEFAULT</p> <p>APOST</p> <p>QUOTE</p>	<p>Specifies the APOST or QUOTE precompiler option to indicate the string delimiter that is used within host language statements. DELIMIT is effective only for COBOL programs; APOST is forced for all other programs.</p> <p>DEFAULT designates the value chosen during installation for the STRING DELIMITER field on the APPLICATION PROGRAMMING DEFAULTS panel.</p> <p>APOST specifies the apostrophe as the string delimiter for host language statements.</p> <p>QUOTE specifies a quotation mark as the string delimiter for host language statements.</p>
ENTRY	<i>entry-name</i>	<p>Specifies the entry point that is assigned by the linkage editor.</p> <p>The default depends on the host language and the value of RUN.</p> <ul style="list-style-type: none"> • For the PL/I language, the ENTRY value default is: <ul style="list-style-type: none"> – NONE if the RUN value is CICS – PLISTART for any other RUN value. • For assembler language, the ENTRY value default is DLITASM if the RUN value is IMS. • For COBOL, the ENTRY value default is DLITCBL if the RUN value is IMS. • For any other language, the ENTRY value default is NONE (no specified entry point) for any RUN value.
FLAG	<p>I</p> <p>C</p> <p>E</p> <p>W</p>	<p>Specifies the messages that you want to see. Use one of the following values to show messages of the corresponding types:</p> <p>I All informational, warning, error, and completion messages</p> <p>W Only warning, error, and completion messages</p> <p>E Only error and completion messages</p> <p>C Only completion messages</p>
FORTLIB	<i>dsname</i>	<p>Specifies the linkage editor include library that is to be used for Fortran routines.</p> <p>The default is "SYS1.VSF2FORT".</p>
FORTLOAD	<i>dsname</i>	<p>Specifies a data set that contains the VS Fortran compiler load module.</p> <p><i>dsname</i> can include a member name.</p> <p>The default is "SYS1.VSF2VCOMP(FORTVS2)".</p>
HOST	<p>ASM</p> <p>C</p> <p>CPP</p> <p>IBMCOB</p> <p>FORTRAN</p> <p>PLI</p>	<p>Defines the host language within which SQL statements are embedded.</p> <p>COBOL and COB2 are also acceptable values but are obsolete.</p> <p>If your program is one of the following types of programs, you cannot use DB2I to prepare it:</p> <ul style="list-style-type: none"> • A COBOL program that uses object-oriented extensions • A C++ program that uses object-oriented extensions and consists of more than one compilation unit <p>The default is the value of the LANGUAGE DEFAULT field, set on the DB2 APPLICATION PROGRAMMING DEFAULTS panel during installation.</p>

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
IMSPRE	<i>prefix</i>	Specifies the prefix for RESLIB, which is used for routines that are to be included by the linkage editor for IMS. The default is IMSVS.
INPUT	<i>dsname</i>	Specifies the data set that contains the host language source and SQL statements. <i>dsname</i> can include a member name.
LINECOUNT	<i>integer</i>	Specifies how many lines, including headings, are to be printed on each page of printed output. The default is 60.
LINK	<u>YES</u> <u>NO</u>	Specifies whether to execute the linkage editor after successful completion of compilation or assembly. YES indicates that the linkage editor is to be executed. The DSNHLI entry point from the precompiler is directed to the appropriate language interface module that is specified by the RUN parameter. NO indicates that linkage editor processing is to be bypassed.
LLIB <i>L</i> <i>n</i> LIB	<u>NONE</u> <i>dsname</i>	Specifies a data set that contains object or load modules that are to be included by the linkage editor. The parameters <i>L n</i> LIB (where <i>n</i> can be 2, 3, or 4) are extensions of LLIB, which is used to simplify passing a list of data set names. The LLIB and <i>L</i> <i>n</i> LIB libraries are concatenated with the XLIB library and the linkage editor include libraries for the specific host language. Object and load module libraries must not be mixed in this concatenation. Use the default , NONE, to specify no data set.
LOAD	<i>dsname</i>	Specifies a data set that is to contain the output from the linkage editor (the load module). <i>dsname</i> can include a member name. The default is RUNLIB.LOAD.
LOPTION	<u>NONE</u> <i>string</i>	Specifies a list of linkage editor options. For information about the options you can use, see the appropriate z/OS publication. Use the default , NONE, to give no options.
MACRO	<u>YES</u> <u>NO</u>	Specifies whether the macro preprocessor is to be executed before the precompilation of a PL/I program. If the PL/I macro processor is used, the PL/I *PROCESS statement must not be used to pass options to the PL/I compiler. The COPTION parameter of the DSNH command can be used to pass the needed options to the PL/I compiler.
NEWFUN	V8 V9 V10	Specifies whether to allow syntax for functions that are introduced by a new version of DB2. NEWFUN(NO) and NEWFUN(YES) are deprecated.

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
NOFOR	<u>NO</u> YES	<p>Specifies whether all FOR UPDATE OF clauses in static SQL statements are optional.</p> <p>When you specify NOFOR(YES), the FOR UPDATE OF clause is optional. Positioned updates can be made to any columns that the user has authority to update.</p> <p>When you specify NOFOR(NO), any query that appears in a DECLARE CURSOR statement must contain a FOR UPDATE OF clause if the cursor is used for positional updates. The clause must designate all the columns that the cursor can update.</p> <p>The option is implied when the STDSQL(YES) option is in effect.</p>
OPTIONS	<u>NO</u> YES	Specifies whether to print the options that are used when executing the precompiler or the CICS command translator with the output listing.
OUTNAME	<u>TEMP</u> <i>string</i>	<p>Specifies the prefix that is used to form intermediate data set names.</p> <p><i>string</i> must not be enclosed between apostrophes and must not have the same initial character as the <i>dsname</i> for INPUT. It cannot contain special characters.</p>
PARMS	<u>NONE</u> <i>string</i>	<p>Specifies a parameter string that is to be passed to the compiled program during its execution. This parameter is valid only if the run time execution environment requested is TSO. If CAF is specified as the run time execution environment, this parameter is ignored.</p> <p>Use the default, NONE, to pass no parameter string.</p>
PASS	ONE or 1 TWO or 2	<p>Specifies how many passes the precompiler is to use. One pass saves processing time, but requires that declarations of host variables in the program precede any reference to those variables. PASS has no effect for COBOL or Fortran; ONE is forced.</p> <p>The default is ONE or 1 for PL/I and C.</p> <p>The default is TWO or 2 for assembler.</p>
PCLOAD	<i>dsname</i>	<p>Specifies the precompiler load module.</p> <p><i>dsname</i> can include a member name.</p> <p>The default is '(DSNHPC)'.</p>
PLAN	<i>plan-name</i>	<p>Specifies the application plan that is created by the bind process.</p> <p>The default plan name is the first of the following available choices defined in the INPUT data set:</p> <ul style="list-style-type: none"> • DBRM member name • Leftmost qualifier <p><i>plan-name</i> must not be DEFAULT.</p> <p>If no name is found, a plan is not created.</p>
PLIB <i>Pn</i> LIB	<u>NONE</u> <i>dsname</i>	<p>Specifies the data set that contains host language source or SQL statements included by the SQL INCLUDE statement during precompilation. The parameters <i>Pn</i>LIB (where <i>n</i> can be 2, 3, or 4) are extensions of PLIB, which is used to simplify passing a list of data set names.</p> <p>Use NONE to specify no data set.</p>

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
PLI2LIB	<i>dsname</i>	Specifies the linkage editor common library that is used for PL/I routines. This parameter is obsolete.
PLILIB	<i>dsname</i>	Specifies the linkage editor base library that is used for PL/I routines. The default is "CEE.SCEELKED".
PLILOAD	<i>dsname</i>	Specifies a data set that contains the PL/I compiler load module. <i>dsname</i> can include a member name. The default is "IBM.SIBMZCMP(IBMZPLI)".
PLIPLNK	<i>dsname</i>	Specifies the data set that contains the IBM Environment prelink editor utility invocation load module that is to be used for preparing PL/I programs. <i>dsname</i> can include a member name. The default is "CEE.SCEERUN(EDCPRLK)".
PLIPMSG	<i>dsname</i>	Specifies the data set that contains the message library that is to be used by the IBM prelink editor for preparing PL/I programs. <i>dsname</i> can include a member name. The default is "CEE.SCEEMSGP(EDCPMSGE)".
POPTION	<u>NONE</u> <i>string</i>	Specifies a list of the C compiler language prelink utility options. Use the default , NONE, to give no options.
PRECOMP	<u>YES</u> <u>NO</u>	Specifies whether to precompile.
PRELINK	<u>YES</u> <u>NO</u>	Specifies whether to execute the C compiler prelink utility to make your program reentrant. This utility concatenates compile-time initialization information (for writable static) from one or more text decks into a single initialization unit. If this step is requested, it must follow the compiler step and precede the link-edit step. This parameter can apply to IBMCOB that also has a prelink step. Whether the prelink step applies to C or IBMCOB is determined by the choice of values C, CPP, or IBMCOB for the HOST parameter. Descriptions of the prelink process for C and IBMCOB are presented in their respective language publications. If PRELINK(YES) is specified or defaulted for a HOST language compiler that does not support the prelink utility, DB2 will issue warning message DSNH760I and prelink utility processing will be bypassed.

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
PRINT	NONE <i>dsname</i> LEAVE TERM	<p>Specifies where to send printed output, including the lists of options, source, cross-reference, error, and summary information.</p> <p>The default, NONE, omits printed output.</p> <p><i>dsname</i> specifies a data set that is to be used for the output. Do not enclose <i>dsname</i> between apostrophes. The current user profile is prefixed to <i>dsname</i>. The following suffixes are also added:</p> <ul style="list-style-type: none"> • SYSCPRT.LIST for PL/I macro listings (these listings are overwritten by the compiler listings) • PCLIST for precompiler listings • CXLIST for CICS command translator listings • LIST for compiler listings <p>The PRINT parameter is ignored for the compiler step when HOST(CPP) is specified.</p> <ul style="list-style-type: none"> • SYSOUT.PRELLIST for C prelink utility listings • LINKLIST for link-edit listings <p>LEAVE sends output to the specified print data set. You can allocate the print data set in one of the following ways:</p> <ul style="list-style-type: none"> • Dynamically • In the JCL that is used to run the DSNH CLIST (if in batch mode) • With the TSO ALLOCATE command (before running DSNH) <p>TERM sends output to the terminal.</p>
PSECSPAC	<i>integer</i>	<p>Specifies the amount of secondary space to allocate for print data sets, in the units given by SPACEUN.</p> <p>The default is 20.</p>
PSPACE	<i>integer</i>	<p>Specifies the primary size of the print data sets in the units given by SPACEUN.</p> <p>The default is 20.</p>
RCTERM	<i>integer</i>	<p>Specifies the minimum value of the return code from the precompile step that prevents execution of later steps.</p> <p>The default is 8.</p>

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
RUN	<u>TSO</u> or <u>YES</u> BATCH or <u>NO</u> CAF CICS IMS RRSAF	<p>Specifies whether to execute the compiled program if the previous steps are successful, and, if so, in which environment it executes. Your choice for the RUN parameter might affect your choice for LLIB.</p> <p>TSO or YES indicates that the application program is to be scheduled for execution in the TSO environment, and executes the compiled program.</p> <p>BATCH or NO indicates that the application program is not to be scheduled for execution, and defaults to TSO as the execution environment.</p> <p>CAF indicates that the application program is to be scheduled for execution in the call attachment facility environment. Specify BATCH or NO with CAF to indicate that the application program is not to be scheduled for execution, but to identify CAF as the execution environment. (BATCH,CAF) or (NO,CAF)</p> <p>CICS indicates that the application program is not to be scheduled for execution, and identifies CICS as the run time execution environment. CICS applications cannot run in TSO.</p> <p>IMS indicates that the application program is not to be scheduled for execution, and identifies IMS as the run time execution environment. IMS applications cannot run in TSO.</p> <p>RRSAF indicates that the application program is not to be scheduled for execution, and identifies RRSAF as the run time execution environment. RRSAF applications cannot run in TSO.</p>
RUNIN	<u>TERM</u> <i>dsname</i> LEAVE NONE	<p>Specifies where to get input for the RUN step.</p> <p>The default, TERM, gets input from the terminal.</p> <p><i>dsname</i> specifies a data set that is to be used for the input.</p> <p>LEAVE gets input from SYSIN if the only steps taken are LINK and RUN. LEAVE gets input from FT05F001 if the language is Fortran. Do not use LEAVE in any other case.</p> <p>NONE allocates no input file.</p>
RUNOUT	<u>TERM</u> <i>dsname</i> LEAVE NONE	<p>Specifies where to send output from the RUN step.</p> <p>The default, TERM, sends output to the terminal.</p> <p><i>dsname</i> specifies a data set to receive output.</p> <p>LEAVE sends output to SYSPRINT if the only steps taken are LINK and RUN. LEAVE sends output to FT06F001 if the language is Fortran. Do not use LEAVE in any other case.</p> <p>NONE allocates no output file for the RUN step.</p>
SOMDLLI	<i>dsname</i>	<p>Specifies the name that of the SOM/MVS DLL import library.</p> <p>This parameter is obsolete.</p>
SOURCE	<u>NO</u> <u>YES</u>	<p>Specifies whether the source code and diagnostics are to be printed with output from the precompiler, CICS command translator, and compiler.</p>

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
SPACEUN	<u>TRACK</u> CYLINDER	Specifies the unit of space for PSPACE and WSPACE. TRACK makes the space unit one track. CYLINDER makes the space unit one cylinder.
SQL	<u>DB2</u> <u>ALL</u>	Specifies how to interpret SQL statements and check syntax for use by either DB2 for z/OS or other database management systems. The default , DB2, indicates that SQL statements are to be interpreted and syntax is to be checked for use by DB2 for z/OS. SQL(DB2) is the recommended mode for DRDA access when the server is a DB2 subsystem. ALL indicates that SQL statements are to be interpreted for use by database management systems that are not DB2 for z/OS. SQL syntax checking is deferred until bind time so that the remote location can bind the resulting DBRM. When SQL(ALL) is in effect, the precompiler issues an informational message if SAA reserved words are used as identifiers. SQL(ALL) is the recommended mode if you have written your application to be executed in a environment that is not DB2 for z/OS. The default is SQL(DB2).
SQLDELIM	<u>DEFAULT</u> <u>APOSTSQL</u> <u>QUOTESQL</u>	Specifies the APOSTSQL or QUOTESQL precompiler option, to set the SQL string delimiter and, by implication, the SQL escape character within SQL statements. Whichever character is chosen to be the string delimiter, the other is used for the SQL escape character. This parameter is effective only for COBOL. For PL/I, Fortran, and assembler language programs, the precompiler forces the APOSTSQL option. DEFAULT designates the value that is chosen during installation for the SQL STRING DELIMITER field on the APPLICATION PROGRAMMING DEFAULTS panel. APOSTSQL specifies that the string delimiter is the apostrophe (') and the escape character is the quotation mark ("). QUOTESQL specifies that the string delimiter is the quotation mark (") and the escape character is the apostrophe (').
STDSQL	<u>NO</u> YES or 86	Specify whether to interpret SQL using a subset of ANSI rules. NO specifies that DB2 rules are used. YES or 86 automatically implies that the NOFOR option is used.
SUFFIX	<u>YES</u> <u>NO</u>	Specifies whether the TSO standard naming convention must be followed. That convention adds a TSO authorization ID prefix and a host language suffix to the name of the input data set (unless that name is enclosed between apostrophes, or already ends in the appropriate suffix). For example, names become <i>userid.name.COBOL</i> , <i>userid.name.PLI</i> , <i>userid.name.Fortran</i> , or <i>userid.name.ASM</i> .
SYSTEM	<i>subsystem-name</i>	Specifies the DB2 subsystem name as it is known to the z/OS operating system. The default is the installation-defined subsystem name (often DSN).

Table 18. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
TERM	<u>TERM</u>	Specifies where to send terminal output, including error information, error statements, and summary information.
	<i>dsname</i> LEAVE NONE	<p>The default, TERM, sends output to the terminal.</p> <p><i>dsname</i> specifies a data set that is to be used for terminal output. Do not enclose <i>dsname</i> between apostrophes. The following suffixes are added to <i>dsname</i> :</p> <ul style="list-style-type: none"> • PCTERM for precompiler output • LIST for compiler output <p>LEAVE sends the output to the current allocation for SYSTEM.</p> <p>NONE omits terminal output.</p>
TIME	ISO JIS USA	Specifies the format for time values that are to be returned, overriding the format that is specified as the location default.
	EUR LOCAL	There is no default because this option overrides the default previously specified.
VERSION	<i>version-id</i>	Specifies the name of the version ID for the program and associated DBRM during the DB2 precompile step.
	AUTO	<p>AUTO specifies that the consistency token is used to generate the version ID. If the consistency token is a timestamp, the timestamp is converted into ISO character format and used as the version identifier.</p> <p>The default is no version ID if specified at precompiler invocation.</p>
WORKUNIT	<i>unit</i>	Specifies the device to use for print and work data sets.
		<p><i>unit</i> can be a unit name or a device type.</p> <p>The default in batch mode is any eligible device.</p> <p>The default in any other mode is the UADS unit name for the current TSO user.</p>
WSECSPAC	<i>integer</i>	Specifies the amount of secondary space to allocate for work data sets, in the units given by SPACEUN.
		The default is 20.
WSPACE	<i>integer</i>	Specifies the primary size of the work data sets in the units given by SPACEUN.
		The default is 20.
XLIB	<i>dsname</i>	Specifies the linkage editor include library that is to be used for DB2 routines.
		The default is " <i>prefix</i> .SDSNLOAD".
XREF	<u>NO</u>	Specifies whether a sorted cross-reference listing of symbolic names that are used in source statements is to be printed with output from the precompiler.
	YES	
Note: Precompiler options do not affect ODBC behavior.		

DSNH/DSN subcommand summary

Table 19 and Table 20 on page 311 differentiate the functions that support BIND PLAN and BIND PACKAGE. Each table associates the DSNH CLIST parameter and its corresponding DSN BIND PLAN or BIND PACKAGE subcommand keyword, if any. In general:

- The function and value of a CLIST parameter is identical to that of its corresponding DSN subcommand keyword unless otherwise noted.
- A DSNH parameter value of NONE indicates that the corresponding DSN keyword is not specified on subcommand invocation. Exceptions are noted where applicable.

DSNH CLIST/BIND PLAN subcommand comparison

Table 19. DSNH CLIST/ BIND PLAN subcommand summary

DSNH CLIST		BIND PLAN subcommand		Comments
Parameter	Value	Keyword	Value	
ACQUIRE	<u>USE</u> ALLOCATE	ACQUIRE	<u>USE</u> ALLOCATE	
ACTION	<u>REPLACE</u> ADD	ACTION	<u>REPLACE</u> ADD	
BDMEM	<u>DEFAULT</u> ¹ <i>dbrm-member-name</i> NONE ²	MEMBER	<i>dbrm-member-name</i>	<p>¹ DBRM member name, which is obtained from one of the following sources, in the order listed:</p> <ul style="list-style-type: none"> • BDBRMLIB member name • DBRMLIB member name • INPUT member name, or generated using <i>dsname</i>. <p>² Keyword is not specified on subcommand invocation.</p>
BIND	<u>YES</u> ¹ NO ²	<i>(command-verb)</i>		<p>¹ Execute BIND PLAN subcommand.</p> <p>² Do not execute BIND PLAN subcommand.</p>
BLIB	<u>NONE</u> ¹ <i>dsname</i>	LIBRARY	<i>dbrm-pds-name</i>	¹ Keyword is not specified on subcommand invocation.
BnLIB ¹	<u>NONE</u> ² <i>dsname</i>	LIBRARY	<i>list of dbrm-pds-names</i>	<p>¹ <i>n</i> can be 2, 3, 4, 5, 6, 7, or 8. Specify the first data set name by using the BLIB parameter. Specify any additional data set names by using this parameter.</p> <p>² No additional data set names.</p>
BMEM ¹	<u>NONE</u> ² <i>list of dbrm-member-names</i>	MEMBER	<i>list of dbrm-member-names</i>	<p>¹ Specify the first DBRM member name using the BDMEM parameter and any additional member names individually using this parameter.</p> <p>² No additional DBRM member names.</p>

Table 19. DSNH CLIST/ BIND PLAN subcommand summary (continued)

DSNH CLIST		BIND PLAN subcommand		
Parameter	Value	Keyword	Value	Comments
CACHESIZE	<u>NONE</u> ¹ <i>decimal-value</i> ²	CACHESIZE	<i>decimal-value</i> ²	¹ The size is provided by the subsystem. ² Specify a size from 0 to 4096 bytes.
CICS	<u>NONE</u> ¹ <i>application-ids</i>	CICS	<i>application-ids</i>	¹ Keyword is not specified on subcommand invocation.
CURRENTDATA	YES <u>NO</u> NONE	CURRENTDATA	YES <u>NO</u>	
CURRENTSERVER	<u>NONE</u> ¹ <i>location-name</i>	CURRENTSERVER	<i>location-name</i>	
DBPROTOCOL	<u>NONE</u> DRDA PRIVATE	DBPROTOCOL	<u>DRDA</u> PRIVATE	If you specify PRIVATE, your application cannot include SQL statements that were added to DB2 after DB2 Version 7.
BDBRMLIB	<u>DEFAULT</u> ¹ <i>dsname(member)</i> NONE ²	LIBRARY	<i>dbrm-pds-name</i>	¹ The precompiler DBRMLIB data set is used. If the precompiler DBRMLIB is not specified, the default-generated DBRMLIB library that is based on the INPUT data set is used. ² Keyword is not specified on subcommand invocation.
DEFER	<u>NONE</u> ¹ PREPARE	DEFER	PREPARE	¹ Keyword is not specified on subcommand invocation.
DEGREE	<u>1</u> ANY	DEGREE	<u>1</u> ANY	
DISABLE	<u>NONE</u> BATCH CICS DB2CALL IMS DLIBATCH IMSBMP IMSMPP RRSAF	DISABLE	<u>NONE</u> BATCH CICS DB2CALL IMS DLIBATCH IMSBMP IMSMPP RRSAF	
DISCONNECT	<u>EXPLICIT</u> AUTOMATIC CONDITIONAL	DISCONNECT	<u>EXPLICIT</u> AUTOMATIC CONDITIONAL	
DLIBATCH	<u>NONE</u> ¹ <i>list of connection-ids</i>	DLIBATCH	<i>connection-name</i>	¹ Keyword is not specified on subcommand invocation.
DYNAMICRULES	<u>RUN</u> BIND	DYNAMICRULES	<u>RUN</u> BIND	
ENABLE	<u>NONE</u> * BATCH CICS DB2CALL IMS DLIBATCH IMSBMP IMSMPP RRSAF	ENABLE	<u>NONE</u> * BATCH CICS DB2CALL IMS DLIBATCH IMSBMP IMSMPP RRSAF	
EXPLAIN	<u>NO</u> YES	EXPLAIN	<u>NO</u> YES	
FLAG	<u>I</u> C E W	FLAG	<u>I</u> C E W	

Table 19. DSNH CLIST/ BIND PLAN subcommand summary (continued)

DSNH CLIST		BIND PLAN subcommand		
Parameter	Value	Keyword	Value	Comments
IMSBMP	<u>NONE</u> ¹ <i>imsid</i>	IMSBMP	<i>imsid</i>	¹ Keyword is not specified on subcommand invocation.
IMSMPP	<u>NONE</u> ¹ <i>imsid</i>	IMSMPP	<i>imsid</i>	¹ Keyword is not specified on subcommand invocation.
ISOLATION	<u>RR</u> RS CS UR	ISOLATION	<u>RR</u> RS CS UR	
KEEPDYNAMIC	<u>NO</u> YES	KEEPDYNAMIC	<u>NO</u> YES	
NODEFER	<u>NONE</u> ¹ <u>PREPARE</u>	NODEFER	<u>PREPARE</u>	¹ Keyword is not specified on subcommand invocation.
OPTHINT	<u>(' ') (' hint-id ')</u>	OPTHINT	<u>(' ') (' hint-id ')</u>	
OWNER	<u>NONE</u> ¹ <i>authorization-id</i>	OWNER	<i>authorization-id</i>	¹ Keyword is not specified on subcommand invocation.
PATH	<i>(schema-name) (</i> USER <i>)</i> <i>(schema-name,</i> USER ...)	PATH	<i>(schema-name) (</i> USER <i>)</i> <i>(schema-name,</i> USER ...)	
PKLIST	<u>NONE</u> ¹ <i>list of collection-ids and package- names</i>	PKLIST	<i>list of collection-ids and package- names</i>	¹ The package names are not specified on subcommand invocation.
PLAN	<i>plan-name</i> ¹	PLAN <i>(primary-keyword)</i>	<i>plan-name</i>	¹ <i>plan-name</i> must not be DEFAULT. The default <i>plan-name</i> is the first of the following available choices that are defined in the INPUT data set: <ul style="list-style-type: none">• DBRM member name• Left-most qualifier If no name is found, a plan is not created.
QUALIFIER	<u>NONE</u> ¹ <i>implicit-qualifier</i>	QUALIFIER	<i>qualifier-name</i>	¹ Keyword is not specified on subcommand invocation.
RELEASE	<u>COMMIT</u> <u>DEALLOCATE</u>	RELEASE	<u>COMMIT</u> <u>DEALLOCATE</u>	
REOPT	<u>NONE</u> ¹ VARS	REOPT	<u>NONE</u> ALWAYS <u>ONCE</u> AUTO	¹ Keyword is not specified on subcommand invocation.
RETAIN	<u>NO</u> ¹ YES ²	RETAIN		¹ Keyword is not specified on subcommand invocation. ² Keyword is specified on subcommand invocation.
SQLRULES	<u>DB2</u> STD	SQLRULES	<u>DB2</u> STD	
VALIDATE	<u>RUN</u> BIND	VALIDATE	<u>RUN</u> BIND	

DSNH CLIST/BIND PACKAGE subcommand comparison

Table 20. DSNH CLIST/ BIND PACKAGE subcommand summary

DSNH CLIST		BIND PACKAGE subcommand		
Parameter	Value	Keyword	Value	Comments
PACTION	<u>REPLACE</u> ADD	ACTION	<u>REPLACE</u> ADD	
PBIND	<u>NO</u> ¹ YES ²	(<i>command-verb</i>)		¹ Do not execute BIND PACKAGE subcommand. ² Execute BIND PACKAGE subcommand.
PCICS	<u>NONE</u> ¹ <i>application-ids</i>	CICS	<i>application-ids</i>	¹ Keyword is not specified on subcommand invocation.
COPY	<u>NONE</u> ¹ <i>collection-id.</i> <i>package-id</i>	COPY	<i>collection-id.</i> <i>package-id</i>	¹ Keyword is not specified on subcommand invocation.
COPYVER	<i>version-id</i>	COPYVER	<i>version-id</i>	
PCURRENTDATA	NO YES <u>NONE</u>	CURRENTDATA	<u>YES</u> NO	
PDBPROTOCO	<u>NONE</u> DRDA <u>PRIVATE</u>	DBPROTOCOL	<u>DRDA</u> PRIVATE	If you specifit PRIVATE, your application cannot include SQL statements that were added to DB2 after DB2 Version 7.
PDBRMLIB	<u>DEFAULT</u> ¹ <i>dsname(member)</i> NONE ²	LIBRARY	<i>dbrm-pds-name</i>	¹ The precompiler DBRMLIB data set is used. If the precompiler DBRMLIB is not specified, the default-generated DBRMLIB library that is based on the INPUT data set is used. ² Keyword is not specified on subcommand invocation.
PDEFER	<u>NONE</u> ¹ PREPARE	DEFER	PREPARE	¹ Keyword is not specified on subcommand invocation.
PDEGREE	<u>1</u> ANY	DEGREE	<u>1</u> ANY	
PDISABLE	<u>NONE</u> BATCH CICS DB2CALL IMS DLIBATCH IMSBMP IMSMPP REMOTE RRSF	DISABLE	<u>NONE</u> BATCH CICS DB2CALL IMS DLIBATCH IMSBMP IMSMPP REMOTE RRSF	
PDLIBATCH	<u>NONE</u> ¹ <i>list of</i> <i>connection-ids</i>	DLIBATCH	<i>connection-name</i>	¹ Keyword is not specified on subcommand invocation.

Table 20. DSNH CLIST/ BIND PACKAGE subcommand summary (continued)

DSNH CLIST		BIND PACKAGE subcommand		
Parameter	Value	Keyword	Value	Comments
PDMEM	<u>DEFAULT</u> ¹ <i>dbrm-member- name</i> NONE ²	MEMBER	<i>dbrm-member- name</i>	¹ DBRM member name, which is obtained from one of the following sources, in the order listed: <ul style="list-style-type: none"> • PDBRMLIB member name • DBRMLIB member name • INPUT member name, or generated using <i>dsname</i> ² Keyword is not specified on subcommand invocation.
PDYNAMICRULES	<u>NONE</u> RUN BIND DEFINE INVOKE	DYNAMICRULES	<u>RUN</u> BIND DEFINE INVOKE	
PENABLE	<u>NONE</u> * BATCH CICS DB2CALL IMS DLIBATCH IMSBMP IMSMPP REMOTE RRSAF	ENABLE	<u>NONE</u> * BATCH CICS DB2CALL IMS DLIBATCH IMSBMP IMSMPP REMOTE RRSAF	
EXPLAIN	<u>NO</u> YES	EXPLAIN	<u>NO</u> YES	
PFLAG	<u>I C E W</u>	FLAG	<u>I C E W</u>	
PIMSBMP	<u>NONE</u> ¹ <i>imsid</i>	IMSBMP	<i>imsid</i>	¹ Keyword is not specified on subcommand invocation.
PIMSMPP	<u>NONE</u> ¹ <i>imsid</i>	IMSMPP	<i>imsid</i>	¹ Keyword is not specified on subcommand invocation.
PISOLATION	<u>NONE</u> ¹ RR RS CS UR NC	ISOLATION ¹	RR RS CS UR NC	¹ For local packages, the default value is the same as that of the plan that is appended at execution time. For remote packages, the default value is RR.
PKEEPDYNAMIC	<u>NONE</u> NO YES	KEEPDYNAMIC	<u>NO</u> YES	
PNODEFER	<u>NONE</u> ¹ PREPARE	NODEFER	PREPARE	¹ Keyword is not specified on subcommand invocation.
POPTHINT	<u>(' ')</u> (' <i>hint-id</i> ')	OPTHINT	<u>(' ')</u> (' <i>hint-id</i> ')	
POWNER	<u>NONE</u> ¹ <i>authorization-id</i>	OWNER	<i>authorization-id</i>	¹ Keyword is not specified on subcommand invocation.
PACKAGE	<u>DEFAULT</u> ¹ <i>location-name.</i> <i>collection-id</i>	PACKAGE	<i>location-name.</i> <i>collection-id</i>	¹ Member name that is defined in the INPUT parameter data set, or the data set name if no member name was specified.
PPATH	(<i>schema-name</i>) (USER)(<i>schema-name</i> , USER , ...)	PATH	(<i>schema-name</i>) (USER)(<i>schema-name</i> , USER , ...)	
PQUALIFIER	<u>NONE</u> ¹ <i>implicit-qualifier</i>	QUALIFIER	<i>qualifier-name</i>	¹ Keyword is not specified on subcommand invocation.

Table 20. DSNH CLIST/ BIND PACKAGE subcommand summary (continued)

DSNH CLIST		BIND PACKAGE subcommand		
Parameter	Value	Keyword	Value	Comments
PRELEASE	<u>NONE</u> ¹ COMMIT DEALLOCATE	RELEASE ¹	COMMIT DEALLOCATE	¹ For local packages, the default value is the same as that of the plan that is appended at execution time. For remote packages, the default value is NONE.
REOPT	<u>NONE</u> ¹ VARS	REOPT	<u>NONE</u> ALWAYS <u>ONCE</u> AUTO	¹ Keyword is not specified on subcommand invocation.
REMOTE	<u>NONE</u> ¹ location-name, <luname>	REMOTE	network-name	¹ Keyword is not specified on subcommand invocation.
REPLVER	<u>NONE</u> ¹ version-id	REPLVER	version-id	¹ version-id is not specified on subcommand invocation.
SQLERROR	<u>NOPACKAGE</u> CONTINUE	SQLERROR	<u>NOPACKAGE</u> CONTINUE	
PVALIDATE	<u>RUN</u> BIND	VALIDATE	<u>RUN</u> BIND	

Usage notes

CICS translator: Do not use CICS translator options in the source language for assembler programs; pass the options to the translator with the CICSOPT option.

COBOL options: The COBOL DYNAM option has several restrictions:

- You cannot use the option with CICS.
- You must use the VS COBOL II library or the Language Environment (z/OS Language Environment) library.
- To use the option with TSO or batch, the SDSNLOAD library must precede the IMS RESLIB in the step library, job library, or link list concatenations.
- To use the option with IMS, the IMS RESLIB must precede DSNLOAD.

Several COBOL options require DD statements that are not provided by the DSNH CLIST, as shown in the following table.

Table 21. COBOL options that require additional DD statements

Option	Statements required for...
CDECK	SYSPUNCH
COUNT	SYSCOUNT, SYSDBG, SYSDBOUT, SYSUT5, a debug file
DECK	SYSPUNCH
DUMP	SYSABEND, SYSDUMP, or SYSUDUMP
FDECK	SYSPUNCH
FLOW	SYSCOUNT, SYSDBG, SYSDBOUT, SYSUT5, a debug file
LVL	SYSUT6
STATE	SYSCOUNT, SYSDBG, SYSDBOUT, SYSUT5, a debug file
SYMDUMP	SYSCOUNT, SYSDBG, SYSDBOUT, SYSUT5, a debug file
SYST	SYSOUT

Table 21. COBOL options that require additional DD statements (continued)

Option	Statements required for...
SYSx	SYSOUx
TEST	SYSUT5

COBOL parameters: The BUF and SIZE parameters passed to the COBOL compiler might need to be changed.

COPTION: Do not use the COPTION parameter to specify values for the LINECOUNT, SOURCE, TERM, and XREF compiler options; use the DSNH LINECOUNT, SOURCE, TERM, and XREF keywords.

Fortran and PL/I considerations: Variable-format input records are not supported.

Library limits: At most, eight bind libraries, four precompile libraries, four compile libraries, and four link-edit libraries can exist.

User-supplied application defaults module (dsnhdecpr): The following steps are required to enable DSNH CLIST to load your user-supplied *dsnhdecpr* module rather than the DB2-supplied *dsnhdecpr* module:

1. The JOBLIB or STEPLIB concatenation of any job or TSO userid that calls DSNH must allocate the library where the user-supplied *dsnhdecpr* module resides (usually *prefix.SDSNEXIT*) before it allocates the library where the DB2-supplied *dsnhdecpr* module resides (*prefix.SDSNLOAD*).
2. The DSNH call should include the PCLOAD parameter, coded as follows:
PCLOAD('*(DSNHPC)')

Link-edit:

- DSNH cannot process programs that need additional link-edit control statements and cannot link-edit programs that use the call attachment facility.
- You cannot use the NOLOAD and SYNTAX link-edit options.

NONE is a reserved word: NONE cannot be the name of an input or a load library, or the value of the string passed with PARMS.

SQL host variables: You must explicitly define SQL host variables.

SYSPROC: If compilation is done, the SYSPROC data set must include the DB2 CLIST library.

WORKUNIT parameter: You must use the WORKUNIT parameter when running the DSNH CLIST in batch mode. This ensures that the temporary and intermediate data sets are allocated to the correct devices.

Examples

Example: Using DSNH to prepare and execute a COBOL application

The following command precompiles, binds, compiles, link-edits, and runs the COBOL program in data set *prefix.SDSNSAMP(DSN8BC4)*.

- The compiler load module is in SYS1.LINKLIB (IKFCBL00).
- Additional load modules to be included are in *prefix.RUNLIB.LOAD* and *prefix.SDSNSAMP*.

- The load module is to be put into the data set *prefix.RUNLIB.LOAD(DSN8BC4)*.
- The plan name is DSN8BC81 for the bind and run.
- DCLGEN data from *prefix.SRCLIB.DATA* is required for the precompile.
- The DSNH CLIST is in your SYSPROC concatenation.

```
DSNH INPUT('prefix.SDSNSAMP(DSN8BC4)') -
COBLOAD('SYS1.LINKLIB(IKFCBL00)') -
LLIB('prefix.RUNLIB.LOAD') -
L2LIB('prefix.SDSNSAMP') -
LOAD('prefix.RUNLIB.LOAD') -
PLAN(DSN8BC81) -
PLIB('prefix.SRCLIB.DATA')
```

Example: Using DSNH to prepare and execute a PL/I application

The following command precompiles, binds, compiles, link-edits, and runs the COBOL program in data set *prefix.SDSNSAMP.PLI(DSN8BP4)*.

- The program is written in PL/I; the macro pass is not needed.
- The PL/I compiler options MAP and LIST are to be used.
- Additional load modules to be included are in *prefix.RUNLIB.LOAD* and *prefix.SDSNSAMP*.
- The PL/I optimizing compiler load module is in library SYS2.LINKLIB(IEL0AA).
- The DB2 subsystem identifier is SSTR.
- The load module is put into the data set *prefix.RUNLIB.LOAD(DSN8BC4)*.
- Printed output is sent to the following data sets:

Output	Data set
Precompiler listings	<i>prefix.PROG.PCLIST</i>
Compiler listings	<i>prefix.PROG.LIST</i>
Link-edit listings	<i>prefix.PROG.LIST</i>

- The plan name is DSN8BC81 for the bind and run.
- The DCLGEN data from *prefix.SRCLIB.DATA* is required for the precompile.

```
DSNH INPUT('prefix.SDSNSAMP(DSN8BP4)') -
HOST(PLI) MACRO(NO) -
COPTION ('MAP LIST') -
LLIB('prefix.RUNLIB.LOAD') -
L2LIB('prefix.SDSNSAMP') -
PLILOAD('SYS2.LINKLIB(IEL0AA)') -
SYSTEM(SSTR) -
LOAD('prefix.RUNLIB.LOAD') -
PRINT(PROG) -
PLAN(DSN8BC81) -
PLIB('prefix.SRCLIB.DATA')
```

The COPTION parameters are enclosed between single apostrophes so that they are passed by TSO as a single parameter. If a single token is being passed as a parameter, no apostrophes are needed. That same rule applies to the PARMs and CICSOPT parameters.

If a data set name is being passed as a parameter, and you want TSO to add your user prefix, no apostrophes are needed. If the usual TSO prefixing and suffixing must not be performed, the data set name must be

enclosed between sets of three apostrophes if the CLIST is executed implicitly, and sets of six apostrophes if the CLIST is executed explicitly.

The user prefix for that example is *prefix*; if it had been SMITH, the listing data set names would be as shown in the preceding example, except that SMITH would be used as the first level qualifier. For example, the compiler listings would have gone to SMITH.PROG.LIST.

Example: Using DSNH to prepare and execute a COBOL application

The following command precompiles, binds, compiles, link-edits, and runs the COBOL program in data set *prefix.SDSNSAMP(DSN8BC4)*.

- The compiler load module is in SYS1.LINKLIB (IKFCBL00).
- Additional load modules to be included are in *prefix.RUNLIB.LOAD* and *prefix.SDSNSAMP*.
- The load module is to be put into the data set *prefix.RUNLIB.LOAD(DSN8BC4)*.
- The plan name is DSN8BC81 for the bind and run.
- DCLGEN data from *prefix.SRCLIB.DATA* is required for the precompile.
- The DSNH CLIST is in your SYSPROC concatenation.

```
DSNH INPUT('prefix.SDSNSAMP(DSN8BC4)') -
COBLOAD('SYS1.LINKLIB(IKFCBL00)') -
LLIB('prefix.RUNLIB.LOAD') -
L2LIB('prefix.SDSNSAMP') -
LOAD('prefix.RUNLIB.LOAD') -
PLAN(DSN8BC81) -
PLIB('prefix.SRCLIB.DATA')
```

Example: Using DSNH to prepare and execute a C application

The following command precompiles, binds, compiles, link-edits, and runs the C program in data set *prefix.SDSNSAMP(DSN8BD3)*.

- The C linkage editor include library is EDC.V1R1M1.SEDCBASE
- The C compiler load module is EDC.V1R1M1.SEDCCOMP(EDCCOMP)
- Printed output is sent to the following data sets:

Output	Data set
Precompiler listings	<i>user-id</i> .TEMP.PCLIST
Compiler listings	<i>user-id</i> .TEMP.LIST
Prelink-edit listings	<i>user-id</i> .TEMP.PRELLIST
Link-edit listings	<i>user-id</i> .TEMP.LINKLIST

- The following C DD names are allocated based on the PRINT keyword value:

DD name	Allocation
SYS CPRT ¹	Used in the compiler step
SYS UT10 ¹	Used in the compiler step
SYS OUT	Used in the prelink-edit step

Note:

1. SYSUT10 and SYSCPRT are always allocated to the same data set or destination.

- SYSTEM is used in the compiler step. It is based on the TERM keyword.

- CEEDUMP is used in the run step. It is based on the RUNOUT keyword.
- The LOPTION keyword values of AMODE(31) and RMODE(ANY) are required when link editing the C sample program to ensure 31-bit addressability during execution.

```

ALLOC      DD(SYSPROC) DSN('prefix.SDSNCLST ') SHR
%DSNH BIND(YES) ACQUIRE(USE) ACTION(REPLACE)-
EXPLAIN(NO) -
CICSXLAT(NO) -
COMPILE(YES) -
CCLLIB('EDC.V1R1M1.SEDCBASE')-
CCLOAD('EDC.V1R1M1.SEDCCOMP(EDCCOMP)')-
DBRM('prefix.DBRMLIB.DATA(DSN8BD3)')-
DECIMAL(PERIOD) DELIMIT(DEFAULT) FLAG(I)-
HOST(C) ISOLATION(RR)-
INPUT('prefix.SDSNSAMP(DSN8BD3)')-
LINK(YES)-
LLIB('prefix.RUNLIB.LOAD')-
L2LIB('prefix.SDSNLOAD')-
LOAD('prefix.RUNLIB.LOAD')-
LOPTION('AMODE(31) RMODE(ANY)')-
MACRO(NO)-
OUTNAME(TEMP)-
PLAN(DSN8BD31) PRECOMP(YES)-
PLIB('prefix.SDSNSAMP')-
PRELINK(NO)-
POPTION(NONE)-
PRINT(TEMP) RCTERM(8)-
RELEASE(COMMIT) RETAIN(YES)-
RUN(NO) RUNIN(TERM)-
RUNOUT(TERM) SOURCE(YES)-
SYSTEM(DSN) SQLDELIM(DEFAULT)-
VALIDATE(RUN)

```

Related reference:

BIND PLAN (DSN)

RUN (DSN)

BIND PACKAGE (DSN)

The DSN command and its subcommands

Chapter 42. END (DSN)

The DSN subcommand END is used to end the DSN session and return to TSO.

Environment

This subcommand originates from a TSO input stream when DSN is running in either background or foreground mode.

Data sharing scope: Member

Authorization

None is required.

Syntax

```
▶▶—END—————▶▶
```

Usage note

Ending the DSN session in batch or foreground: In batch, if END is not found in the SYSIN stream, /* or // ends the DSN session. From the foreground, pressing the ATTENTION key twice ends the DSN session.

Examples

Example: Ending a DSN session

The following example shows how TSO responds when a user ends a DSN session.

```
TSO prompt : READY
USER enters: DSN SYS (SSTR)
DSN prompt : DSN
USER enters: RUN PROGRAM (MYPROG)
DSN prompt : DSN
USER enters: END
TSO prompt : READY
```

Chapter 43. FREE PACKAGE (DSN)

The DSN subcommand FREE PACKAGE can be used to delete a specific version of a package, all versions of a package, or whole collections of packages.

The FREE PACKAGE subcommand deletes corresponding table entries from the catalog tables. Authorization for a package name is removed only when no more versions of the package exist. After a version of a package has been freed, that package name is then available for use in a BIND PACKAGE subcommand to create a new package.

The FREE PACKAGE subcommand does not proceed until all currently running applications using the package finish running.

Environment

You can enter this subcommand from DB2I, or from a DSN session under TSO that is running in either foreground or background.

Data sharing scope: Group

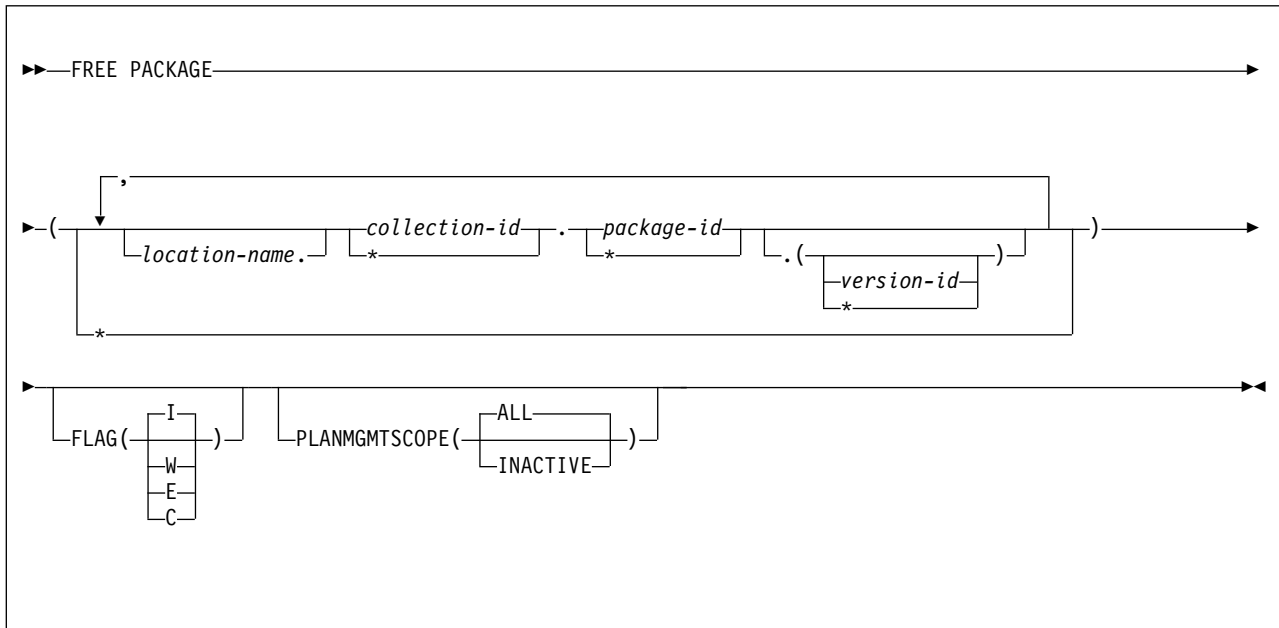
Authorization

To execute this subcommand, you must use a privilege set of the process that includes one of the following privileges or authorities:

- Ownership of the package
- BINDAGENT privilege granted by the owner of the package
- System DBADM authority
- SYSCTRL authority
- SYSADM authority
- PACKADM authority for the collection or for all collections

The BIND privilege on a package is not sufficient to allow a user to free a package.

Syntax



Option descriptions

location-name

Specifies the location of the DBMS where the package is to be freed. The location name must be defined in the SYSIBM.LOCATIONS table. If this table does not exist or the DBMS is not found, you receive an error message. If the location name is specified, the name of the local DB2 subsystem must be defined.

The **default** is the local DB2 subsystem if you omit *location-name*.

collection-id or (*)

Identifies the collection of the package to be freed. There is no default.

You can use an asterisk (*) to free all local packages with the specified *package-id* in all the collections that you are authorized to free. (You cannot use the * to free remote packages.)

collection-id can be an undelimited or a delimited identifier. The delimiter for *collection-id* is double quotation marks ("). If *collection-id* is delimited, DB2 does not convert the value to uppercase.

package-id or (*)

Identifies the package to be freed. There is no default.

You can use an asterisk (*) to free all local packages in the *collection-id* that you are authorized to free. (You cannot use the * to free remote packages.)

package-id can be an undelimited or a delimited identifier. The delimiter for *package-id* is double quotation marks ("). If *package-id* is delimited, DB2 does not convert the value to uppercase.

version-id or (*)

Identifies the version of the package to be freed.

You can use an asterisk (*) to free all local packages in the *collection-id* and *package-id* that you are authorized to free. (You cannot use the * to free remote packages.)

If you specify () for *version-id* , the empty string is used for the version ID.

If you omit the *version-id* , the default depends on how you specify *package-id* .
If you use * for *package-id* , *version-id* defaults to *. If you provide an explicit value for *package-id* , *version-id* defaults to an empty string.

(*)

Frees all local DB2 packages that you are authorized to free.

Specifying (*) is equivalent to specifying the package name as (*.*(*)) or (*.*) .

FLAG

Indicates what messages you want the system to display. Use one of the following values to show messages of the corresponding types.

(I) All: informational, warning, error, and completion messages.

(W) Only warning, error, and completion messages.

(E) Only error and completion messages.

(C) Only completion messages.

PLANMGMTSCOPE

Allows you to manually free inactive copies of packages. Use one of the following options to help reclaim disk space:

ALL Frees the entire package, including copies. This is the default.

INACTIVE

Frees only previous or original copies from the directory, catalog, and access path repository. FREE PACKAGE with PLANMGMTSCOPE(INACTIVE) succeeds even if the package has no inactive copies.

Usage notes

Freeing multiple packages

If you free multiple packages with this subcommand, each successful free is committed before freeing the next package.

If an error occurs on a certain package specified explicitly in a list or implicitly with (*), FREE PACKAGE terminates for that package and continues with the next package to be processed.

Freeing trigger packages

You cannot use the FREE PACKAGE command to free a trigger packages.

:

Examples


Free version *newver* of the package TEST.DSN8BC81 located at USIBMSTODB22. Generate only warning, error, and completion messages (not informational messages):

```
FREE PACKAGE (USIBMSTODB22.TEST.DSN8BC81.(newver)) FLAG(W)
```

Free all packages at the local server in the collection named TESTCOLLECTION.

```
FREE PACKAGE (TESTCOLLECTION.*)
```

Related concepts:

 Package copies (DB2 Performance)

Chapter 44. FREE PLAN (DSN)

The DSN subcommand FREE PLAN deletes application plans from DB2.

The FREE PLAN subcommand deletes corresponding table entries from the SYSIBM.SYSPLAN catalog tables. All authorization against an application plan name is dropped. The application plan name is then available for use in a BIND PLAN subcommand to create a new package.

The FREE PLAN subcommand does not proceed until all currently executing applications using that plan finish executing.

Environment

You can enter this subcommand from DB2I, or from a DSN session under TSO that is running in either foreground or background.

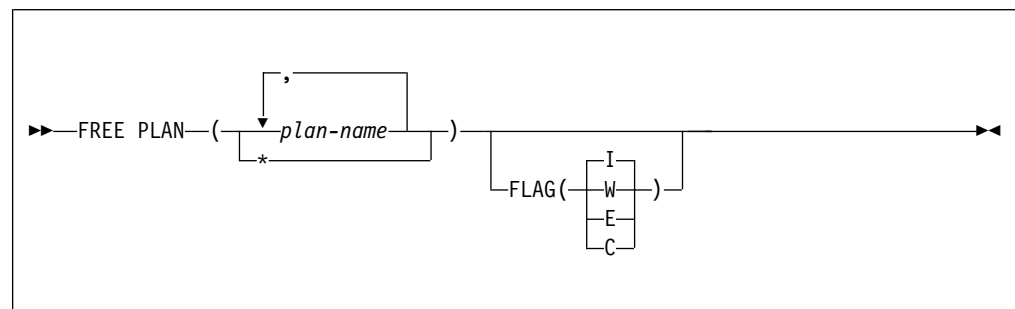
Data sharing scope: Group

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- Ownership of the plan
- BIND privilege on the plan
- BINDAGENT privilege granted by the plan owner
- System DBADM authority
- SYSCTRL authority
- SYSADM authority

Syntax



Option descriptions

(*plan-name* , ...)

Lists the names of one or more plans you want to free.

(*)

Frees *all* application plans over which you have BIND authority. Be careful when using this form of the command.

FLAG

Indicates what messages you want the system to display. Use one of the values listed to show messages of the corresponding types.

- (I) All: informational, warning, error, and completion messages.
- (**W**) Only warning, error, and completion messages.
- (**E**) Only error and completion messages.
- (**C**) Only completion messages.

Usage notes

Freeing multiple plans: If you free multiple plans with this subcommand, each successful free is committed before freeing the next plan.

If an error occurs on a certain plan specified explicitly in a list or implicitly with (*), FREE PLAN terminates for that plan and continues with the next plan to be processed.

Example

Free plan DSN8BC81 from DB2. Generate only warning, error, and completion messages (not informational messages).

```
FREE PLAN (DSN8BC81) FLAG (W)
```

Chapter 45. FREE QUERY (DSN)

The DSN subcommand FREE QUERY removes from certain catalog tables for one or more queries. If any of the specified queries are in the dynamic statement cache, FREE QUERY purges them from the dynamic statement cache.

Environment

You can use FREE QUERY from DB2I, or from a DSN session under TSO that runs in either the foreground or background. You can also use the SYSPROC.ADMIN_COMMAND_DSN stored procedure to submit this subcommand from a remote requester.

Data sharing scope: Group

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SQLADM authority
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

Syntax

```
►► FREE QUERY filter-block package-block ◄◄
```

filter-block:

```
►► FILTER('filter-name') ◄◄  
  PACKAGE(package-name) ◄◄  
  QUERYID(number) ◄◄  
  QUERYID(ALL) ◄◄
```

package-block:

```
►► location-name . collection-id . package-id . (version-id) ◄◄  
  * * * *
```

Option descriptions

FILTER('filter-name')

Specifies the queries that are to be removed from the access path repository.

filter-name is a value in the USERFILTER column of the SYSIBM.SYSQUERY catalog table. During FREE QUERY processing, all the rows in the SYSIBM.SYSQUERY table that have the USERFILTER value *filter-name* are removed. Deletions from the SYSIBM.SYSQUERY table are cascaded to the SYSIBM.SYSQUERYPLAN table or the SYSIBM.SYSQUERYOPTS table.

PACKAGE(*package-name*)

The name of the package from which the queries are to be freed.

QUERYID(*number*)

Frees an entry in the SYSIBM.SYSQUERY table that has the same QUERYID value, and the corresponding entries in the SYSIBM.SYSQUERYPLAN table or the SYSIBM.SYSQUERYOPTS table.

QUERYID(ALL)

Frees all the entries from the SYSIBM.SYSQUERY table and the corresponding entries from the SYSIBM.SYSQUERYPLAN table or the SYSIBM.SYSQUERYOPTS table.

location-name

Specifies the location of the data server where the query is to be freed. Only the location name of the local DB2 subsystem can be specified. If the location name is specified, the name of the local DB2 subsystem must be defined in the SYSIBM.LOCATIONS table. If this table does not exist or the data server is not found, an error message is issued.

The default value is the local DB2 subsystem.

collection-id or (*)

Identifies the collection that is associated with the query to be freed. There is no default.

You can use an asterisk (*) to free all packages with the specified *package-id* in all the collections that you are authorized to free.

package-id or (*)

Identifies the package that is associated with the query to be freed. There is no default.

You can use an asterisk (*) to free all packages in the *collection-id* that you are authorized to free.

version-id or (*)

Identifies the version of the package for which the associated query is to be freed.

You can use an asterisk (*) to free all local packages in the *collection-id* and *package-id* that you are authorized to free. You cannot use the * to free remote packages.

If you specify () for *version-id* , the empty string is used for the version ID.

If you omit *version-id* , the default depends on how you specify *package-id* . If you use * for *package-id* , *version-id* defaults to *. If you provide an explicit value for *package-id* , *version-id* defaults to an empty string.

(*)

Frees all local DB2 packages that you are authorized to free.

Specifying (*) is equivalent to specifying the package name as (*.*(*)) or (*.*) .

Usage notes

Freeing multiple queries: If you use FREE QUERY to free multiple queries, each successful free operation is committed before the next query is freed. If an error occurs on a query, FREE QUERY terminates for that package and continues processing the next query.

Examples

Example 1: Free all access paths for all queries:

```
FREE QUERY QUERYID(ALL)
```


Example 2: Free queries for which the USERFILTER column of the SYSIBM.SYSQUERY catalog table contains SALESAPP:


```
FREE QUERY  
    FILTER('SALESAPP')
```

Example 3: Free all queries in package SALESPACK:

```
FREE QUERY  
    PACKAGE(SALESCOLL.SALESPACK)
```

Related tasks:

 [Influencing access path selection \(DB2 Performance\)](#)

 [Freeing statement-level access paths \(DB2 Performance\)](#)

Related reference:

[BIND QUERY \(DSN\)](#)

 [SYSIBM.SYSQUERY table \(DB2 SQL\)](#)

 [SYSIBM.SYSQUERYOPTS table \(DB2 SQL\)](#)

 [SYSIBM.SYSQUERYPLAN table \(DB2 SQL\)](#)

Chapter 46. MODIFY *admtproc*,APPL=SHUTDOWN

The MODIFY *admtproc*, APPL=SHUTDOWN command stops the administrative task scheduler from accepting requests and starting new task executions. It also shuts down the administrative task scheduler.

When the SHUTDOWN option is specified, the administrative task scheduler waits until the execution of all currently running tasks completes. When all running tasks are complete, the administrative task scheduler terminates.

Environment

This command can be issued only from a z/OS console.

Data sharing scope: Member

Authorization

The command requires an appropriate level of operating system authority.

Syntax

```
▶▶—MODIFY—admtproc,APPL=—SHUTDOWN—◀◀
```

You cannot include spaces when you specify options.

Option descriptions

admtproc

Specifies the procedure name of the scheduled task of the administrative task scheduler that you want to modify.

Examples

Example 1: This command modifies the *admtproc* scheduler in order to shut it down.

Enter the following command on the system console:

```
modify admtproc,appl=SHUTDOWN
```

Chapter 47. MODIFY *admtproc*,APPL=TRACE

The MODIFY *admtproc*, APPL=TRACE command starts or stops traces in the administrative task scheduler.

It is not required to stop the scheduler to access the trace. If you want to turn trace on or off when the administrative task scheduler starts, you can take one of the following actions:

- Modify the procedure parameter TRACE in the JCL job that starts the administrative task scheduler. This job has the name *admtproc* and was copied into one of the PROCLIB library during the installation. Specify TRACE=ON or TRACE=OFF.
- Dynamically overwrite the trace parameter on the operator's console when starting the administrative task scheduler. This option does not exist when DB2 starts the administrative task scheduler automatically, and can only be done manually.

Environment

This command can be issued only from a z/OS console.

Data sharing scope: Member

Authorization

The command requires an appropriate level of operating system authority.

Syntax

```
►►—MODIFY—admtproc,APPL=—TRACE=—ON  
                                  └─OFF─┘                                  ◄◄
```

You cannot include spaces when you specify options.

Option descriptions

admtproc

Specifies the procedure name of the administrative task scheduler task that you want to modify.

ON

Turns the trace on.

OFF

Turns the trace off.

Examples

Example 1: This command modifies the *admtproc* scheduler and turns the trace on.

Enter the following command on the system console:

```
modify admtproc,appl=trace=on
```

Response from z/OS console:

```
STC00072 DSN672I MODIFY COMMAND FOR ADMIN SCHEDULER V91AADMT  
NORMAL COMPLETION
```

Example 2: This command modifies the *admtproc* scheduler and turns the trace off.

Enter the following command on the system console:

```
modify admtproc,appl=trace=off
```

Response from z/OS console:

```
STC00072 DSN672I MODIFY COMMAND FOR ADMIN SCHEDULER V91AADMT  
NORMAL COMPLETION
```

Chapter 48. -MODIFY DDF (DB2)

The MODIFY DDF command modifies information regarding the status and configuration of DDF, as well as statistical information regarding connections or threads controlled by DDF.

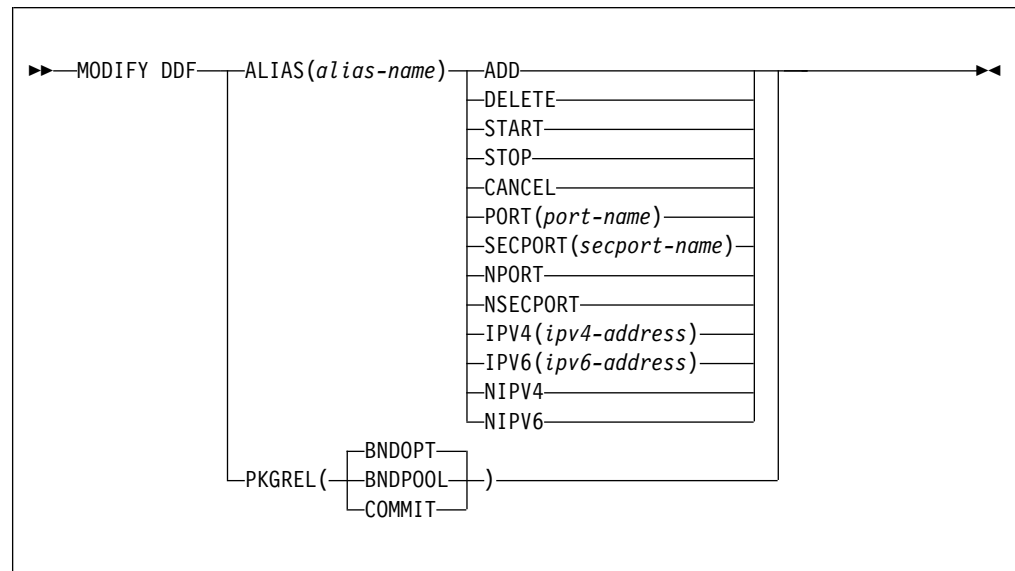
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

ALIAS

Specifies the creation, changes to, or deletion of a specified location alias.

alias-name

Specifies the name of an alias for a DDF location. An alias is an alternative for the location name that can be used for connection processing. The alias name must meet all of the following requirements:

- Contains no more than 16 characters.
- Contains only letters (excluding alphabetic extenders), numbers, or the underscore character.

ADD

Creates a new alias with the specified name.

DELETE

Deletes the specified alias.

START

DB2 starts accepting connection requests to the specified alias if DDF is started. If DDF is not started, the alias is marked eligible for starting, and DB2 automatically starts accepting connection requests to the alias when DDF is started.

If the subsystem is part of a data sharing group, DB2 registers the alias with WLM and DB2 participates in sysplex workload balancing for connections to the alias.

STOP

DB2 stops accepting new connection requests to the specified alias. Existing database access threads that process connections to the alias remain unaffected. Inactive connections related to the alias are closed.

A stopped alias is marked ineligible for starting and does not start automatically when DDF starts. If the subsystem is part of a data sharing group, DB2 de-registers the alias with WLM and DB2 stops participating in sysplex workload balancing for connections to the alias.

CANCEL

DB2 stops accepting new connection requests to the specified alias. All database access threads that process connections to the alias are canceled and inactive connections related to the alias are closed.

A canceled alias is marked ineligible for starting and does not start automatically when DDF starts. If the subsystem is part of a data sharing group, DB2 de-registers the alias with WLM and DB2 stops participating in sysplex workload balancing for connections to the alias.

PORT (*port-name*)

Adds or replaces an existing port that can be used by DDF to accept distributed requests for the specified alias. The value specified for *port-name* value must be a decimal number between 1 and 65535, including 65535, and must be different than the values for the other specified ports. Specify a PORT value for an alias when you want to identify a subset of data sharing members to which a distributed request can go.

SECPORT (*secpport-name*)

Adds or replaces an existing secure port that can be used by DDF to accept secure distributed requests using SSL for the specified alias. The value specified for *secpport-name* must be a decimal number between 1 and 65535, including 65535, and must be different than the values for the other specified ports. Specify a SECPORT value for an alias when you want to identify a subset of data sharing members to which a secure distributed request can go.

NPORT

Deletes the alias port, if one exists.

NSECPORT

Deletes the alias secure port, if one exists.

IPV4(IPv4-address)

Adds or replaces an existing member-specific IPv4 address. Clients use that address when they use one of the following capabilities for a connection to a dynamic location alias:

- Sysplex workload balancing
- XA protocols for coordinating distributed transactions

This address must be specified in the dotted decimal form.

IPV6(IPv6-address)

Adds or replaces an existing member-specific IPv6 address. Clients use that address when they use one of the following capabilities for a connection to a dynamic location alias:

- Sysplex workload balancing
- XA protocols for coordinating distributed transactions

This address must be specified in the colon hexadecimal form.

NIPV4

Deletes the alias IPv4 address, if one exists.

NIPV6

Deletes the alias IPv6 address, if one exists.

PKGREL

Specifies whether DB2 honors the bind options of packages that are used for remote client processing.

BNDOPT

The rules of the RELEASE bind option that was specified when the package was bound are applied to any package that is used for remote client processing. The high performance DBAT that is used by a terminating client connection is deallocated. BNDOPT is the default value of the MODIFY DDF PKGREL command.

BNDPOOL

The rules of the RELEASE bind option that was specified when the package was bound are applied to any package that is used for remote client processing. The high performance DBAT that is used by terminating a client connection is pooled.

COMMIT

The rules of the RELEASE(COMMIT) bind option are applied to any package that is used for remote client processing. COMMIT is the default value when the CMTSTAT subsystem parameter is set to ACTIVE. If the MODIFY DDF PKGREL command had never been issued, then COMMIT is the default value and the CMTSTAT subsystem parameter is set to INACTIVE.

Usage notes

When to use PKGREL options: You can specify that DDF uses the PKGREL(BNDOPT) or PKGREL(BNDPOOL) option during normal production operating hours. For environments where client configurations control the use of packages which were bound with the RELEASE(DEALLOCATE) bind option, BNDOPT may be the preferred value of the PKGREL option. For environments where any client is capable of using packages bound with the RELEASE(DEALLOCATE) bind option, BNDPOOL might be the preferred value of the PKGREL option. Either option value offers improved performance by reducing

the CPU costs for allocation and deallocation of packages. However, packages that run under the rules of the `RELEASE(DEALLOCATE)` bind option are likely to remain allocated and prevent maintenance activities such as objects modifications and bind operations. Consequently, you can use the specify the `PKGREL(COMMIT)` option during routine and emergency maintenance periods.


Delayed effects of `PKGREL(COMMIT)`: When you issue the `MODIFY DDF` command and specify the `PKGREL(COMMIT)` option, the effects are not immediate. After the command is issued, any database access thread that was running `RELEASE(DEALLOCATE)` packages is terminated when the connection becomes inactive. At the next unit-of-work from the client, a new database access thread is created in `RELEASE(COMMIT)` mode. Any database access thread that remains active waiting for a new unit-of-work request from its client because of the rules of `RELEASE(DEALLOCATE)` is terminated by the DDF service task that runs every two minutes. Consequently, within approximately two minutes all database access threads run under the rules of the `RELEASE(COMMIT)` bind option.

Changes to alias attributes: The attributes of an existing alias can be modified only when the alias is stopped. The modified alias attributes take effect when the alias is started. By default, aliases created by the `DSNJU003` utility are started and those created by the `MODIFY DDF` command are stopped. `DSNJU004` does not print any information for aliases that are created by the `MODIFY DDF` command. You can use the output of the `DISPLAY DDF` command to find the status of a aliases created by the `MODIFY DDF` command.


How DB2 uses the `IPV4` and `IPV6` values: DB2 does not activate the IP addresses that you specify with the `IPV4` or `IPV6` parameters. Clients use those addresses for routing purposes only. DB2 does not require that the addresses are dynamic virtual IP addresses (DVIPAs). The IP addresses are used to reach the DVIPA network that serves the DB2 data sharing group. You can specify the IP addresses in the `MODIFY DDF` command only after the `DSNJU003` is run with the `IPV4` or `IPV6` parameter to specify a member-specific location address. The specified IP addresses are returned in the WLM weighted server list when clients connect to a dynamic location alias, based on the following conditions:

- If the client connects using an IPv6 address, the server list contains the alias IP addresses that were specified with the `IPV4` and `IPV6` parameters of the `MODIFY DDF` command. If an IPv4 address was not specified in the `MODIFY DDF` command, the location IP address that was specified using the `IPV4` keyword of the `DSNJU003` utility is returned instead. Similarly, if an IPv6 address is not specified in the `MODIFY DDF` command, the location IP address that was specified using the `IPV6` keyword of the `DSNJU003` utility is returned instead.
- If the client connects using an IPv4 address, the server list contains the alias IP address that was specified with the `IPV4` parameter of the `MODIFY DDF` command. If an IPv4 address was not specified in the `MODIFY DDF` command, the location IP address that was specified using the `IPV4` keyword of the `DSNJU003` utility is returned instead.

Related concepts:

 [Member-specific location aliases \(DB2 Data Sharing Planning and Administration\)](#)

Related tasks:

 [Defining dynamic location aliases \(DB2 Data Sharing Planning and Administration\)](#)

|
|

➔ Managing dynamic location aliases (DB2 Data Sharing Planning and Administration)

Chapter 49. MODIFY irlmproc,ABEND (z/OS IRLM)

The MODIFY *irlmproc* , ABEND command terminates IRLM abnormally. IRLM processes this command even if a DB2 subsystem is identified to it.

Abbreviation: F

Environment

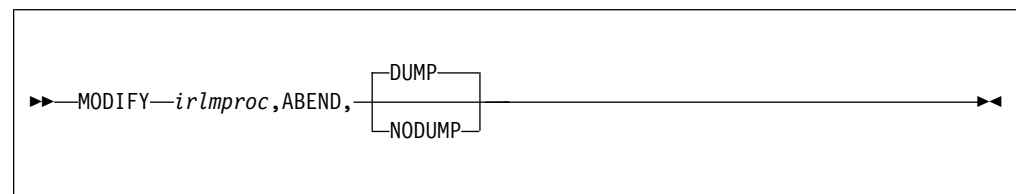
This command can be issued only from a z/OS console.

Data sharing scope: Member

Authorization

The command requires an appropriate level of operating system authority.

Syntax



Option descriptions

Parameters must be separated by commas with no spaces.

irlmproc

Specifies the procedure name of the IRLM that is to be terminated.

DUMP

Specifies that IRLM is to terminate abnormally with a U2020 abend. A system dump is taken to the SYS1.DUMPxx data set. IRLM does not deregister from ARM.

NODUMP

Specifies that IRLM is to FORCE the DBMS off and terminate normally without generating a dump. All DBMS work is quiesced and IRLM stops itself. NODUMP **requires** that IRLM be functioning normally. **Do not** use this option if IRLM appears to be hung.

A second invocation causes IRLM to terminate abnormally with a U2020 abend; no dump is taken.

Usage notes

Terminating IRLM: Use the STOP irlmproc (z/OS IRLM) command to terminate IRLM.

Deregistering IRLM: You can use the NODUMP option to deregister IRLM before stopping it. This action prevents the automatic restart manager from immediately trying to restart IRLM.

Example

Enter the following command on the system console:

```
F KRLM001,ABEND
```

Response on the z/OS system console is as follows:

```
DXR124E IR21001 ABENDED VIA MODIFY COMMAND
*IEA911E COMPLETE DUMP ON SYS1.DUMP00
  FOR ASID(0004)
  ERROR ID = SEQ00001 CPU00 ASID0004 TIME08.34.59.9
DXR121I IR21001 END-OF-TASK CLEANUP SUCCESSFUL
IEF450I IR21001 IR21001 - ABEND=S000 U2020 REASON=00000000
```

The default is dump. If you do not want a dump, you must specify the following command:

```
F KRLM001,ABEND,NODUMP
```

Chapter 50. MODIFY *irlmproc*,DIAG (z/OS IRLM)

The MODIFY *irlmproc* , DIAG command initiates diagnostic dumps for IRLM subsystems.)

If IRLM detects a delay in the child-lock propagation process, it retries the XES calls in order to recover. Use the MODIFY *irlmproc* , DIAG command under the direction of IBM Software Support if this situation occurs.

Abbreviation: F

Environment

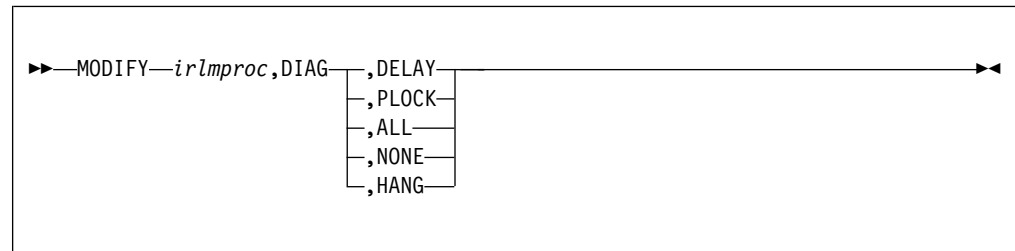
This command can be issued only from a z/OS console.

Data sharing scope: Group

Authorization

The command requires an appropriate level of operating system authority.

Syntax



Option descriptions

Parameters must be separated by commas, with no spaces.

irlmproc

Specifies the procedure name of the IRLM instance that is to be diagnosed.

DIAG

Specifies that this is a diagnostic dump.

DELAY

Directs IRLM to generate a dump the first time it detects that child lock propagation to the coupling facility is taking longer than 45 seconds. The dump is placed in the SYS1.DUMPxx data set.

PLOCK

Directs IRLM to generate a dump the first time it detects that P-lock negotiation is taking longer than four minutes. Dumps of the IRLM and DB2 address spaces are placed in the SYS1.DUMPxx data set.

ALL

Directs IRLM to generate diagnostic dumps for IRLM or DBMS subsystems in a data sharing group for the following unusual conditions:

- P-lock negotiation takes longer than four minutes.
- Child-lock propagation takes longer than 72 seconds. IRLM retries the child lock propagation after 78 seconds if this condition still exists.
- If IRLM detects a delay in the child-lock propagation process, it retries the XES calls in order to recover.

NONE

Disables generating all diagnostic dumps.

HANG

Collects IRLM SYSPLEX dumps when DEADLOCK or TIMEOUT issues are suspected. The dumps are taken during DEADLOCK processing. The DEADLOCK processing is stopped, and the dynamic deadlock storage is collected. z/OS DUMP services then schedules an SRB to restart DEADLOCK processing. Message DXR183I is issued by each IRLM as DEADLOCK processing is restarted. If message DXR183I is not issued by an IRLM, that IRLM must be terminated and restarted. You must start the IRLM XCF CTRACE internally and wait 30 seconds before issuing this command.

Usage note

The `MODIFY irlmproc,DIAG` command should be used only under the direction of IBM Software Support.

This command is active for only one incident per IRLM, that is, after an IRLM instance detects the delay and initiates the dump. You can initiate one dump per IRLM in the group. You must enter the command again to initiate another dump. Be aware that when you enter this command for one member of the data sharing group, **any** member that detects the delay initiates a dump.

The *irlmproc* identifies the procedure name for IRLM. If multiple IRLM instances exist in the same system, each procedure must have a unique name.

Example

Issue this command to initiate one diagnostic dump for the IR21PROC IRLM subsystem. The dump occurs once, after the propagation of child locks takes longer than 45 seconds.

```
MODIFY IR21PROC,DIAG,DELAY
```

Chapter 51. MODIFY irlmproc,PURGE (z/OS IRLM)

The MODIFY irlmproc,PURGE command releases IRLM locks retained due to a DB2, IRLM, or system failure.

The command causes all retained locks for the specified DB2 to be deleted from the system, thereby making them available for update. Because retained locks protect updated resources, it should be used only after understanding what the resources are and the consequence to data integrity if they are deleted.

Abbreviation: F

Environment

This command can be issued only from a z/OS console.

Data sharing scope: Member

Authorization

The command requires an appropriate level of operating system authority.

Syntax

```
►►—MODIFY—irlmproc,PURGE,db2name—◄◄
```

Option descriptions

Use commas with no spaces to separate parameters.

irlmproc

Specifies the active IRLM that is to process the command.

db2name

Specifies the inactive DB2 name, as displayed by the STATUS command.

Usage notes

DB2 subsystem inactive: The DB2 subsystem that owns the retained locks must be inactive or else this command fails.

The irlmproc must be the procedure name of an active IRLM that is connected to the same sysplex group as the failed member. Issuing a purge request using an inactive IRLM returns error IEE341I.

Example

Example: For an active DB2 subsystem named db2b with irlmproc name db2birlm, issue the following command to display all active and inactive subsystems in a data sharing sysplex:

```
F db2bir1m,STATUS,ALLD
```

If the subsystem db2a is inactive, enter the following command:

```
F db2bir1m,PURGE,db2a
```

Response on the MVS system console for completed purge request:

```
DXR109I IR2B002 PURGE COMMAND COMPLETED FOR DB2A
```

Explanation: In a sysplex environment, if the DB2 database is inactive and the database IRLM has stopped or is disconnected, the operator of the z/OS system uses one of the other active IRLM members to query retained locks and issue the PURGE request.

Chapter 52. MODIFY irlmproc,SET (z/OS IRLM)

The MODIFY irlmproc,SET command dynamically sets various IRLM operational parameters

The MODIFY irlmproc,SET command performs the following tasks:

- Dynamically sets the maximum private storage allowed from IRLM.
- Dynamically sets the number of trace buffers allowed for IRLM.
- Dynamically sets the number of LOCK LTE entries to be specified on the next connect to the XCF LOCK structure.
- Dynamically sets the timeout value for a specified subsystem.
- Dynamically sets the local deadlock frequency.

Abbreviation: F

Environment

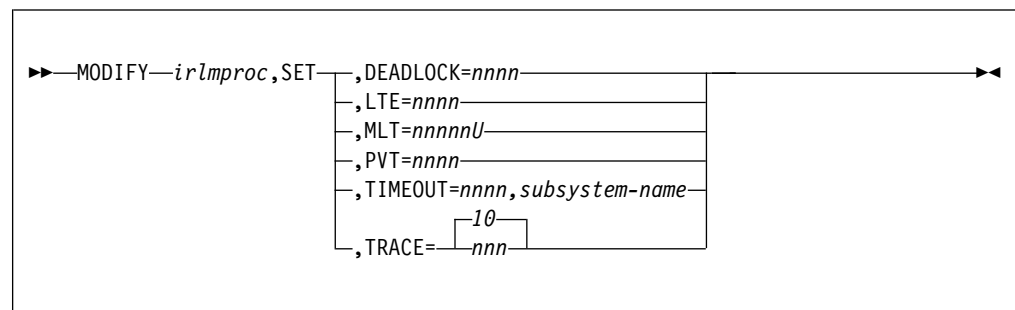
This command can be issued only from a z/OS console.

Data sharing scope: Group or Member, depending on whether you specify the DEADLOCK or LTE options.

Authorization

The command requires an appropriate level of operating system authority.

Syntax



Option description

Use commas with no spaces to separate parameters.

irlmproc

Specifies the IRLM that is to process the command.

SET

Sets the following values for this IRLM:

DEADLOCK= nnnn

Specifies the number, in milliseconds, indicating how often the local deadlock processing is scheduled. *nnnn* must be a number from 100

through 5000 milliseconds. If a member of a sysplex group and all IRLMs are not enabled for subsecond deadlock processing, message DXR106E is issued.

LTE= *nnnn*

Specifies the number of lock table entries that are to be specified on the next connect to the XCF lock structure. *nnnn* must be a number from 0 through 2048, and it must be an exact power of 2. Each increment in value represents 1 048 576 LTE entries. Note that this parameter is used for data sharing only.

MLT= *nnnnU*

Specifies the upper limit of private storage above the two gigabyte bar, also called the MEMLIMIT, which is managed by MVS. This command allows you to dynamically change the MEMLIMIT for IRLM. This storage is used for locks. The value *nnnnn*. must be a five digit number from 0 through 99999, and *U* must be a one character units indicator with the value of M for megabytes, G for gigabytes, T for terabytes, or P for petabytes. MEMLIMIT specified for IRLM must be larger than both the 2 GB minimum and the amount of above the bar storage in use at the time of the command. If the specified value is out of range or lower than the amount in use (or the 2 GB minimum), then the command is rejected with message DXR106E.

This MEMLIMIT update is temporary, holding only as long as the execution of this IRLM instance. To make a permanent MEMLIMIT change, update the corresponding IRLM startup procedure with a new value for the MEMLIMIT JCL EXEC parameter.

PVT= *nnnn*

Specifies the upper limit of private storage, below the two gigabyte bar. *nnnn* must be a four digit number from 1 through 1800. You can specify this value in megabytes or gigabytes by specifying M (for megabytes) or G (for gigabytes) after the value, as follows, *nnnn* M or *nnnn* G. IRLM monitors the amount of private storage used for locks. If the specified limit is reached, new lock requests will be rejected unless they are **must complete**. If the specified value is out of range or if IRLM's use of private storage is already larger than the specified value, the command is rejected with message DXR106E. No reserve for must complete locks is calculated from the specified PVT= value.

TIMEOUT= *nnnn, subsystem-name*

Requests that IRLM dynamically set the timeout value, in seconds, for the specified subsystem. *nnnn* must be a number from 1 through 3600. *subsystem-name* is the DB2 subsystem name, as displayed by the MODIFY irlmproc,STATUS command.

TRACE= *nnn*

Requests that IRLM dynamically set the maximum number of 64 KB trace buffers per trace type to the value you specify in *nnn*. *nnn* must be a number from 10 through 255. If you specify a value outside of this range, IRLM automatically adjusts the value to a value within the range.

The **default** is 10.

This value is used only when the external CTRACE writer is not active. The trace buffers are allocated from extended common storage area (ECSA).

IRLM does not immediately acquire the number of trace buffers you set using this command; IRLM allocates buffers as needed, not to exceed the number of buffers you specify. If the number of trace buffers that you set is less than the number of currently allocated buffers, IRLM brings the number within your specified range by releasing the oldest buffers at the end of the next deadlock cycle.

Usage notes

Effect of an IRLM restart: The values you set using the MODIFY irlmproc,SET command do not persist through a stop and restart of IRLM. The number of trace buffers for each trace type returns to the default value of 10.

TIMEOUT considerations: The TIMEOUT value must be a multiple of the local deadlock parameter. If the value entered is not an even multiple of the deadlock parameter, IRLM will increase the timeout value to the next highest multiple. This new value is used until the IRLM or identified subsystem is terminated, or the timeout is changed again by the operator. The value specified on the command does *not* affect the timeout value in the DB2 subsystem parameters.

The LTE value is used in the following order:

1. The value specified using the MODIFY irlmproc,SET,LTE= command, if the value is greater than zero.
2. The value from the LTE= in the IRLMPROC, if the value is greater than zero.
3. The value determined by the existing logic, which divides the XES structure size returned on the XESQUERY call by 2 multiplied by the LTE width. The result is rounded to the nearest power of 2, which the existing logic uses for the value.

Note: The LTE width is determined by the MAXUSRS value.

If an attempt is made to use a nonzero value from either option number 1 or 2, and that value is too large for the structure size that is returned on the QUERY, the value from the next option in the sequence is used instead.

Deadlock value range for non-supporting members: When an IRLM that supports subsecond deadlock joins a group that has a member that does not support subsecond deadlock, if the deadlock value of the new member is less than one second, the value is set to one second.

Examples

Example 1: Enter the following command on a z/OS system console:

```
F IR21PROC,SET,TRACE=20
```

Response on the z/OS system console is as follows:

```
DXR177I IR21033 THE VALUE FOR TRACE IS SET TO 20.
```

Example 2: Enter the following command on a z/OS system console:

```
F IR21PROC,SET,TIMEOUT=60,DBMS
```

Response on the z/OS system console is as follows:

```
DXR177I IR21033 THE VALUE FOR TIMEOUT IS SET TO 60 FOR DBMS
```

Example 3: Enter the following command on a z/OS system console:

```
F IR21PROC,SET,LTE=1024
```

Response on the z/OS system console is as follows:

```
DXR177I IR21033 THE VALUE FOR LTE IS SET TO 1024
```

Example 4: Enter the following command on a z/OS system console:

```
F IR21I,SET,DEADLOCK=1000
```

Response on the z/OS system console is as follows:

```
DXR177I IR21033 THE VALUE FOR DEADLOCK IS SET TO 1000 MILLISECONDS
```

Example 5: Enter the following command on a z/OS console:

```
F IR21I,SET,PVT=1000
```

Response from the z/OS system console is as follows:

```
DXR177I IR21033 THE VALUE FOR PVT IS SET TO 1000
```

Example 6: Enter the following command on a z/OS console:

```
F IR21I,SET,MLT=4G
```

Response from the z/OS system console is as follows:

```
DXR177I IR21033 THE VALUE FOR MLT IS SET TO 4G
```

Chapter 53. MODIFY irlmproc,STATUS (z/OS IRLM)

The MODIFY irlmproc,STATUS command displays information for one or more subsystems connected to the IRLM that is specified using *irlmproc* .

Each subsystem connected to the specified IRLM is listed, including subsystem name, status, work unit, lock information, the current values of the IRLM TIMEOUT, and DEADLOCK parameter values. Additionally, you can list an IRLM's ID and service level. For a specified IRLM, you can display the current storage allocated, as well as the greatest amount of storage that was allocated since the last time this IRLM was started.

Abbreviation: F

Environment

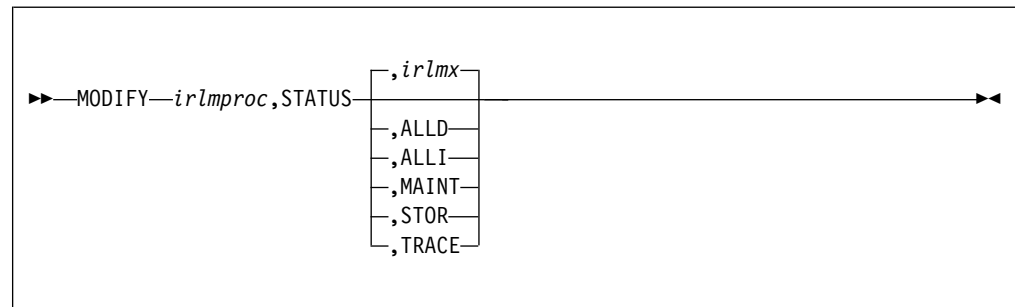
This command can be issued only from a z/OS console.

Data sharing scope: Member or group, depending on which option you choose

Authorization

The command requires an appropriate level of z/OS authority.

Syntax



Option descriptions

irlmproc

Specifies the IRLM that is to process the command.

irlmx

Specifies which IRLM's status is to be displayed. *irlmx* is the concatenation of the IRLM subsystem name and IRLM member ID as specified in the IRLM startup procedure (DB2 installation panel DSNTIPI). An example is DJ2A2 (the member ID is 2).

ALLD

Requests the DB2 subsystem name and status of a DB2 that is identified to an IRLM. In a data sharing group, this command lists information about all DB2 subsystems that are currently identified to an IRLM, assuming that the IRLM on which the command is issued is connected to the data sharing group. You

can determine if the IRLM is connected by issuing a `MODIFY irlmproc,STATUS` command and checking that the output shows `SCOPE=GLOBAL`.

If a DB2 is down and holds retained locks, that DB2 is also displayed. However, the IRLM that is displayed with that DB2 can vary depending on several circumstances:

- Normally, it is the last IRLM to which the DB2 subsystem identified.
- If a rebuild of the lock structure occurred after the retained locks were created, the IRLM with the lowest member ID at the time the rebuild occurred is displayed.
- If a group restart is occurring and one DB2 subsystem is recovering on behalf of another DB2 subsystem, the IRLM that is displayed is the one associated with the DB2 subsystem doing the peer recovery. For example, if DB1A is doing a peer recovery of DB2A, the display might show the following information:

NAME	STATUS	...	IRLM_NAME
DB1A	UP		IRLA
DB2A	DOWN		IRLA

ALLI

Requests the IRLM subsystem name, ID, status, and service level. In a data sharing group, this command lists information about all IRLM subsystems in the data sharing group, assuming that the IRLM on which the command is issued is connected to the data sharing group. You can determine if the IRLM is connected by issuing a `MODIFY irlmproc,STATUS` command and checking that the output shows `SCOPE=GLOBAL`.

In a list of IRLM subsystems in a data sharing group, the name of the IRLM that is the global deadlock manager is followed by an asterisk (*).

If an IRLM is down, it is displayed only if its associated DB2 subsystem is down and holds retained locks. The IRLM that is displayed can vary depending on several circumstances:

- Normally, it is the last IRLM to which the DB2 subsystem identified.
- If a rebuild of the lock structure occurred after the retained locks were created, the IRLM with the lowest member ID at the time the rebuild occurred is displayed.
- If the failed DB2 subsystem had recovery done on its behalf by another DB2 subsystem, the IRLM that is displayed is the one associated with the DB2 subsystem that did the peer recovery.

MAINT

For this IRLM only, displays the maintenance levels of IRLM load module CSECTS in a two-column format.

STOR

For this IRLM only, displays the current and maximum allocation for CSA, ECSA, and private extended storage.

TRACE

Requests information about IRLM subcomponent trace types. Information includes whether a subcomponent trace type is active, how many trace buffers are used by the trace, and whether the component trace external writer is active for the trace.

Output

Output from the command is presented in a DXR message. The command options determine which DXR message is used.

- DXR100I (IRLM)
- DXR101I (IRLM)
- DXR102I (IRLM)
- DXR103I (IRLM)
- DXR104I (IRLM)
- DXR179I (IRLM)

Usage notes

Messages: If *irlmx* is not specified, or if this IRLM is in a non-data-sharing environment, message DXR101I is issued. That message lists each subsystem connected to the IRLM specified by *irlmx*, with an indication as to whether the connection is active.

Displaying IRLM IDs: If *irlmproc* is started specifying SCOPE=GLOBAL, the second line of the display indicates the IRLM IDs of the IRLM subsystems.

Examples

Example 1: Enter the following command on the z/OS system console:

```
MODIFY IRTPROC,STATUS
```

Response on the z/OS system console:

```
DXR101I IR2T001 STATUS SCOPE=LOCAL
DEADLOCK:0500
      SUBSYSTEMS IDENTIFIED          PT01
      NAME      T/OUT  STATUS  UNITS  HELD  WAITING  RET_LKS
      DSNT1     0010   UP      0005   0010   0002     0
```

Explanation: The operator on the z/OS system has requested information about the DB2 systems connected to the IRLM identified by the IRLM procedure named IRTPROC.

If the IRLM is SCOPE=GLOBAL on the *irlmproc* and is not connected to any group, the status message shows:

```
DXR101I IR21001 STATUS SCOPE=DISCON DEADLOCK: dddd SUBSYSTEMS IDENTIFIED
NAME T/OUT STATUS UNITS HELD WAITING RET_LKS
ssname zzz  wun  reh  rew  rtlks
```

Use the information in message DXR101I (IRLM) to interpret the output.

Example 2: Assume that you have a data sharing group. Enter the following command on the system console:

```
MODIFY DB1GIRLM,STATUS,ALLD
```

The response on the system console is as follows:

```

11.11.07 STC00061 DXR102I DJ1G001 STATUS C
SUBSYSTEMS IDENTIFIED
NAME STATUS RET_LKS IRLMID IRLM_NAME IRLM_LEVEL
DB1G UP 0 001 DJ1G 2.022
DB2G UP 0 002 DJ2G 1.022
DXR102I End of display

```

Explanation: The output shows all the DB2 subsystems that are connected to IRLMs in this data sharing group (the group to which the IRLM processing the request belongs).

Use the information in message DXR102I (IRLM) to interpret the output.

Example 3: To display information about a specific member of a data sharing group, enter the following command:

```
MODIFY DB1GIRLM,STATUS,DJ1G002
```

Response on system console is as follows:

```

11.11.21 STC00061 DXR102I DJ1G001 STATUS C
SUBSYSTEMS IDENTIFIED
NAME STATUS RET_LKS IRLMID IRLM_NAME IRLM_LEVEL
DB1G UP 0 002 DJ1G 2.022
DXR102I End of display

```

Explanation: This output shows information similar to the output that is shown in example 1, but this command specifies a specific IRLM in the data sharing group.

Use the information in message DXR102I (IRLM) to interpret the output.

Example 4: Again, assume data sharing is in effect. Enter the following command on the system console:

```
MODIFY DB1GIRLM,STATUS,ALLI
```

The response on the console is as follows:

```

11.11.00 STC00061 DXR103I DJ1G001 STATUS C
IRLMS PARTICIPATING IN DATA SHARING GROUP FUNCTION LEVEL=1.022
IRLM_NAME IRLMID STATUS LEVEL SERVICE MIN_LEVEL MIN_SERVICE
DJ1G 001 UP 2.022 HIR2220 2.022 HIR2220
DJ2G 002 UP 1.022 PQ52360 1.012 PN90337
DXR103I End of display

```

Explanation: The output shows the IRLMs that are participating in this data sharing group (the group which includes the IRLM processing the request).

Use the information in message DXR103I (IRLM) to interpret the output.

Example 5: Assume that this command is issued in a non-data-sharing environment. Enter the following command on the system console:

```
MODIFY DB1GIRLM,STATUS,ALLI
```

The response on the console is as follows:

```

11.11.03 STC00082 DXR103I DJ1G001 STATUS C
IRLMS PARTICIPATING IN DATA SHARING GROUP FUNCTION LEVEL=2.022
IRLM_NAME IRLMID STATUS LEVEL SERVICE MIN_LEVEL MIN_SERVICE
DJ1G 001 UP 2.022 HIR2220 1.022 PQ523690
DXR103I End of display

```


Explanation: The output shows information only for the specified IRLM. The group function level that is shown is the function level for the specified IRLM.

Use the information in message DXR103I (IRLM) to interpret the output.

Example 6: Enter the following command on the system console:

```
MODIFY IR21PROC,STATUS,STOR
```

The response on the console is as follows:

```
DXR100I PR21001 STOR STATS 150
      PC: YES      LTEW: 32 LTE:      OM RLE: 3046      RLEUSE: 0
      BB PVT: 1738M AB PVT (MEMLIMIT): 2G
      CSA USE: ACNT: OK AHWM: OK CUR: 312K HWM: 312K
      ABOVE 16M: 15 312K BELOW 16M: 0 0K
CLASS  TYPE  SEGS    MEM  TYPE  SEGS    MEM  TYPE  SEGS    MEM
ACCNT  T-1    1   2048K  T-2    1   1024K  T-3    1    4K
PROC   WRK    4    20K   SRB    1     1K   OTH    1    1K
MISC   VAR    7   733K  N-V    10   247K  FIX    1   24K
DXR100I  END OF DISPLAY
```

Explanation: The example shows that current storage allocated for IRLM is 312 KB, and the greatest amount that has been allocated since the last time IRLM was started is also 312 KB. The storage for the locking structures (RHB and RLB) is contained within IRLM private storage.

Use the information in message DXR100I (IRLM) to interpret the output.

Example 7: Enter the following command on the system console:

```
MODIFY PR21PROC,STATUS,TRACE
```

The command displays the following output on the system console:

```
DXR179I PR21034 TRACE USAGE
TRACE BUFFER STORAGE IN USE: 256 KB
MAXIMUM NUMBER OF TRACE BUFFERS ALLOWED PER TRACE TYPE: 10
TRACE TYPE  ACTIVE  BUFFERS IN USE  CTRACE WRITER
-----
SLM         N         0              N
XIT         Y         2              N
XCF         N         0              N
DBM         N         0              N
EXP         Y         1              N
INT         Y         1              N
```

Explanation: This example shows that the storage currently allocated for IRLM tracing is 256 KB, the maximum number of trace buffers allowed per trace type is set to 10, and the external CTRACE writer is not active.

Use the z/OS TRACE CT command to activate or deactivate traces. You cannot turn off the EXP and INT traces. The XIT (for data sharing), EXP, and INT traces are automatically activated when you start IRLM. All traces are automatically activated with IRLMPROC TRACE=YES.

The trace size for each buffer is 64 KB. Use the MODIFY irlmproc,SET,TRACE=*nnn* command to change the maximum number of trace buffers.

Use the information in message DXR179I (IRLM) to interpret the output.

Example 8: Enter the following command on the system console:

```
MODIFY IR21I,STATUS,MAINT
```

The command displays the following output on the system console:

```
DXR104I IR21240 MAINTENCE LEVELS
      LMOD.Csect      MaintLv  Date          Csect  APAR    DATE
DXRRLM00.DXRRL010  PQ35083 02/22/00  DXRRL020 PQ35083 02/22/00
              DXRRL030  PQ27464 08/18/99  DXRRL040 PQ35083 02/22/00
```

Explanation: The output shows the maintenance levels of IRLM load module CSECTS in a two-column format.

Use the information in message DXR104I (IRLM) to interpret the output.

Chapter 54. -MODIFY TRACE (DB2)

The DB2 command MODIFY TRACE changes the IFCIDs (trace events) associated with a particular active trace.

The DB2 command MODIFY TRACE completes the following actions:

- Changes the trace events (IFCIDs) being traced for a particular active trace.
- Stops any IFCID previously active for the specified trace.
- Writes statistics records.

Abbreviation: -MOD TRA

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

Traces started by a IFI/IFC program: Before you modify an active trace, ensure that an IFI application program or the IFC Selective Dump utility (DSN1SDMP) did not start the trace. If you modify a trace started by DSN1SDMP, the DSN1SDMP utility abnormally terminates. When DSN1SDMP terminates, it stops the trace. This stop could interfere with the MODIFY TRACE command, which stops and restarts the trace.

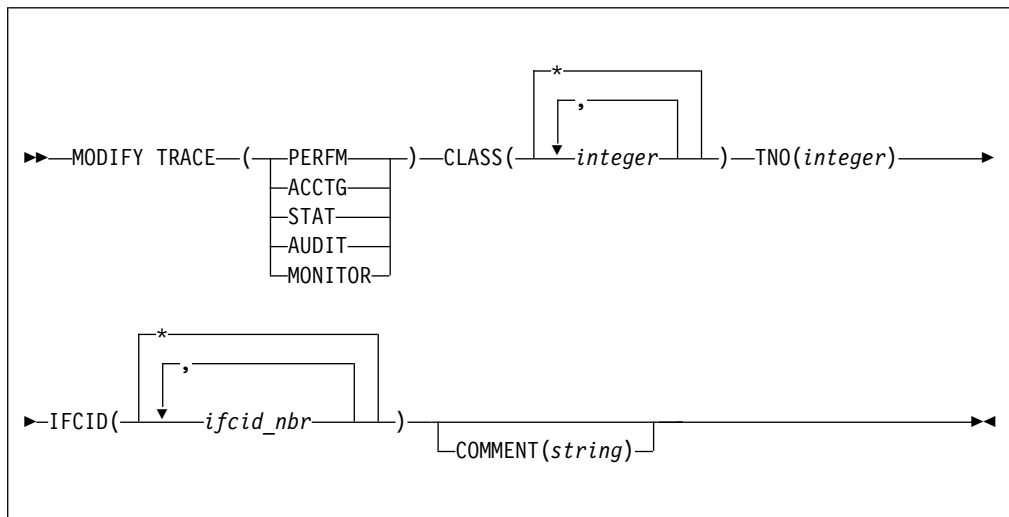
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- TRACE privilege
- SQLADM authority
- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority
- SECADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

TRACE

Determines which IFCIDs are started. The following table lists each trace type, its abbreviation, and a brief description of each type.

One additional trace type is not described in the table because it is intended for service and is to be used under the direction of IBM Software Support.

Table 22. Trace types

Type	Abbreviation	Description
PERFM	P	Performance records of specific events
ACCTG	A	Accounting records for each transaction
STAT	S	Statistical data
AUDIT	AU	Audit data
MONITOR	MON	Monitor data

CLASS(*integer* , ...)

Limits the list to IFCIDs started for specified classes.

Abbreviation: C

integer is a class to which the list of IFCIDs started is limited.

The **default** is CLASS() , which starts all default IFCID classes.

TNO(*integer*)

Specifies the particular trace to be modified, identified by its trace number (1 to 32, 01 to 09). You can specify only one trace number. TNO is a required option for the MODIFY TRACE command.

No default exists for the TNO keyword.

IFCID(*ifcid_nbr* , ...)

Specifies which other IFCIDs (trace events), in addition to those IFCIDs contained in the classes specified in the CLASS option, are to be started. To

start only those IFCIDs specified in the IFCID option, use trace classes 30-32. These classes have no predefined IFCIDs and are available for a location to use.

If you do not specify the IFCID option, only those IFCIDs contained in the activated trace classes are started.

The maximum number of IFCIDs is 156. The range of values that are valid for the IFCID option is 1 through 350, with the exception of: 4, 5, 185, 187, 217, 232, 234, 240, and 241.

The **default** is IFCID(*).

COMMENT (*string*)

Specifies a comment that is reproduced in the trace output record (except in the resident trace tables).

string is any character string; it must be enclosed between apostrophes if it includes a blank, comma, or special character.

Example

Example 1: Change trace number six so that it collects only statistics and accounting data. You can define CLASS(30) at your site.

```
-MODIFY TRACE(S) IFCID(1,2,3) TNO(6) CLASS(30)  
  COMMENT ('STATS AND ACCOUNTING ON')
```

Related reference:

- STOP TRACE (DB2)
- START TRACE (DB2)
- DISPLAY TRACE (DB2)

Chapter 55. REBIND PACKAGE (DSN)

The DSN subcommand REBIND PACKAGE rebinds an application package when you make changes that affect the package, but have not changed the SQL statements in the program.

For example, you can use REBIND PACKAGE after you complete the following activities:

- Migrate to a new DB2 release
- Apply maintenance to DB2
- Modify authorizations
- Create a new index that statements in the package can use
- Collect statistics using the RUNSTATS utility or with another utility by using the STATISTICS keyword

When the REBIND PACKAGE(*) command is issued, trigger packages are not affected.

REBIND PACKAGE is generally faster and more economical than BIND PACKAGE. You should use BIND PACKAGE with the ACTION(REPLACE) option under the following conditions:

- When you change the SQL statements
- When you recompile the program
- When you have previously run BIND PACKAGE with the SQLERROR(CONTINUE) option

Environment

You can use REBIND PACKAGE through DB2I, or enter the REBIND PACKAGE subcommand from a DSN session running in foreground or background.

Data sharing scope: Group

Authorization

The package owner must have authorization to execute **all** SQL statements embedded in the package for REBIND PACKAGE to build a package without producing error messages. For VALIDATE(BIND), DB2 verifies the authorization at bind time. For VALIDATE(RUN), DB2 verifies the authorization initially at bind time, but if the authorization check fails, DB2 rechecks it at run time.

When the EXPLAIN(ONLY) the option is specified, you must have the EXPLAIN privilege.

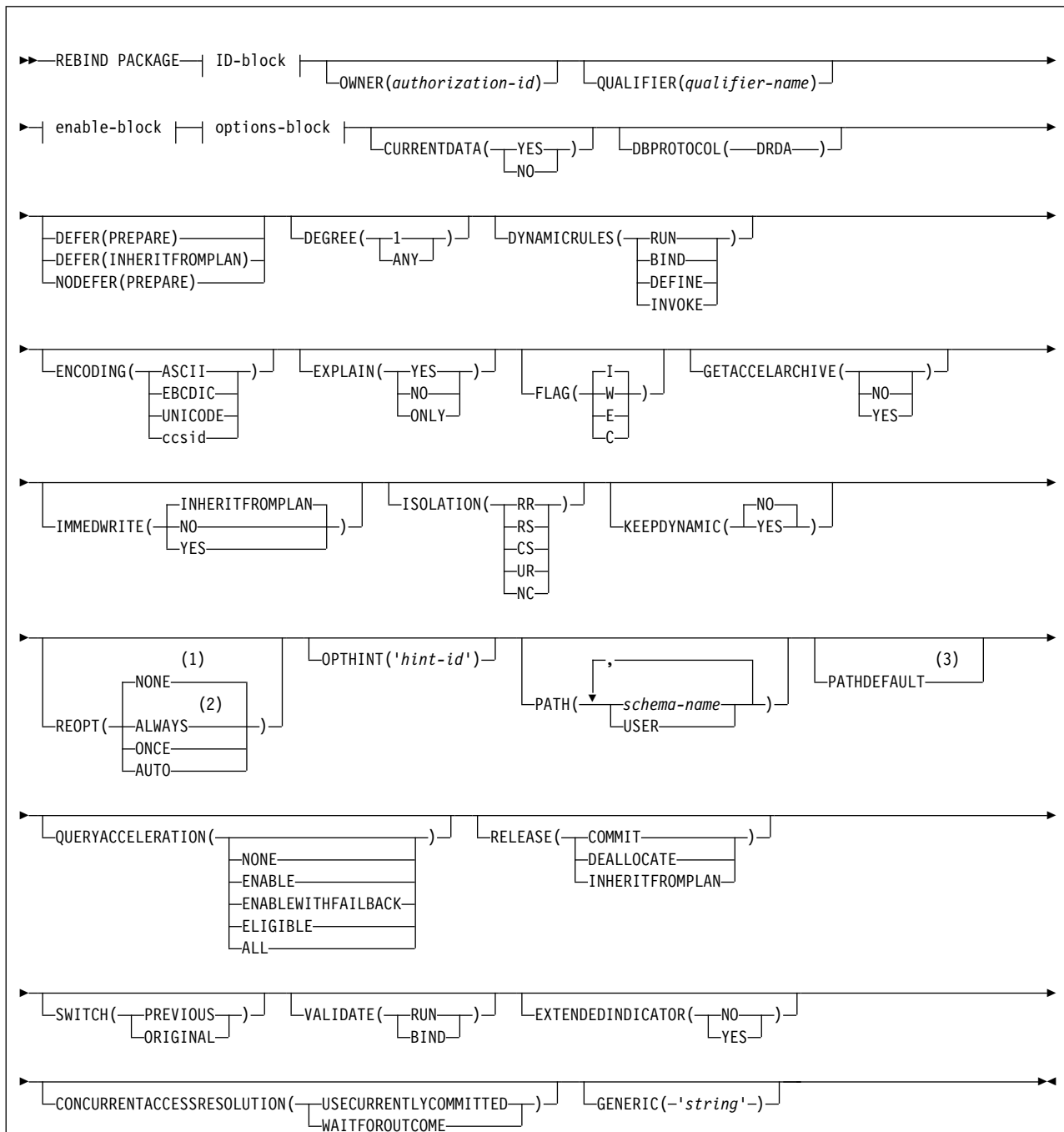
The package owner must be a role to execute REBIND PACKAGE in a trusted context with role ownership.

The following table explains the authorization required to run REBIND PACKAGE, depending on the options specified.

Table 23. Summary of privileges for REBIND PACKAGE

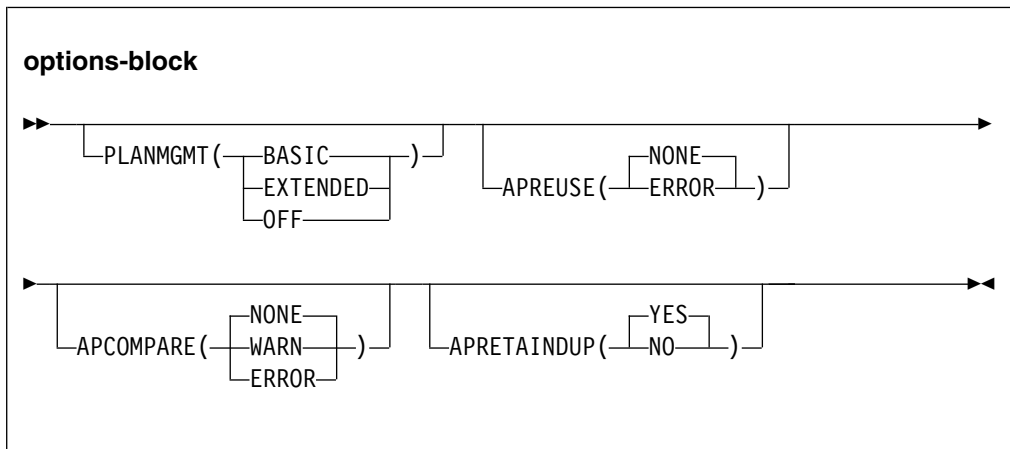
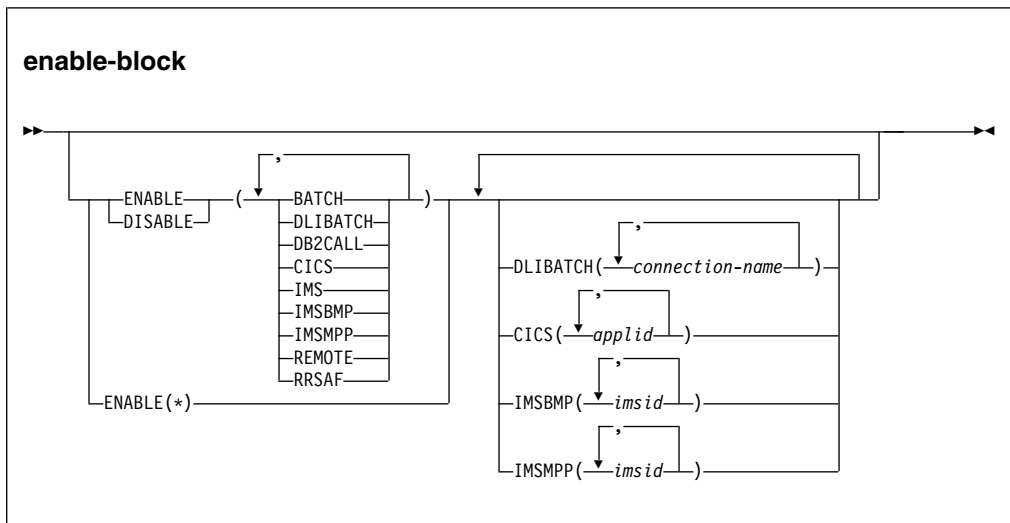
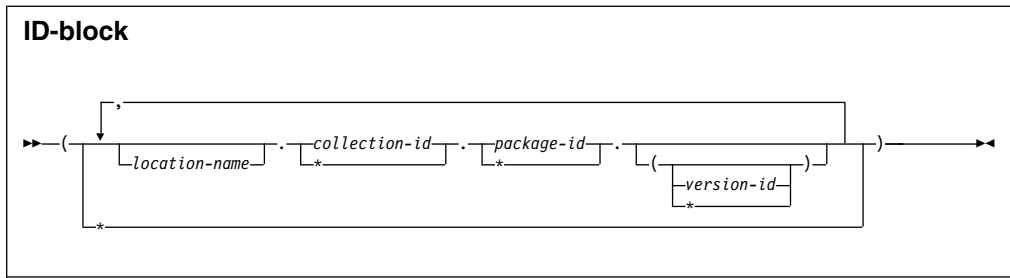
Option	Authorization required to run REBIND PACKAGE
REBIND PACKAGE with no change in ownership because the OWNER keyword is not specified.	<p>The authorization IDs of the process must have one of the following authorities:</p> <ul style="list-style-type: none"> • Ownership of the package • BIND privilege on the package • BINDAGENT privilege from the owner of the package • PACKADM authority on the collection or on all collections • SYSADM or SYSCTRL or System DBADM authority
REBIND PACKAGE with no change in ownership, although the original owner is specified for the OWNER keyword.	<p>The authorization IDs of the process must have one of the following authorities:</p> <ul style="list-style-type: none"> • OWNER <i>authorization-id</i> must be one of the primary or secondary authorization IDs of the binder • BINDAGENT privilege from the owner of the package
REBIND PACKAGE with change of ownership. (An authorization ID that is not the original owner is specified in the OWNER keyword.)	<p>The new OWNER must have one of the following authorities:</p> <ul style="list-style-type: none"> • BIND privilege on the package • PACKADM authority on the collection or on all collections • SYSADM or SYSCTRL or System DBADM authority <p>Specifying the OWNER: If any of the authorization IDs have the BINDAGENT privilege granted from the owner, the <i>authorization-id</i> can specify the grantor as OWNER. Otherwise, OWNER <i>authorization-id</i> must be one of the primary or secondary authorization IDs of the binder.</p>

Syntax



Notes:

- 1 NOREOPT(VARS) can be specified as a synonym of REOPT(NONE)
- 2 REOPT(VARS) can be specified as a synonym of REOPT(ALWAYS)
- 3 The PATHDEFAULT keyword is mutually exclusive with the PATH keyword. Do not specify both keywords in the same REBIND command.



Option descriptions

For descriptions of the options shown in the syntax diagram, see the topic Chapter 18, “BIND and REBIND options for packages and plans,” on page 101.

Usage notes

Rebinding multiple packages: If you rebind multiple packages, DB2 commits each successful rebind before rebinding the next package.

Rebinding a package for a native SQL procedure: If you issue a REBIND PACKAGE command against a native SQL procedure package, the only bind options that you can change are EXPLAIN, PLANMGMT, SWITCH,

| APRETAINDUP, APREUSE, and APCOMPARE. If you try to change other bind
| options the command will fail and return message DSNT215I. The REBIND
| PACKAGE command rebinds only the SQL statements included in the procedure
| and not the control statements in the procedure definition.

| **Rebinding a package for an SQL function:** If you issue a REBIND PACKAGE
| command against a package for an SQL function, the only bind options that you
| can change are EXPLAIN, PLANMGMT, and SWITCH. If you try to change other
| bind options, the command will fail and return message DSNT215I. The REBIND
| PACKAGE command rebinds only the SQL statements included in the function
| and not the control statements in the function definition.

Example

Rebind packages TEST.DSN8BC81.(MAY_VERSION) and PRODUCTION.DSN8BC81.(DEC_VERSION), both of which are located at the local location USIBMSTODB22. The packages can run only from the CICS or the DLIBATCH environments if the connection ID is CON2. This replaces the CON1 that is specified on the BIND PACKAGE command.

```
REBIND PACKAGE (USIBMSTODB22.TEST.DSN8BC81.(MAY_VERSION),  
                USIBMSTODB22.PRODUCTION.DSN8BC81.(DEC_VERSION)) -  
  ENABLE (CICS,DLIBATCH) CICS (CON2)
```

Related reference:

REBIND PLAN (DSN)

BIND PACKAGE (DSN)

BIND and REBIND options for packages and plans

Chapter 56. REBIND PLAN (DSN)

The DSN subcommand REBIND PLAN rebinds an application plan when you make changes to the attributes of the plan, such as the package list.

For example, you can use REBIND PLAN when you change authorizations, modify package lists for the plan, or use RUNSTATS. If the rebind is successful, the process prepares an application plan and updates its description in the catalog table SYSPLAN.

REBIND PLAN is generally faster and more economical than BIND PLAN. But if you change the SQL statements or recompile a program, you should use BIND PLAN with the option ACTION(REPLACE).

Environment

You can use REBIND PLAN through DB2I, or enter the REBIND PLAN subcommand from a DSN session running in foreground or background.

Data sharing scope: Group

Authorization

The plan owner must have authorization to execute **all** SQL statements embedded in the plan for REBIND PLAN to build a plan without producing error messages. For VALIDATE(BIND), DB2 verifies the authorization at bind time. For VALIDATE(RUN), DB2 initially verifies the authorization at bind time, but if the authorization check fails, DB2 rechecks it again at run time. If you use the PKLIST keyword, you must have EXECUTE authority for the packages or collections specified on PKLIST.

The plan owner must be a role to execute REBIND PLAN in a trusted context with role ownership.

The following table explains the authorization required to run REBIND PLAN, depending on the options specified.

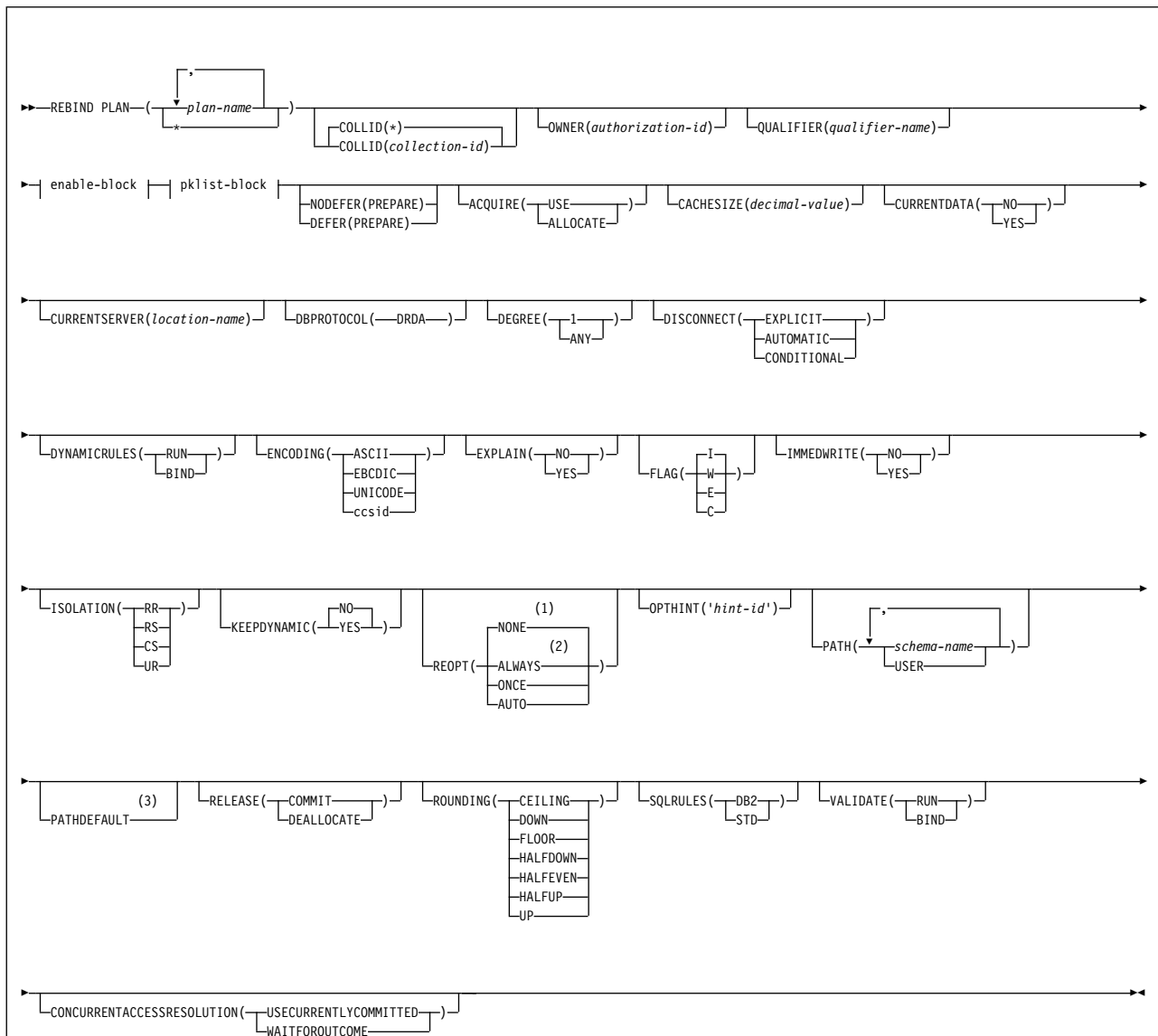
Table 24. Summary of privileges for REBIND PLAN

Option	Authorization required to run REBIND PLAN
REBIND PLAN with no change in ownership because the OWNER keyword is not specified.	The authorization IDs of the process must have one of the following authorities: <ul style="list-style-type: none">• Ownership of the plan• BIND privilege on the plan• BINDAGENT privilege from the owner of the plan• SYSADM or SYSCTRL or System DBADM authority
REBIND PLAN with no change in ownership, although the original owner is specified for the OWNER keyword.	The authorization IDs of the process must have one of the following authorities: <ul style="list-style-type: none">• OWNER <i>authorization-id</i> must be one of the primary or secondary authorization IDs of the binder• BINDAGENT privilege from the owner of the plan• SYSADM or SYSCTRL or System DBADM authority

Table 24. Summary of privileges for REBIND PLAN (continued)

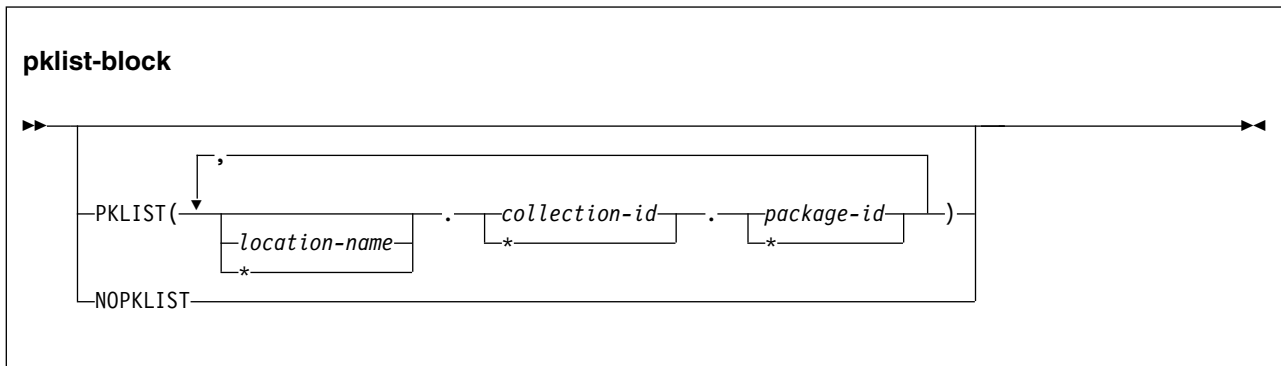
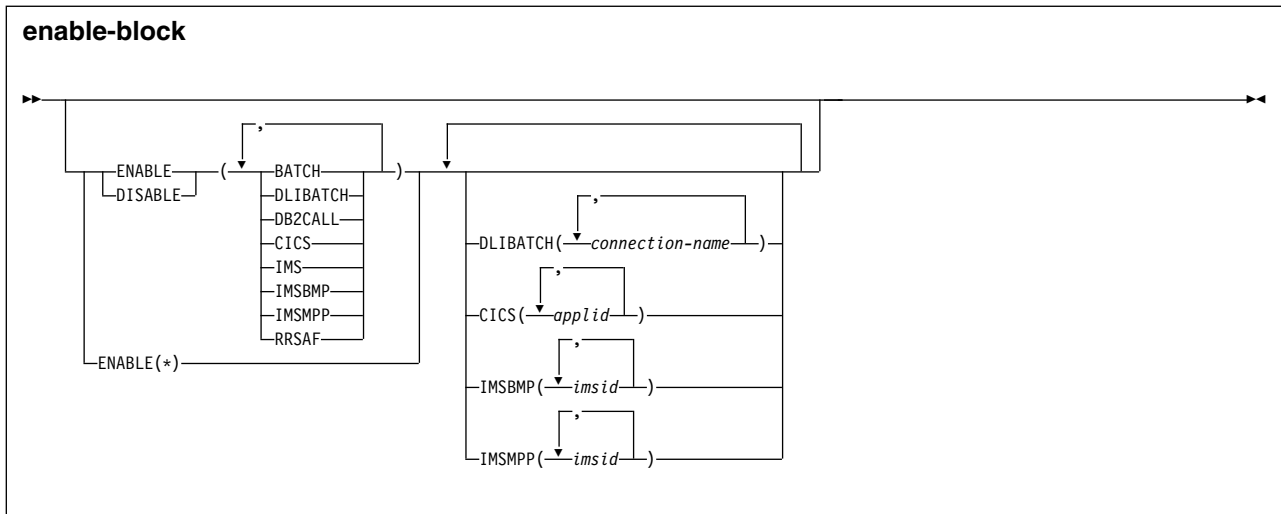
Option	Authorization required to run REBIND PLAN
REBIND PLAN with change of ownership. (An authorization ID that is not the original owner is specified in the OWNER keyword.)	<p>The new OWNER must have one of the following authorities:</p> <ul style="list-style-type: none"> • BIND privilege on the plan • SYSADM or SYSCTRL or System DBADM authority <p>Specifying the OWNER: If any of the authorization IDs has the BINDAGENT privilege granted from the owner, then <i>authorization-id</i> can specify the grantor as OWNER. Otherwise, OWNER <i>authorization-id</i> must be one of the primary or secondary authorization IDs of the binder.</p>
COLLID, specifying (*), indicating all packages in the collection	You do not need any authorization privileges for this option.
COLLID, specifying individual packages	<p>Authorization ID of the process must include one of the following authority:</p> <ul style="list-style-type: none"> • CREATEIN privilege on the COLLID you specified.
PKLIST, specifying individual packages	<p>Authorization ID of the process must include one of the following authorities:</p> <ul style="list-style-type: none"> • EXECUTE privilege on each package specified in the PKLIST • PACKADM authority on specific collections containing packages or on collection * • SYSADM or DATAACCESS authority
PKLIST, specifying (*), indicating all packages in the collection	<p>Authorization ID of the process must include one of the following authorities:</p> <ul style="list-style-type: none"> • EXECUTE privilege on <i>collection-id</i> .* • PACKADM authority on <i>collection-id</i> or on * • SYSADM or DATAACCESS authority

Syntax



Notes:

- 1 REOPT(VARS) can be specified as a synonym of REOPT(ALWAYS)
- 2 NOREOPT(VARS) can be specified as a synonym of REOPT(NONE)
- 3 The PATHDEFAULT keyword cannot be specified with the PATH keyword.



Option descriptions

For descriptions of the options shown in the syntax diagram, see the topic Chapter 18, “BIND and REBIND options for packages and plans,” on page 101.

Usage notes

Rebinding multiple plans: If you rebind multiple plans, DB2 commits each successful rebind before rebinding the next plan.

When you cannot rebind a plan: You cannot rebind a plan while that plan is executing.

Examples

Example: Rebinding a plan to replace the package list

Suppose that PLANA uses package list COLLA.* Suppose that you want to replace that package list with COLLB.* Issue a command like this one:

```

REBIND PLAN (PLANA) -
  PKLIST(COLLB.*) -
  FLAG(W) -
  VALIDATE(BIND) -
  ISOLATION(CS)
  
```

This REBIND command also does the following things:

- Uses FLAG(W) to issue warning, error, and completion messages, but not informational messages.
- Uses VALIDATE(BIND) to point out any error conditions during the bind process.
- Uses ISOLATION(CS) to prevent other applications from changing the database values that this application uses only while the application is using them. This isolation level protects changed values until the application commits or terminates. In this example, the isolation is not set for the packages, so ISOLATION(CS) becomes the isolation level for the plan and the packages.
- Omits the OWNER keyword to leave the plan's owner authorization ID the same.
- Omits the ENABLE or DISABLE keywords to use the connections previously defined for the plan.

Related reference:

REBIND PACKAGE (DSN)

BIND PLAN (DSN)

BIND and REBIND options for packages and plans

Chapter 57. REBIND TRIGGER PACKAGE (DSN)

The DSN subcommand REBIND TRIGGER PACKAGE rebinds a package that was created when DB2 executed a CREATE TRIGGER statement.

You can use this subcommand to change a limited subset of the default bind options that DB2 used when creating the package. You might also rebind a trigger package to re-optimize its SQL statements after you create a new index or use the RUNSTATS utility. Additionally, you can rebind a trigger package if it has been marked invalid because an index, or another object it was dependent on, was dropped.

If the rebind is successful, the trigger package is marked valid. When REBIND TRIGGER PACKAGE(*) is issued, the rebind will affect all trigger packages that the issuer is authorized to rebind.

Trigger packages cannot be rebound remotely. The location name is permitted when specifying the package name on a REBIND TRIGGER PACKAGE subcommand. However, the location name must not refer to a remote location.

Environment

You can use REBIND TRIGGER PACKAGE through DB2I, or enter the REBIND TRIGGER PACKAGE subcommand from a DSN session that is running in foreground or background.

Data sharing scope: Group

Authorization

To build a package without producing error messages, the package owner must have authorization to execute **all** SQL statements that are embedded in the package for REBIND TRIGGER PACKAGE.

To execute this subcommand, you must use a privilege set of the process that includes one of the following privileges or authorities:

- Ownership of the trigger package
- BIND privilege on the trigger package
- BINDAGENT privilege from the owner of the trigger package
- PACKADM authority on the collection or on all collections
- System DBADM authority
- SYSCTRL authority
- SYSADM authority

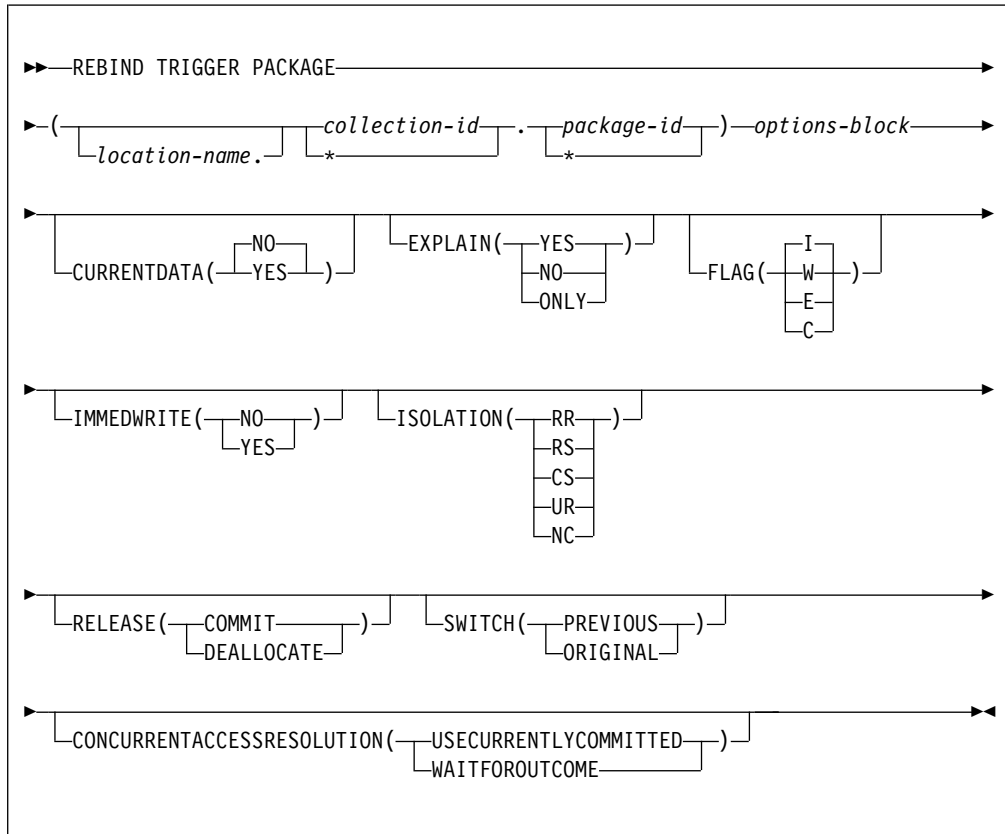
When the trigger package is bound, the privileges of the current authorization ID are used when checking authority to bind statements within the triggered action. On REBIND TRIGGER PACKAGE, you need one of the following privileges or authorities:

- Ownership of the trigger package
- BIND privilege on the trigger package

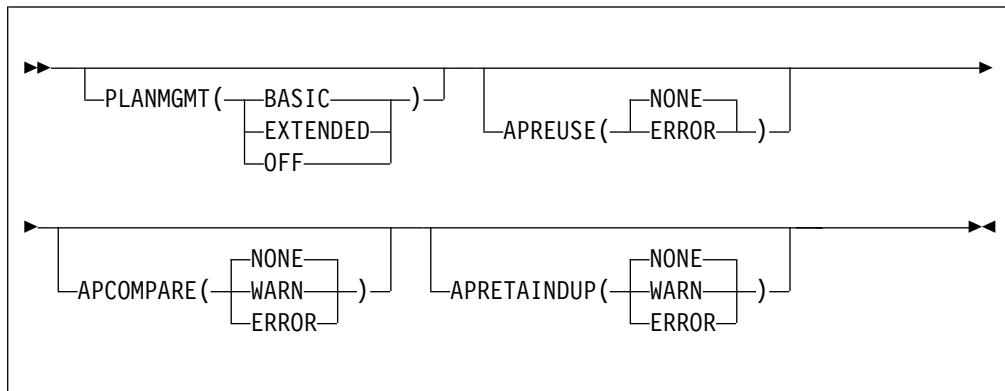
- BINDAGENT privilege from the owner of the trigger package
- PACKADM authority on the collection or on all collections
- System DBADM authority
- SYSCtrl authority
- SYSADM authority

When the EXPLAIN(ONLY) the option is specified, you must have the EXPLAIN privilege.

Syntax



options-block



Option descriptions

TRIGGER PACKAGE

Determines what trigger package or packages to rebind.

The following options identify the location, collection, and package name of the package. You can identify a location and collection. For REBIND TRIGGER PACKAGE, you must identify a trigger package name.

location-name

Identifies the current local location. Remote rebind of a trigger package is not allowed. *location-name* is the location of the DBMS where the package rebinds and where the description of the package resides.

The **default** is the local DBMS.

collection-id or *

Identifies the schema-name that already contains the trigger package to rebind. No default exists.

For REBIND TRIGGER, you can use an asterisk (*) to rebind all local packages with the specified *package-id* in all the collections for which you have bind privileges.

package-id or *

Identifies the name of the trigger package to rebind, as listed in the NAME column of the SYSPACKAGE catalog table. No default exists.

You can use the pattern-matching character (*) to rebind all local triggers in *collection-id* for which you have bind privileges.

For descriptions of the options that are shown in the syntax diagram, see the topic Chapter 18, "BIND and REBIND options for packages and plans," on page 101.

Usage notes

Restrictions on trigger packages: A trigger package can be explicitly rebound, but it cannot be explicitly bound using the BIND PACKAGE subcommand.

A trigger package cannot be explicitly freed using the FREE PACKAGE subcommand or the DROP PACKAGE statement. Use the DROP TRIGGER statement to delete the trigger package.

A trigger package cannot be copied, and it can only be rebound locally. Remote rebind of a trigger package is not allowed.

Rebinding multiple trigger packages: If you rebind multiple trigger packages, DB2 commits each successful rebind before rebinding the next package.

Restriction on trigger names: Although DBCS characters are generally allowed in trigger names, trigger names that contain DBCS characters cannot be used in REBIND TRIGGER PACKAGE operations.

Output

REBIND TRIGGER PACKAGE updates the COLLID and NAME columns in the SYSPACKAGE catalog table.

Example


Enter the following command to rebind trigger package TRIG1 in the ADMF001 collection of packages:

```
REBIND TRIGGER PACKAGE (ADMF001.TRIG1);
```

This command produces output that is similar to the following output:

```
DSNT254I - DSNTBRB2 REBIND OPTIONS FOR
          PACKAGE = STLEC1.ADMF001.TRIG1.()
          ACTION
          OWNER          ADMF001
          QUALIFIER      ADMF001
          VALIDATE       BIND
          EXPLAIN        NO
          ISOLATION      CS
          RELEASE        COMMIT
          COPY
DSNT255I - DSNTBRB2 REBIND OPTIONS FOR
          PACKAGE = STLEC1.ADMF001.TRIG1.()
          SQLERROR       NOPACKAGE
          CURRENTDATA    YES
          DEGREE         1
          DYNAMICRULES  BIND
          NODEFER        PREPARE
          REOPT          NONE
          KEEPDYNAMIC    NO
          DBPROTOCOL     DRDA
          QUERYOPT       1
          PATH
"SYSIBM", "SYSFUN", "SYSPROC", "SYSADM", "ADMF001"
DSNT232I - SUCCESSFUL REBIND FOR
          PACKAGE = STLEC1.ADMF001.TRIG1.()
```

Related concepts:

 [Package copies \(DB2 Performance\)](#)

Related reference:

[BIND and REBIND options for packages and plans](#)

 [CREATE TRIGGER \(DB2 SQL\)](#)

Chapter 58. -RECOVER BSDS (DB2)

The DB2 command RECOVER BSDS reestablishes dual bootstrap data sets (BSDS) after one has been disabled by a data set error.

Follow these steps to reestablish dual BSDS mode:

1. Use access method services to rename or delete the failing BSDS, which the DB2 system has deallocated, and define a new BSDS with the same name as the failing BSDS. You can find control statements in job DSNTIJIN.
2. Issue the DB2 command RECOVER BSDS to make a copy of the remaining BSDS in the newly allocated data set and to reestablish dual BSDS mode.

Abbreviation: -REC BSDS

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- BSDS privilege
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax

```
▶▶—RECOVER BSDS—————▶▶
```

Example

Reestablish dual BSDS mode.

```
-RECOVER BSDS
```

Related tasks:

- ▶ Adding a second BSDS (DB2 Installation and Migration)
- ▶ Recovering the BSDS from a backup copy (DB2 Administration Guide)

Chapter 59. -RECOVER INDOUBT (DB2)

The DB2 command RECOVER INDOUBT recovers threads that are left in an indoubt state because DB2 or a transaction manager could not automatically resolve the indoubt status with the commit coordinator.

This command should only be used when automatic resolution will not work. The commit coordinator must determine the commit or abort decision.

Abbreviation: -REC IND

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

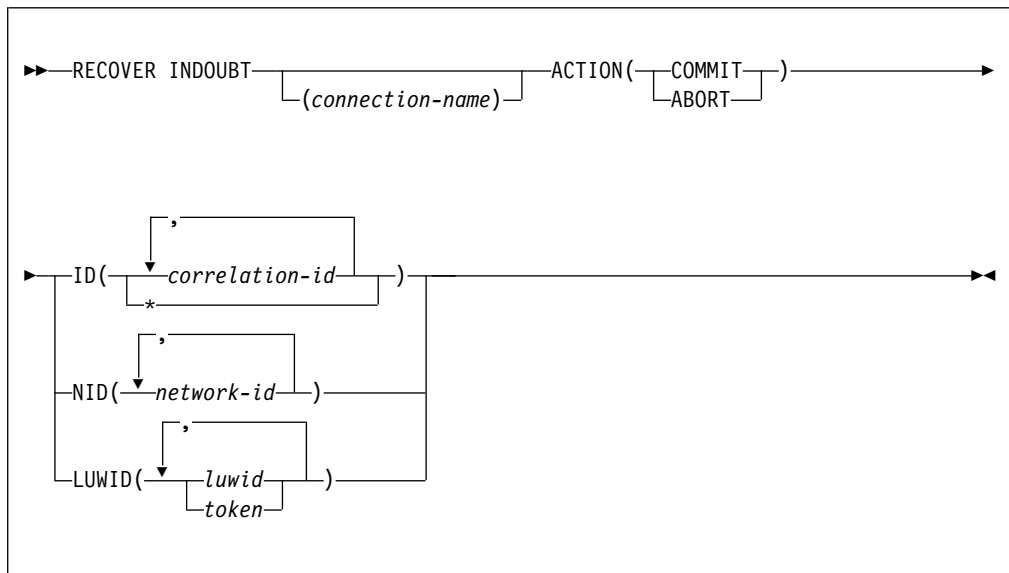
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- | • RECOVER privilege
- | • System DBADM authority
- | • SYSOPR authority
- | • SYSCTRL authority
- | • SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

(*connection-name*)

Specifies a one- to eight-character connection name. Allied threads (including those that are distributed) that belong to the connection name are recovered. This parameter is ignored if LUWID is specified.

The **default** is the connection name from which you enter the command. If you enter this command from a z/OS console, and you are recovering an allied thread using the ID or NID parameter, you **must** supply a connection name; no default connection name is available.

ACTION

Specifies whether to commit or abort the indoubt thread. If there are any downstream participants for which the local thread is the coordinator, the commit or abort decision is propagated to these participants.

Abbreviation: ACT

(COMMIT)

Commits the thread.

(ABORT)

Aborts the thread.

ID(*correlation-id* , ...)

Specifies whether to recover a specific allied thread or all allied threads (including those that are distributed) that are associated with the connection name.

correlation-id

Is the correlation ID (1 to 12 characters) of a specific thread that is to be recovered. If you use more than one correlation ID, separate the IDs with commas.

Do not use a correlation ID that is associated with more than one network ID. Instead, use the NID option.

(*)

Recovers all indoubt threads that are associated with the connection name. Even threads that have the same correlation ID are resolved.

NID(*network-id* , ...)

Specifies the network IDs of threads to recover.

network-id is a network ID that is associated with an individual thread. A single connection might have multiple network IDs associated with it. *network-id* is one of the following types of IDs:

- For IMS and CICS connections, *network-id* is specified as *net-node.number*:

net-node

The network node name of the system that originated the unit of work. *net-node* is one to eight bytes in length.

number

A unique number within the system of origin. *number* is 1 to 16 bytes in length.

- For RRSAF connections, *network-id* is the z/OS RRS unit of recovery ID (URID) that is used to uniquely identify a unit of work. A z/OS RRS URID is a 32-byte number.

The network ID appears in the recovery log of the commit coordinator as a 16-byte unique identification of a unit of work.

- For IMS and CICS connections, the network ID is an 8-byte node name immediately followed by an 8-byte number.
- For RRSAF connections, the network ID is a 16-byte number.

LUID

Recovers the indoubt thread that has the specified LUWID.

luwid

Consists of an LU network name, an LUW instance number, and a commit sequence number.

The LU network name consists of a one- to eight-character network ID, a period, and a one- to eight-character network LU name. The LUW instance number consists of a period followed by 12 hexadecimal characters. The last element of the LUWID is the commit sequence number of 4 hexadecimal characters, preceded by a period.

token

A token is an alternate way to express an LUWID. DB2 assigns a token to each thread that it creates. It is a one- to six-digit decimal number that appears after the equal sign in all DB2 messages that display a LUWID.

If you enter one- to six-decimal digits, DB2 assumes that you are supplying a token. The token that DB2 assigns to a specific LUWID is unique for that DB2 subsystem, but not necessarily unique across all subsystems.

Usage note

When to use a network ID: A *network-id* is not normally needed, because a *correlation-id* can identify indoubt threads. However, if the *correlation-id* is not unique, *network-id* must be used. You do not need a *network-id* if you specify a LUWID.

If you specify a thread in the command that is part of a global transaction, the command is executed against all threads that are in the global transaction.

Examples

Example 1: Recover indoubt allied threads. Schedule a commit for all threads that are associated with the connection name from which the command is entered.

```
-RECOVER INDOUBT ACTION(COMMIT) ID(*)
```

Example 2: Recover an indoubt thread from a remote requester. Schedule a commit for the indoubt thread whose token is 1332.

```
-RECOVER INDOUBT ACTION(COMMIT) LUWID(1332)
```

Example 3: Recover indoubt threads from remote requesters. Schedule an abort for two indoubt threads. The first thread has an LUWID of DB2NET.LUNSITE0.A11A7D7B2057.0002. (The 0002 in the last segment of the LUWID represents the commit sequence number.) The second thread has a token of 442.

```
-RECOVER INDOUBT ACTION(ABORT)  
LUWID (DB2NET.LUNSITE0.A11A7D7B2057.0002, 442)
```

Chapter 60. -RECOVER POSTPONED (DB2)

The DB2 command RECOVER POSTPONED completes back-out processing for units of recovery that are left incomplete during an earlier restart (POSTPONED ABORT units of recovery). Use this command when automatic resolution is not selected.

Abbreviation: -REC POST

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), or an IMS or CICS terminal.

Data sharing scope: Member

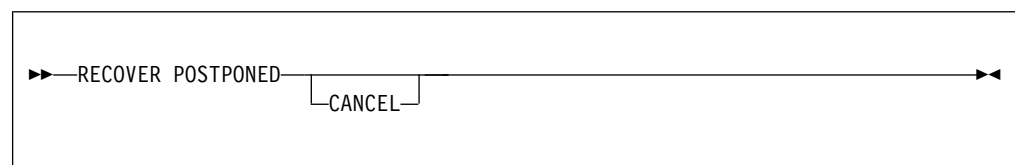
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- RECOVER privilege
- System DBADM authority
- SYSOPR authority
- SYSTRM authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

CANCEL

Specify to stop DB2 processing of all postponed abort units of recovery immediately. Canceling postponed abort units of recovery leaves objects in an inconsistent state.

Objects that the postponed units of recovery modify are recovered (backed out). If back out processing fails, the objects are marked as REFRESH PENDING (REFP) and either RECOVER PENDING (RECP) or REBUILD PENDING (RBDP or PSRBD) in the database exception table. Resolve the REFP status of the object by running the RECOVER utility to recover the object to a prior point in time or by running LOAD REPLACE on the object.

Usage notes

Recovery action: Recovery (rollback) action is always taken for all POSTPONED ABORT units of recovery.

Output

The output from RECOVER POSTPONED consists of informational messages only.

Progression of RECOVER POSTPONED: Message DSNI024I indicates the completion of backout work against the page set or partition, and the removal of the page set or partition from the restart-pending status.

If backout processing lasts for an extended period of time, progress message DSNR047I is displayed at periodic intervals until backout processing is complete.

DB2 issues message DSN9022I after successful completion of the RECOVER POSTPONED command, or message DSN9023I if the command completed unsuccessfully. Message DSNV434I indicates that RECOVER POSTPONED was issued when no postponed-abort units of recovery needed to be resolved.

Example

Enter the following command to recover postponed-abort units of recovery.

```
-RECOVER POSTPONED
```



If postponed-abort units of recovery are found, output that is similar to the following output is generated:


```
DSNV435I - RESOLUTION OF POSTPONED ABORT URS HAS BEEN SCHEDULED
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
          FOR PAGESET DBKD0103.IPKD013A PART 00000004.
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
          FOR PAGESET DBKD0103.TPKD0103 PART 00000004.
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
          FOR PAGESET DBKD0103.IXKD013C PART (n/a).
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
          FOR PAGESET DBKD0103.IUKD013B PART (n/a).
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
          FOR PAGESET DBKD0103.IPKD013A PART 00000002.
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
          FOR PAGESET DBKD0103.TPKD0103 PART 00000002.
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
          FOR PAGESET DBKD0101.IXKD011C PART (n/a).
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
          FOR PAGESET DBKD0101.IXKD011B PART (n/a).
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
          FOR PAGESET DBKD0101.IUKD011A PART (n/a).
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
          FOR PAGESET DBKD0101.TLKD0101 PART (n/a).
DSN9022I - DSNVRP 'RECOVER POSTPONED' NORMAL COMPLETION
```

If no postponed units of recovery are found, the following output is returned:

```
DSNV434I - DSNVRP NO POSTPONED ABORT THREADS FOUND
DSN9022I - DSNVRP 'RECOVER POSTPONED' NORMAL COMPLETION
```

Related tasks:

-  [Resolving postponed units of recovery \(DB2 Administration Guide\)](#)
-  [Recovering from an error during RECOVER POSTPONED processing \(DB2 Administration Guide\)](#)

 [Deferring restart processing \(DB2 Administration Guide\)](#)

Chapter 61. -REFRESH DB2,EARLY (DB2)

The DB2 command -REFRESH DB2,EARLY reloads the ERLY code modules that were loaded at IPL time, and rebuilds the ERLY control block.

| Executing the -REFRESH DB2,EARLY command is an alternative to doing an IPL
| of z/OS for activating maintenance to early (ERLY) code after you apply that
| maintenance to the *prefix.SDSNLINK* data set.

When -REFRESH DB2,EARLY is run, previous copies of the ERLY modules are deleted from the system the next time that DB2 is started. This command is only valid when DB2 is inactive.

| **Restriction:** The data set that contains the new ERLY code must have been in the
| link list concatenation since z/OS IPL time. -REFRESH DB2,EARLY does not search
| for ERLY code in data sets that were added to the link list after IPL time.

| **Important:** If *prefix.SDSNLINK* is LLA-managed, you must perform an LLA
| refresh after you apply ERLY code maintenance, and before you issue the
| -REFRESH DB2,EARLY command. You can perform an LLA refresh by issuing the
| DB2 for z/OS command MODIFY LLA,REFRESH.

Abbreviation: -REF DB2

Environment

| This command can be issued only from a z/OS console. In a non-data-sharing
| environment, the command must be issued on individual subsystems. In a data
| sharing environment, the command must be issued on individual members of a
| data sharing group. The name of the DB2 subsystem is determined by the
| command prefix. For example, -START indicates that the DB2 subsystem to be
| started is the one with '-' as the command prefix.

The command is rejected if the DB2 subsystem is already active. The ERLY code can be properly updated only when DB2 is not active.

| Applications that use documented DB2 interfaces do not need to be stopped before
| issuing the REFRESH DB2,EARLY command.

Data sharing scope: Member

Authorization

None is required. However, the command can be executed only from a z/OS console with the START command capability. This command is rejected when DB2 is active, so no other authorization is required.

Chapter 62. -RESET GENERICLU (DB2)

The RESET GENERICLU command allows you to purge information stored by VTAM in the coupling facility for one or more partners of a particular DB2 subsystem.

The command must be issued from the DB2 subsystem that has the VTAM affinity to the particular partner LU whose information you are purging.

Abbreviation: -RESET GENERIC

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

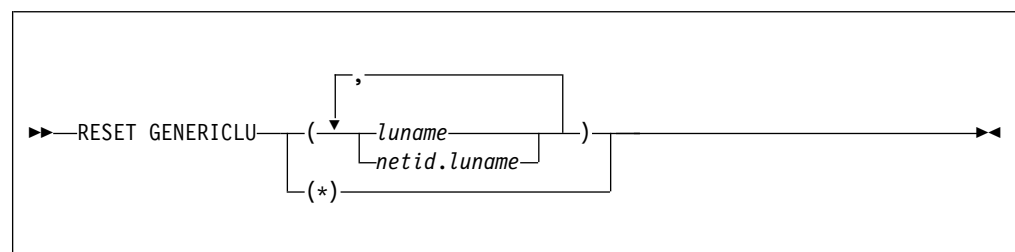
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

(*luname*)

Specifies the real VTAM LU name of the partner whose generic LU name mapping is to be purged. The NETID of this partner LU must be the same as the local DB2 NETID.

(*netid.luname*)

Specifies that the VTAM shared memory information that is associated with the specified NETID and LUNAME is purged.

- (*) Purges the VTAM shared memory information for all partners of this DB2 subsystem. This command option should only be used if you are planning to remove this DB2 subsystem from the DB2 group.

Usage notes

The following conditions must be satisfied for the RESET GENERICLU command to be successful:

- DDF must be started.
- No VTAM sessions can be active to the partner LU that is specified on the command.
- DB2 must not have any indoubt thread resolution information associated with the specified partner LU.

Examples

Example 1: Purge the VTAM generic name mapping that is associated with partner NET1.USER5LU.

```
-DB2A RESET GENERICLU(NET1.USER5LU)
```

Example 2: Purge the VTAM generic name mappings for all LUs that are partners of this DB2 subsystem. Use this version of the command only when removing this DB2 subsystem from the data sharing group.

```
-DB2A RESET GENERICLU(*)
```

Chapter 63. -RESET INDOUBT (DB2)

The DB2 command RESET INDOUBT purges the information that is displayed in the indoubt thread report that is generated by the DISPLAY THREAD command.

When the DISPLAY THREAD TYPE(INDOUBT) command output shows a RESET value of YES, use the RESET INDOUBT command without the FORCE option to purge indoubt thread information for threads in the following situations:

- Threads in which the DB2 database manager has a coordinator role that it cannot fulfill because of participant cold start, sync point protocol errors, or indoubt resolution protocol errors.
- Threads that were indoubt but were resolved with the RECOVER INDOUBT command. Heuristic damage occurred during subsequent resynchronization with the coordinator.

When the DISPLAY THREAD TYPE(INDOUBT) command output shows a RESET value of null, use the RESET INDOUBT command with the FORCE option to purge indoubt thread information in the following situations:

- A DB2 database manager has a coordinator role. No errors occurred that preclude automatic resolution with the participants. Resynchronization with affected participants is not performed.
- A DB2 database manager has a participant role. No errors occurred that preclude automatic resolution with the coordinator. Resynchronization with the coordinator is not performed.

The DISPLAY THREAD TYPE(INDOUBT) command output must indicate a commit or abort decision before an entry can be purged.

Abbreviation: -RESET IND

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

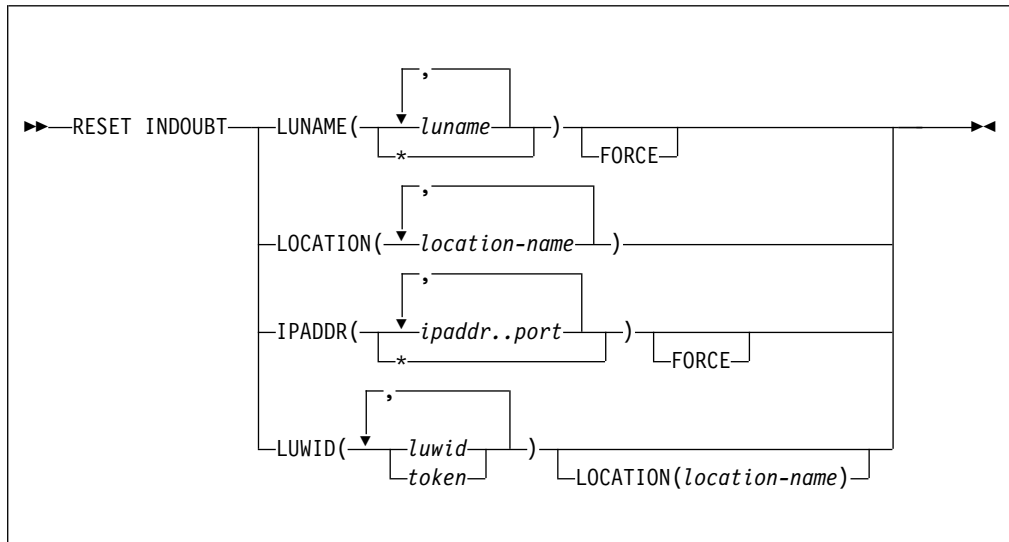
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- RECOVER privilege
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

LUNAME(*luname* , ...)

Purges all qualifying indoubt information that pertains to the specified LUNAME.

luname

Is expressed as a one- to eight-character name. If you use more than one LUNAME, separate each name with a comma.

(*)

Purges indoubt information for all SNA locations.

FORCE

Forces the purging of coordinator and participant indoubt resolution responsibility even when no errors that preclude automatic resolution have been detected. FORCE can be used in conjunction with IPADDR or LUNAME.

Purging resynchronization information when no errors that preclude automatic resynchronization have been detected simulates a cold start. Thus, no connections can exist between DB2 and the named partner when this command is executed. After you run the FORCE option, the next connection with the named partner location will be a cold start connection. If a connection with the named partner exists at the time this command is run, the command fails with message DSNL448I.

FORCE can be used to bypass warm start connectivity problems when errors that are occurring in the recovery log name exchange result in the partner refusing the connection attempt.

LOCATION(*location-name* , ...)

Purges all qualifying indoubt information that pertains to the named location.

location-name is expressed as a 1- to 39-character name, which identifies the partner, whether it is a requester or server. If the partner is not a DB2 for z/OS subsystem, the location name can be expressed as one of the following formats:

- A one- to eight-character *luname*, as defined to VTAM at the server location. This name must be enclosed in the less-than (<) and the greater-than (>) characters to distinguish it from a DB2 location name.

- A dotted decimal or colon hexadecimal IP address.

IPADDR(*ipaddr..port*)

Purges all qualifying indoubt information that pertains to the dotted decimal or colon hexadecimal IP address that is associated with the resync port number.

This keyword can be used in place of the LUNAME keyword when the partner uses TCP/IP instead of SNA.

ipaddr..port

Is the dotted decimal or colon hexadecimal IP address of the remote site, followed by the resync port number. The IP address and port must be delimited by a double period (..). If you use more than one IP address and port number, use commas to separate the items in the list.

(*)

Purges indoubt information for all TCP/IP locations.

LUID

Purges indoubt information for the thread with the specified LUWID.

luwid

Consists of an LU network name, an LUW instance number, and a commit sequence number.

The LU network name consists of a 1- to 8-character network ID, a period, and a 1- to 8-character network LU name. The LUW instance number consists of a period followed by 12 hexadecimal characters. The last element of the LUWID is the commit sequence number, which consists of a period followed by four hexadecimal characters.

token

A token is an alternate way to express an LUWID. DB2 assigns a token to each thread that it creates. It is a one- to six-digit decimal number that appears after the equal sign in all DB2 messages that display an LUWID.

If you enter one- to six-decimal digits, DB2 assumes that you are supplying a token. The token that DB2 assigns to a specific LUWID is unique for that DB2 subsystem, but it is not necessarily unique across all subsystems.

Output

The response from this command includes any of the messages from DSNL440I through DSNL449I.

If you specify RESET INDOUBT incorrectly, you receive message DSNL440I.

Usage notes

Purging participant indoubt information: Use caution when you specify the FORCE option to purge participant indoubt information. Normally, after the use of the RECOVER INDOUBT command, automatic resolution with the coordinator determines if heuristic damage has occurred. This detection is lost if RESET INDOUBT is used before automatic resolution with the coordinator can be achieved.

Purging coordinator indoubt information: Use caution when you specify the FORCE option to purge coordinator indoubt information when no errors are precluding automatic resolution. When the information is purged, any participant

that is indoubt is forced to use a heuristic decision process to resolve the indoubt logical unit of work.

Example

```
| The following command resets an indoubt unit of work by specifying an IP  
| address:  
| RESET INDOUBT IPADDR(::FFFF:10.97.217.50..1332)
```

Chapter 64. RUN (DSN)

The DSN subcommand RUN executes an application program, which can contain SQL statements.

Environment

This subcommand can be issued under the DSN command processor running in either foreground or background mode, or it can be issued by using the DB2I RUN panel.

Data sharing scope: Member

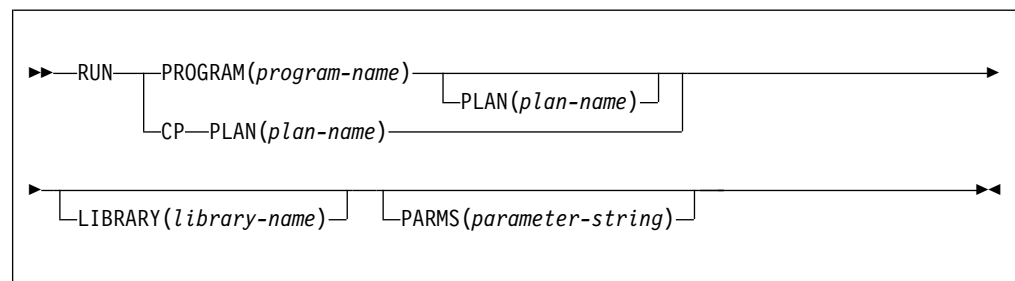
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- EXECUTE privilege on the plan
- Ownership of the plan
- SYSADM authority

To run an application, the plan must be enabled for your local server. Any associated packages from which you execute statements must also be enabled.

Syntax



Option descriptions

Use at least one of the two following clauses, but do not use the same clause twice.

PROGRAM (*program-name*)

Identifies the program that you want to run.

CP

Directs input to the user's command processor, and causes a prompt to be issued: ENTER TSO COMMAND. This is useful for running command processors and debugging programs (for example, COBTEST).

Processing the specified TSO command creates a new task control structure under which the TSO command executes. All application programs that are initiated from this TSO command session also execute under the same task structure, and must establish a new connection to DB2 if they use SQL requests.

When the TSO command completes, the new task structure is terminated, and control is returned to the original DB2 connection and task structure established by the DSN command.

Later TSO commands can be issued directly from the DSN session, or through the RUN subcommand with the CP option.

PLAN(*plan-name*)

Is optional after the PROGRAM option, but required after the CP option.

plan-name is the name of the application plan for the program.

When PROGRAM is used, the **default** plan name is *program-name* .

LIBRARY(*library-name*)

Specifies the name of the data set that contains the program to be run.

If *library-name* is not specified, normal z/OS library searching is used. The data sets that are specified in the STEPLIB DD statements are first searched for the entry point name of the program. If STEPLIB is not present, the data sets that are specified in the JOBLIB DD statements are searched. If the entry point name is not found there, the link list is searched.

Subprograms: Normal z/OS library searching is **always** used for any subprograms that is loaded by the main program. If the subprograms reside in the same library as the main program, the library-name must also be defined for the normal z/OS search pattern (STEPLIB, JOBLIB, link list). If a library that is defined in that way contains both the main program and any loaded subprograms, you do not need to use the LIBRARY option.

PARMS(*parameter-string*)

parameter-string is a list of parameters that are to be passed to your application program. Separate items in the list with commas, blanks, or both, and enclose the list between apostrophes. If the list contains apostrophes, represent each apostrophe by using two consecutive apostrophes. The list is passed as a varying-length character string of 1 to 100 decimal characters.

For Assembler: Use a list of the form 'program parameters'. There are no run time parameters.

No run time or application parameter validation is performed by the RUN subcommand on the *parameter-string* that is passed to your application program. All specified parameter values are assumed to adhere to the parameter syntax and format criteria defined by the language in which the application program is written.

For C: Use a list of the form A/B, where A represents a list of run time options, and B represents a list of parameters for the C application program. If run time options are not needed, write the list in the form /B. If the NOEXECOPS run time option is in effect, omit the “/”.

For COBOL: If Language Environment is not the run time environment, use a list of the form B/A, where B represents a list of parameters for the COBOL application program, and A represents a list of run time options. If program parameters are not needed, write the list in the form of /A.

If Language Environment is the run time environment, use a list of the form A/B, where A represents a list of run time options, and B represents a list of parameters for the COBOL application program. If run time options are not needed, write the list in the form of /B. For compatibility, Language Environment provides the CBLOPTS run time option. When CBLOPT(YES) is specified in CEEDOPT or CEEUOPT and the main routine is COBOL, specify

the list in the form of B/A, the same form as when the run time environment is not Language Environment. CBLOPT(NO) is the default.

For Fortran: Use a list of the form A/B, where A represents a list of Fortran run time options and B represents a list of parameters for the Fortran application program. If Fortran run time options are not needed, write the list in the form of B or /B. The second form must be used if a slash is present within the program arguments. If only Fortran run time options are present, write the list in the form of A/.

For PL/I: Use a list of the form A/B, where A represents a list of run time options, and B represents a list of parameters for the PL/I application program. If run time options are not needed, write the list in the form /B. If the PL/I NOEXECOPS procedure option is specified, omit the "/". An informational system message is issued if you omit the slash, or if the value that is passed to the PL/I run time package is not valid.

Usage note

Multitasking restriction: When running a program that uses a multitasking environment, the first task to issue an SQL statement must issue all subsequent SQL calls. That is, only one task in a multitasking environment can issue SQL calls. This task must be a subtask of, or running at the same TCB level as, the DSN main program.

Examples

Example 1: Run application program DSN8BC4. The application plan has the same name. The program is in library ' *prefix* .RUNLIB.LOAD'.

```
DSN SYSTEM (DSN)
RUN PROGRAM (DSN8BC4) LIB ('prefix.RUNLIB.LOAD')
```

Example 2: Run application program DSN8BP4. The application plan is DSN8BE81. The program is in library ' *prefix* .RUNLIB.LOAD'. Pass the parameter O'TOOLE to the PL/I application program with no PL/I run time options.

```
DSN SYSTEM (DSN)
RUN PROGRAM (DSN8BP4) PLAN (DSN8BE81) -
LIB ('prefix.RUNLIB.LOAD') PARM ('/O'TOOLE')
```

Chapter 65. -SET ARCHIVE (DB2)

The DB2 command SET ARCHIVE sets the maximum number of tape units for the archive log. It also sets the maximum deallocation time of tape units for the archive log.

This command overrides the values that are specified during installation or in a previous invocation of the SET ARCHIVE command. The changes that SET ARCHIVE makes are temporary; at restart, DB2 again uses the values that are set during installation.

Abbreviation: -SET ARC

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

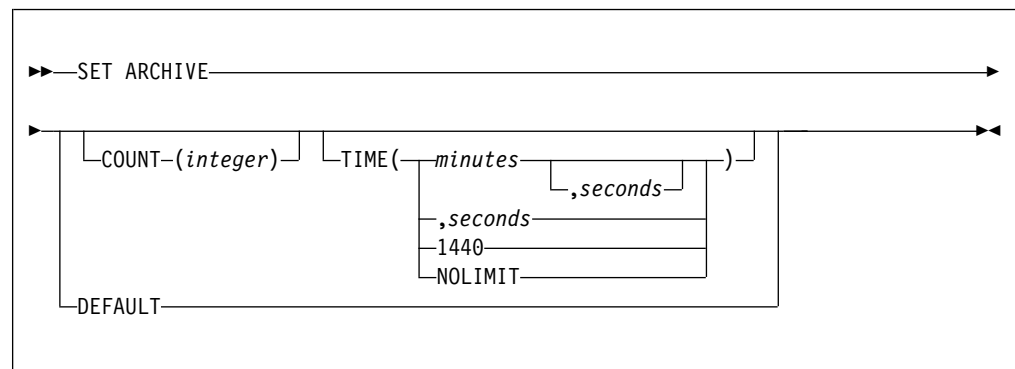
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- ARCHIVE privilege
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

The following options override the READ TAPE UNITS(COUNT) and DEALLC PERIOD TIME subsystem parameters that are specified during installation.

COUNT(*integer*)

Specifies the maximum number of tape units that can be dedicated to reading archive logs. This value affects the number of concurrent reads that are allowed for unique archive data sets that reside on tapes.

integer can range from 1 to 99.

- If the number that you specify is greater than the current specification, the maximum number of tape units allowable for reading archive logs increases.
- If the number that you specify is less than the current specification, tape units that are not being used are immediately deallocated to adjust to the new COUNT value. Active (or premounted) tape units remain allocated; only tape units that are inactive are candidates for deallocation because of a lowered COUNT value.

TIME

Specifies the length of time during which an allocated archive read tape unit is allowed to remain unused before it is deallocated.

(*minutes*)

Specifies the maximum number of minutes.

minutes must be an integer between 0 and 1439.

(*seconds*)

Specifies the maximum number of seconds.

seconds must be an integer between 1 and 59.

(NOLIMIT) or (1440)

Indicates that the tape unit will never be deallocated. Specifying TIME(1440) is equivalent to TIME(NOLIMIT). The seconds specification is not allowed when you specify that TIME is 1440.

DEFAULT

Resets the COUNT and TIME parameters back to the values that were specified during DB2 installation.

Usage notes

Archive tape reading performance: To achieve the best performance for reading archive tapes, specify the maximum values that are allowed (within system constraints) for both the COUNT and TIME options.

IEF238D "REPLY DEVICE NAME OR CANCEL" message: Replying "CANCEL" to this message resets the COUNT value to the current number of tape units. For example, if the current COUNT value is 10, but you reply "CANCEL" to the request for the seventh tape unit, the COUNT value is reset to 6.

Delaying tape deallocation in a data sharing environment: When you submit a recover job on a member of a data sharing group that requires a tape unit that must remain unused for a certain length of time before it is deallocated, the archive tape is not available to any other member of the group until the tape is deallocated. Unless all recover jobs will be submitted from the same member, you might not want to use the COUNT option and ensure that field DEALLOC PERIOD on installation panel DSNTIPA has a value of 0.

Output

The response from this command includes any of the messages from DSNJ334I through DSNJ337I.

Examples

Example 1: Allocate two tape units that can remain unused for 30 seconds before they are deallocated.

```
-SET ARCHIVE COUNT(2) TIME(,30)
```

Example 2: Allocate four tape units that can remain unused for 2 minutes before they are deallocated.

```
-SET ARCHIVE COUNT(4) TIME(2)
```

Example 3: Allocate one tape unit that is never deallocated.

```
-SET ARCHIVE COUNT(1) TIME(1440)
```


Chapter 66. -SET LOG (DB2)

The DB2 command SET LOG modifies the checkpoint frequency that is specified during installation. This command also overrides the value that was specified in a previous invocation of the SET LOG command.

The changes that SET LOG makes are temporary; at restart, DB2 uses the values that were used for restart. The LOGLOAD value takes effect following the next system checkpoint. You can use SET LOG to suspend or resume logging and update activity for the current DB2 subsystem. You can also use the NEW LOG option of SET LOG to add an active log to the configuration. Changes made by NEW LOG are pervasive.

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program that uses the instrumentation facility interface (IFI).

Data sharing scope: Member

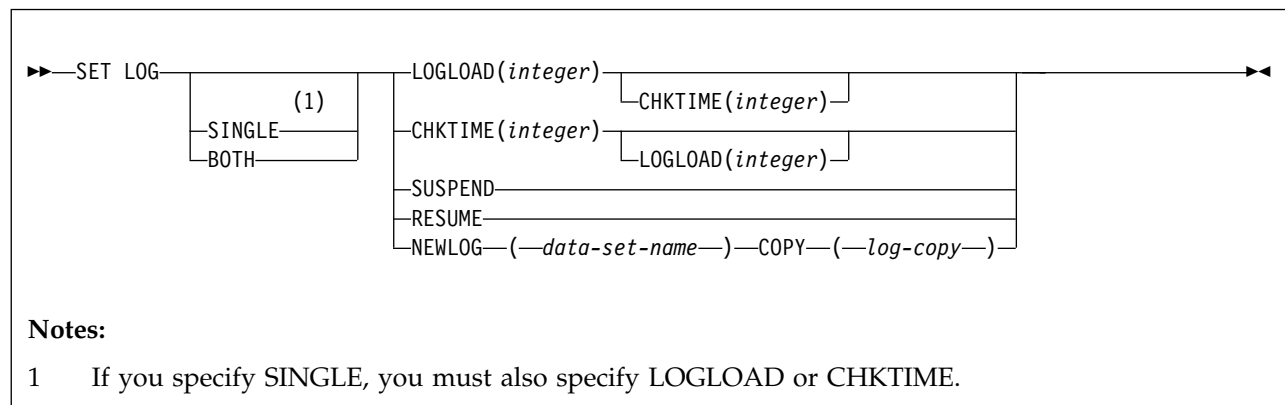
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- ARCHIVE privilege
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Notes:

- 1 If you specify SINGLE, you must also specify LOGLOAD or CHKTIME.

Option descriptions

The following option overrides the LOGLOAD subsystem parameter that is specified in the CHECKPOINT FREQ field on installation panel DSNTIPN.

SINGLE

Specifies that only a single option, either LOGLOAD or CHKTIME, is used to control checkpoint frequency. If you specify SINGLE, you must specify LOGLOAD or CHKTIME.

SINGLE is optional. If you do not use this option, the existing mode, SINGLE or BOTH, is used. If you specify SINGLE but BOTH was previously in effect, the mode changes to SINGLE.

BOTH

Specifies that both LOGLOAD and CHKTIME are used control checkpoint frequency. The threshold that is reached first triggers a system checkpoint and resets both thresholds.

BOTH is optional. If you do not use this option, the existing mode, SINGLE or BOTH, is used. If you specify BOTH but SINGLE was previously in effect, the mode changes to BOTH and the input values for LOGLOAD and CHKTIME are used. If you do not specify LOGLOAD or CHKTIME, the existing value for the option not specified remains in effect. If the value for CHKTIME or LOGLOAD has not been set and the option is not specified in the SET LOG command, the default value for the parameter that is not specified is used.

The default value for CHKTIME is 5 minutes. The default value for LOGLOAD is 500000 log records.

LOGLOAD(*integer*)

Specifies the number of log records that DB2 writes between the start of successive checkpoints. You can specify a value of 0 to initiate a system checkpoint without modifying the current LOGLOAD value.

Possible values of *integer* are:

- 1000 to 16000000, if SINGLE is explicitly specified in the SET LOG command
- 0, or 1000 to 16000000, if SINGLE mode is in effect
- 0, or 1000 to 99999999, if BOTH is explicitly specified in the SET LOG command, or BOTH mode is in effect

If DB2 was previously running in SINGLE mode, CHKTIME was previously controlling checkpoints, and you specify a value greater than 0 for LOGLOAD, LOGLOAD controls future checkpoints, and CHKTIME is not used.

CHKTIME(*integer*)

Specifies the number of minutes between the start of successive checkpoints.

integer is any integer from 0 to 1439. Specifying 0 starts a system checkpoint immediately without modifying the checkpoint frequency.

Possible values of *integer* are:

- 1 to 1439, if SINGLE is explicitly specified in the SET LOG command
- 0 to 1439, if SINGLE mode is in effect
- 0 to 1439, if BOTH is explicitly specified in the SET LOG command, or BOTH mode is in effect

If DB2 was previously running in SINGLE mode, LOGLOAD was previously controlling checkpoints, and you specify a value greater than 0 for CHKTIME, CHKTIME controls future checkpoints, and LOGLOAD is not used.

SUSPEND

Specify to suspend logging and update activity for the current DB2 subsystem until SET LOG RESUME is issued. DB2 externalizes unwritten log buffers, takes a system checkpoint (in non-data-sharing environments), updates the BSDS with the high-written RBA, and then suspends the update activity. Message DSNJ372I is issued and remains on the console until update activity resumes.

SUSPEND quiesces the writes for 32-KB pages and the data set extensions for all page sizes. If a 32-KB page write is in progress when you take volume-level copies of your data, SUSPEND prevents an inconsistent copy of a 32-KB page when the copy of your data is restored. If a data set extension is in progress, SUSPEND prevents inconsistencies between the VSAM catalog and the DB2 data set when the copy of your data is restored.

This option is not allowed when the ARCHIVE LOG or STOP DB2 commands activate a system quiesce. Update activity remains suspended until SET LOG RESUME or STOP DB2 is issued. (Also, when logging is suspended, do not issue the ARCHIVE LOG command without also specifying CANCEL OFFLOAD.)

Recommendation: Do not keep log activity suspended during periods of high activity or for long periods of time. Suspending update activity can cause timing-related events such as lock timeouts or DB2 and IRLM diagnostic dumps.

RESUME

Specify to resume logging and update activity for the current DB2 subsystem and to remove the message DSNJ372I from the console. Resumes 32-KB page writes and data set extensions for pages of all sizes.

Recommendation: Issue this command from a z/OS console or from the installation SYSADM ID to avoid possible contention during command authorization checking. When logging is suspended by the SET LOG SUSPEND command, the contention that is generated by holding the log-write latch can cause command-authorization checking to hang until logging resumes.

NEWLOG(*data set name*)

Adds a newly defined active log data set to the active log inventory. If DB2 can open the newly defined data set, the log is added to the active log inventory in the BSDS data sets and is immediately available for use without recycling DB2.

Before you issue this command, you must define the data set with IDCAMS.

Recommendation: Format the new active log data set with the DSNJLOGF utility before you issue the SET LOG command to add the data set to the active log inventory.

COPY(*log copy*)

Specifies the log copy number for the new active log data set.

The value of *log copy* can be 1 or 2. Specify 1 for copy 1 of the active log data set or 2 for copy 2 of the active log data set.

Recommendation: If DB2 is in dual logging mode, add log data sets for both copy 1 and copy 2 of the new active log data set.

Usage notes

How LOGLOAD and CHKTIME values affect DB2 performance: LOGLOAD and CHKTIME values can affect the amount of time needed to restart DB2 after abnormal termination. A large value for either option can result in lengthy restart times. A low value can result in DB2 taking excessive checkpoints. However, when you specify LOGLOAD(0) or CHKTIME(0), the checkpoint request is synchronous when issued from a batch job, and it is asynchronous when issued from a z/OS or TSO console.

The behavior of LOGLOAD(0) and CHKTIME(0) is different in a data sharing environment. Avoid issuing SET LOG LOGLOAD(0) or SET LOG CHKTIME(0) in the data sharing environment if logging has been suspended with SET LOG SUSPEND on any member in the group. If you specify LOGLOAD(0) or CHKTIME(0), the synchronous checkpoint might be suspended until all logging has been resumed when you issue the SET LOG RESUME command.

Use the DISPLAY LOG command to display the current checkpoint parameters. You can see if CHKTIME, LOGLOAD, or both are being used to schedule checkpoints.

The value that you specify for LOGLOAD or CHKTIME is reset to the value specified in the subsystem parameter when DB2 is restarted. If you load a different value by issuing the command SET SYSPARM, the new value is used.

When to suspend logging: Specify SET LOG SUSPEND before making a remote copy of the entire database and logs for a system-level, point-in-time recovery or disaster recovery. You can make remote copies with peer-to-peer remote recovery (PPRC) and FlashCopy®. Suspending logging to make a remote copy of the database lets you avoid quiescing update activity. Read-only activity continues while logging is suspended.

The backup that is made between the SET LOG SUSPEND and the SET LOG RESUME window might contain uncommitted data. If you must restore the entire DB2 subsystem to the time when the log was suspended, restore the entire database and logs from the backup, and then restart DB2 to recover the entire DB2 subsystem to a consistent state.

Avoiding deadlocks when using SET LOG SUSPEND in a data sharing environment: To avoid deadlock in a data sharing environment, issue the SET LOG SUSPEND command on one data sharing member first, wait until it completes, and then issue the command for the remaining members.

Order in which newly defined active log data sets are used: Currently, when data sets are added to the active log inventory with SET LOG NEWLOG, the DB2 database manager uses those data sets in the reverse order from the order in which they are added. This behavior might change in the future, so schemes for adding and switching active logs should not depend on this order.

For example, suppose that the BSDS lists active log data sets DS01, DS02, and DS03. You issue the SET LOG NEWLOG command to add data set DS04 to the active log inventory. Then you issue SET LOG NEWLOG again, to add data set DS05 to the active log inventory. The data sets are listed in the BSDS in the order DS01, DS02, DS03, DS04, DS05. When the current active log data set is DS03, and the database manager switches to a new active log data set, it switches to data set DS05.

This process differs from the order that the database manager uses when data sets are added to the active log inventory through the DSNJU003 utility (change log inventory utility). After the NEWLOG control statement is used to add new data sets, the database manager uses active log data sets in the order in which they are added.

Suppose that the active log inventory contains DS01, DS02, and DS03. You run DSNJU003 with two NEWLOG control statements. The first control statement adds data set DS04, and the second control statement adds data set DS05 to the active log inventory. After DSNJU003 runs, the data sets are listed in the BSDS in the order DS01, DS02, DS03, DS04, DS05. When the current active log data set is DS03, and the database manager switches to a new active log data set, it switches to data set DS04.

Examples

Example 1: Initiate a system checkpoint without modifying the current LOGLOAD value.

```
-SET LOG LOGLOAD(0)
```

Example 2: Modify the system checkpoint interval to every 150000 log records.

```
-SET LOG LOGLOAD(150000)
```

Example 3: Suspend logging activity.

```
-SET LOG SUSPEND
```

Example 4: Resume logging activity.

```
-SET LOG RESUME
```

Example 5: Change checkpoint scheduling to use both log records and time.

```
-SET LOG BOTH CHKTIME(10) LOGLOAD(500000)
```

Example 6: Change checkpoint scheduling to use only log records.

```
-SET LOG SINGLE LOGLOAD(500000)
```

Example 7: Change checkpoint scheduling to use both log records and time by using the existing value for the parameter that was controlling checkpoints and the default value for the other.

```
-SET LOG BOTH
```

Example 8: Add copies of a new active log data set to the active log inventory (DB2 is in dual logging mode).

```
//JOB LIB DD DSN=prefix.SDSNLOAD,DISP=SHR
//NEWLOG EXEC PGM=IKJEFT01,DYNAMNBR=20
//DSNTRACE DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYS SPRT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(DSN1)
-SET LOG NEWLOG(DSN1.LOGCOPY1.DS04) COPY(1)
-SET LOG NEWLOG(DSN1.LOGCOPY1.DS05) COPY(1)
-SET LOG NEWLOG(DSN1.LOGCOPY1.DS06) COPY(1)
-SET LOG NEWLOG(DSN1.LOGCOPY1.DS07) COPY(1)
-SET LOG NEWLOG(DSN1.LOGCOPY2.DS04) COPY(2)
-SET LOG NEWLOG(DSN1.LOGCOPY2.DS05) COPY(2)
-SET LOG NEWLOG(DSN1.LOGCOPY2.DS06) COPY(2)
```

```
| -SET LOG NEWLOG(DSNCAT.LOGCOPY2.DS07) COPY(2)  
| END  
| /*  
| //SYSIN DD *  
| /*
```

The following messages are issued:

```
| DSNJ363I ) DSNJW106 COPY1 LOG DATA SET  
| DSNCAT.LOGCOPY1.DS04 ADDED TO THE ACTIVE LOG INVENTORY  
| DSNJ363I ) DSNJW106 COPY1 LOG DATA SET  
| DSNCAT.LOGCOPY1.DS05 ADDED TO THE ACTIVE LOG INVENTORY  
| DSNJ363I ) DSNJW106 COPY1 LOG DATA SET  
| DSNCAT.LOGCOPY1.DS06 ADDED TO THE ACTIVE LOG INVENTORY  
| DSNJ363I ) DSNJW106 COPY1 LOG DATA SET  
| DSNCAT.LOGCOPY1.DS07 ADDED TO THE ACTIVE LOG INVENTORY  
| DSNJ363I ) DSNJW106 COPY2 LOG DATA SET  
| DSNCAT.LOGCOPY2.DS04 ADDED TO THE ACTIVE LOG INVENTORY  
| DSNJ363I ) DSNJW106 COPY2 LOG DATA SET  
| DSNCAT.LOGCOPY2.DS05 ADDED TO THE ACTIVE LOG INVENTORY  
| DSNJ363I ) DSNJW106 COPY2 LOG DATA SET  
| DSNCAT.LOGCOPY2.DS06 ADDED TO THE ACTIVE LOG INVENTORY  
| DSNJ363I ) DSNJW106 COPY2 LOG DATA SET  
| DSNCAT.LOGCOPY2.DS07 ADDED TO THE ACTIVE LOG INVENTORY
```

Chapter 67. -SET SYSPARM (DB2)

You can use the SET SYSPARM command to change subsystem parameters that can be updated online, while DB2 is started.

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program that uses the instrumentation facility interface (IFI).

Data sharing scope: Member

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

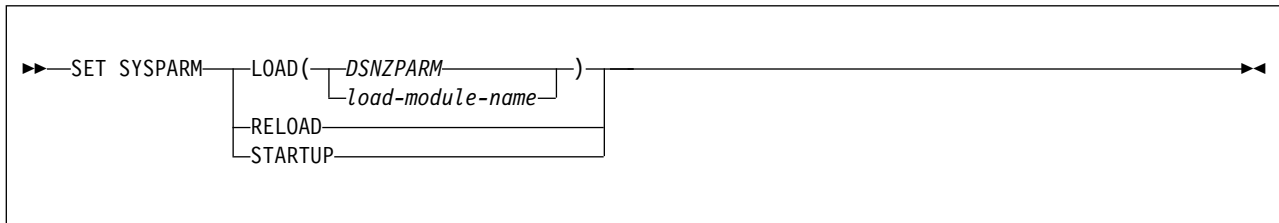
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority
- SECADM authority

To change the following subsystem parameters, you must use a privilege set of the process that includes installation SYSADM authority or SECADM authority:

- AUTHCACH
- BINDNV
- DBACRVW
- EXTSEC
- REVOKE_DEP_PRIVILEGES
- SECADM1
- SECADM1_TYPE
- SECADM2
- SECADM2_TYPE
- SEPARATE_SECURITY
- SYSADM
- SYSADM2
- SYSOPR1
- SYSOPR2
- TCPALVER

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

LOAD(*load-module-name*)

Specifies the name of the load module to load into storage. The default load module is **DSNZPARM**.

RELOAD

Reloads the last named subsystem parameter load module into storage.

STARTUP

Resets loaded parameters to their startup values.

Usage notes

To update the subsystem parameters on a subsystem, follow these steps:

1. Run through the installation process in Update mode.
2. Produce a new subsystem parameter load module.
3. Issue the SET SYSPARM command.

Important:

- Do not issue SET SYSPARM while you are compressing the data set that contains the new subsystem parameter load module.
- If you are compressing the data set that contains the new subsystem parameter load module, and the data set resides in the system LLA, after you compress the data set, and before you issue SET SYSPARM, issue command F LLA, REFRESH. Wait for message CSV210I, which indicates that F LLA, REFRESH is complete.

If you attempt to change installation SYSADM or installation SYSOPR subsystem parameters and you do not have the proper authority, the parameter values that are in place prior to the load of the new subsystem-parameter module are used instead of the unauthorized values in the new module. DB2 issues message DSNZ015 for each attempt of an unauthorized change to a subsystem parameter.

Examples

Example 1: Change from DSNZPARM to ADMPARM1.

```
-SET SYSPARM LOAD(ADMPARM1)
```


Example 2: Reload ADMPARM1 if it is the currently running load module.

```
-SET SYSPARM RELOAD
```


Example 3: Reload the subsystem parameters that the DB2 subsystem loaded at startup.

```
-SET SYSPARM STARTUP
```



Related concepts:

 [Subsystem parameters \(Introduction to DB2 for z/OS\)](#)

Related tasks:

 [Updating subsystem parameter and application default values \(DB2 Installation and Migration\)](#)

Related reference:

 [Directory of subsystem parameters, panel fields, and application default values \(DB2 Installation and Migration\)](#)

Chapter 68. SPUFI (DSN)

The DSN subcommand SPUFI executes the SQL processor using file input.

Environment

You can use this subcommand only under ISPF. You can issue it from ISPF option 6, or from a CLIST.

Data sharing scope: Member

Authorization

None is required.

Syntax

```
▶▶—SPUFI—▶▶
```

Usage notes

SPUFI session: The SPUFI subcommand runs SPUFI and presents the SPUFI panel as the start of a SPUFI session.

In the SPUFI session, you can access the CURRENT SPUFI DEFAULTS panel. You can change DB2I defaults by splitting the screen and accessing the DB2I DEFAULTS panel, or by changing the defaults before starting the SPUFI session.

SPUFI panel variables: The SPUFI panel variables you enter after invoking SPUFI directly with the DSN command are not saved in the same place. Panel variables, therefore, vary depending on whether you execute the facility directly, or through DB2I.

Chapter 69. /SSR (IMS)

The IMS /SSR command allows the IMS operator to enter an external subsystem command.

Environment

This command can be issued only from an IMS terminal.

Data sharing scope: Member

Authorization

This command requires an appropriate level of IMS authority.

In addition, the set of privileges held by the primary authorization ID or any of the secondary authorization IDs must include the authority to enter the DB2 command that follows /SSR.

Syntax



```
>>—/SSR—subsystem-command—<<
```

Option description

subsystem-command

Specifies a valid subsystem command. The first character following /SSR must be the subsystem recognition character of the subsystem to which the command is to be directed (in this case DB2). The IMS subsystem recognition character is defined in the IMS SSM member for the external subsystem.

Usage note

Routing the command: IMS uses the command recognition character (CRC) to determine which external subsystem, in this case DB2, receives the command. The only action taken by IMS is to route the command to the appropriate subsystem.

Chapter 70. -START ACCEL (DB2)

The DB2 command START ACCEL notifies the DB2 subsystem that it should use the specified accelerator servers.

Abbreviation: -STA ACCEL

On successful completion of the command, queries for the specified accelerator servers can begin to run. DB2 resets trace statistics to 0 each time that you execute the START ACCEL command.

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program that uses the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the SCOPE option.

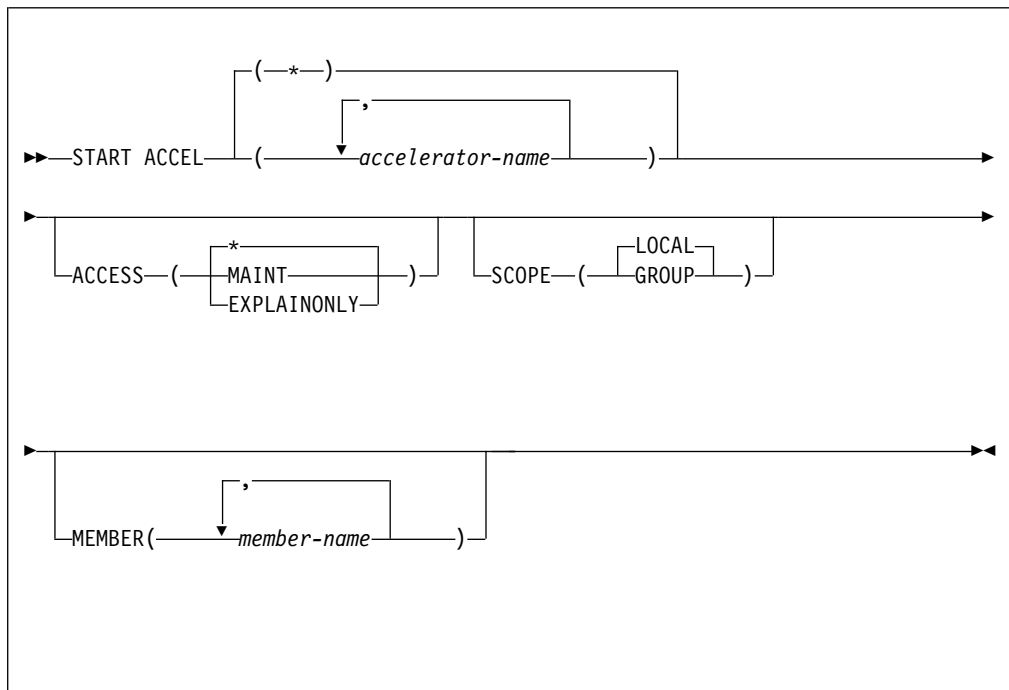
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization by using primary and secondary authorization IDs.

Syntax



Option descriptions

(*accelerator-name*)

The accelerator server name.

(*)

Indicates that the start command applies to all accelerator servers.

ACCESS

Specifies whether access to accelerator servers is general or restricted. Valid values are:

(*) Allows general access; all authorized users can use the accelerator server.

(MAINT)

Allows only installation SYSADM and installation SYSOPR access to the accelerator server.

(EXPLAINONLY)

Prohibits use of the accelerator server except for SQL EXPLAIN execution. No queries will be executed by this server.

SCOPE

Specifies the scope of the command. In a non-data-sharing environment, the option is ignored. Valid values are:

(LOCAL)

Starts the accelerator server for the current member.

(GROUP)

Starts accelerator servers for the entire data sharing group.

MEMBER

Restricts the start of the identified accelerator server to specific members of the

data sharing group. The default is to start accelerator server on the local member. In a non-data-sharing environment, the option is ignored.

Usage notes

| You must issue START ACCEL when you add a new accelerator server to the
| SYSACCELERATORS pseudo-catalog table, or when you have modified the
| information in this table and issued the -STOP ACCEL command.

If both SCOPE(GLOBAL) and MEMBER(*member-name*) are specified, the command will be executed only on the specified member or members.

You can monitor the status of a started accelerator server by running the -DISPLAY ACCEL command.

Example

The following command starts all accelerator servers.

```
-START ACCEL
```


Sample results:

```
DSNX811I ) DSNX8STA START ACCELERATOR SUCCESSFUL FOR BLINK1  
DSNX811I ) DSNX8STA START ACCELERATOR SUCCESSFUL FOR BLINK2  
DSNX811I ) DSNX8STA START ACCELERATOR SUCCESSFUL FOR BLINK3  
DSNX819I ) DSNX8STA ALL ACCELERATORS STARTED  
DSN9022I ) DSNX8CMD '-START ACCEL' NORMAL COMPLETION
```

Related tasks:

 [Enabling DB2 for IBM DB2 Analytics Accelerator for z/OS \(DB2 Performance\)](#)

Related reference:

 [Support Portal: IBM DB2 Analytics Accelerator for z/OS](#)

Chapter 71. START admtproc

The START *admtproc* command starts the scheduler that is specified in the *admtproc* parameter.

This command can be started at the operator's console, or during DB2 startup or initialization. Once started, the administrative task scheduler is always up, unless it is stopped by a STOP command at the operator's console.

Each DB2 subsystem has a coordinated administrative task scheduler address space for starting a z/OS started task procedure, therefore if there are many DB2 subsystems running on one z/OS, there is a separate administrative task scheduler with a separate name for each. Two instances of the same administrative task scheduler cannot run simultaneously. To avoid starting up a duplicate administrative task scheduler, at startup the administrative task scheduler checks all of the address spaces for duplicate names. If another address space with the same name is already running, the administrative task scheduler that is starting up will immediately shut down with a console error message. The administrative task scheduler can only check the address spaces in the same system, but not the entire Sysplex.

Environment

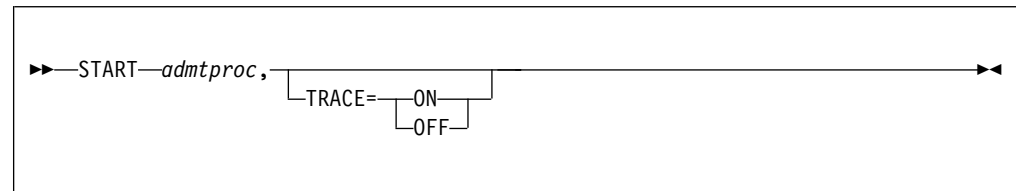
This command can be issued only from a z/OS console.

Data sharing scope: Member

Authorization

The command requires an appropriate level of operating system authority.

Syntax



Options must be separated by commas, with no spaces.

Option descriptions

admtproc

Specifies the procedure name of the administrative task scheduler task that you want to start.

trace

Traces can be turned on or off using this option.

Examples

Example: This command starts the *admtproc* scheduler.

Enter the following command on the system console:

```
start admtproc
```

Chapter 72. /START (IMS)

The IMS /START command (with the SUBSYS parameter) makes the connection between IMS and the specified external subsystem available. Establishing the connection allows application programs to access resources managed by the external subsystem.

The following information is only a partial description of the /START command.

Environment

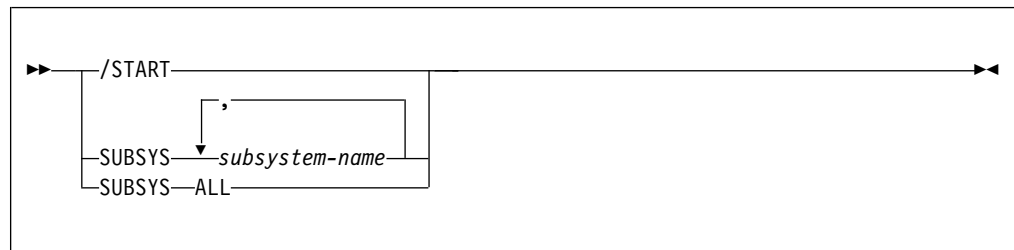
This command can be issued only from an IMS terminal.

Data sharing scope: Member

Authorization

This command requires an appropriate level of IMS authority.

Syntax



Option descriptions

SUBSYS

Specifies one or more names of external subsystems to be connected to IMS, or all external subsystems.

subsystem-name , ...

Identifies one or more names of external subsystems to be connected to IMS.

ALL

Indicates that all external subsystems are to be connected to IMS.

Usage note

Inactive entries: The copy in main storage of the external subsystem PROCLIB entry is refreshed as part of /START command function when that entry is not active (that is, when the connection does not exist). This allows the installation to stop the subsystem connection, change the specifications in the PROCLIB entry, and restart the subsystem connection without bringing down IMS.

Chapter 73. -START DATABASE (DB2)

The **START DATABASE** command makes the specified database available for use.

Depending on which options you specify, the following objects can be made available for read-only processing, read-write processing, or utility-only processing:

- Databases
- Table spaces
- Index spaces
- Physical partitions of partitioned table spaces or index spaces (including index spaces housing data-partitioned secondary indexes (DPSIs))
- Logical partitions of nonpartitioned secondary indexes.

The command is typically used after one of the following events:

- The STOP DATABASE command is issued
- A table space, partition, or index is placed in group buffer pool RECOVER-pending status (GRECP)
- Pages have been put on the logical page list (LPL) for a table space, partition, or index

In a data sharing environment, the command can be issued from any DB2 subsystem in the group that has access to the specified database.

Abbreviation: -STA DB

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- STARTDB privilege
- DBMAINT authority
- DBCTRL authority
- DBADM authority
- System DBADM authority
- SYSCTRL authority
- SYSADM authority

When you are using a privilege set that does not contain the STARTDB privilege for a specified database, DB2 issues an error message.

For implicitly created databases, the database privilege or authority can be held on the implicitly created database or on DSNDB04. If the START DATABASE

command is issued on specific table spaces or index spaces in an implicitly created database, ownership of the table spaces is sufficient to start them. This means that the owner can display information about an implicitly created table space or index space if the command explicitly specifies that table space or index space name.

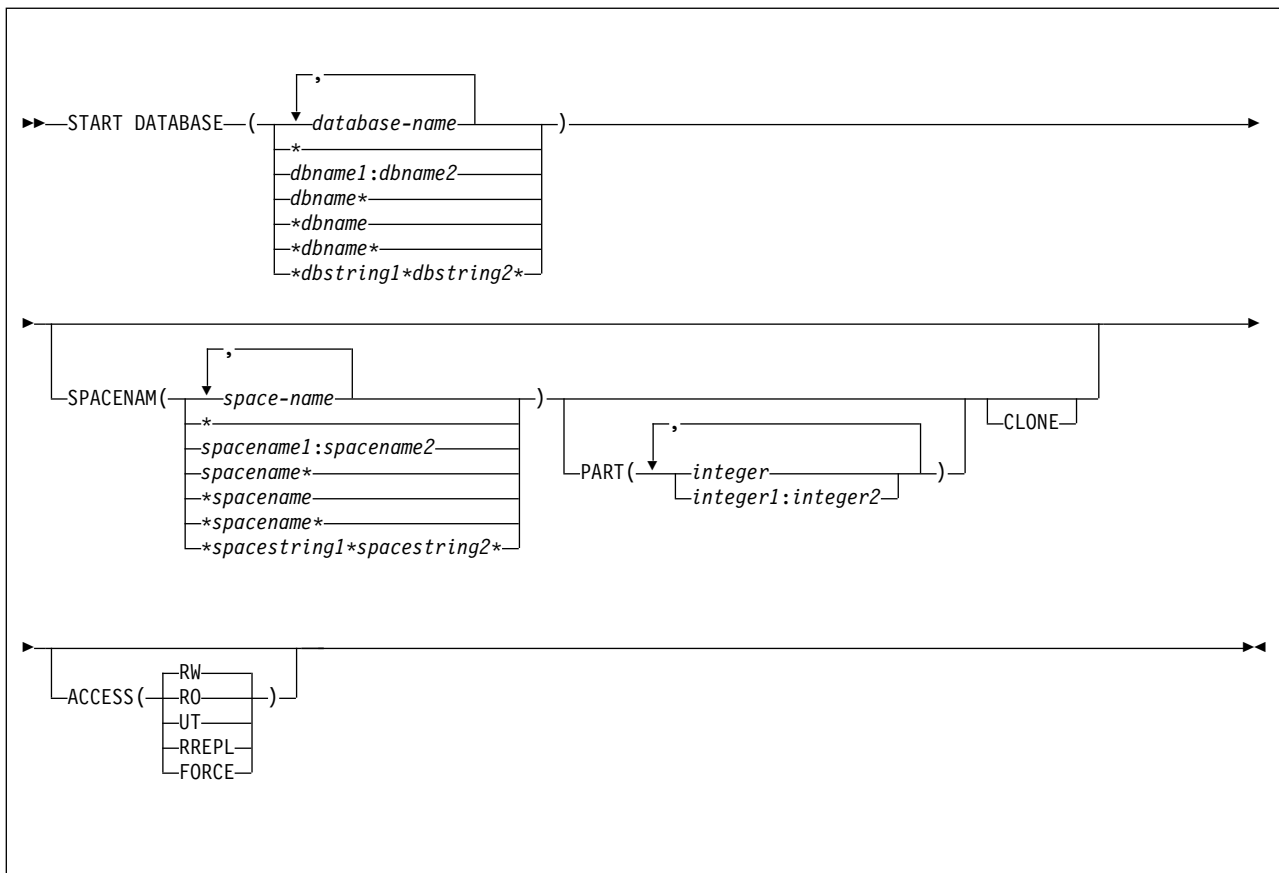
All specified databases with the STARTDB privilege included in the privilege set of the process are started.

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

When data definition control is active, installation SYSOPR or installation SYSADM authority is required to start a database, a table space, or an index space containing a registration table or index.

Table space DBD01 in database DSNDB01 and table spaces and index spaces in database DSNDB06 are required to check the authorization for using the START DATABASE command. If a table space or index space required for this authorization check is stopped, or is unavailable because it is in LPL or GRECP status, installation SYSADM authority is required to start any database, table space, or index space, including the ones required for the authorization check.

Syntax



Option descriptions

(*database-name* , ...)

Specifies the name of a database, or a database for the table spaces or index spaces that are to be started. If you use more than one name, separate names in the list with commas.

(*)

Starts all databases for which the privilege set of the process has at least DBMAINT authority or STARTDB privilege (except databases that are already started). You cannot use (*) with ACCESS(FORCE).

You can start DSNDB01, DSNDB06, and work file databases, such as DSNDB07, only by explicitly specifying them (for example, START DATABASE(DSNDB01)).

dbname and *dbstring* can have any of the forms in the following list (where *dbname1* and *dbname2* represent any 1- to 8-character string, and *dbname* represents any 1- to 7-character string):

Form Starts

dbname1:dbname2

All databases whose names, in Unicode, are greater than or equal to *dbname1* and less than or equal to *dbname2*

dbname*

All databases whose names begin with the string *dbname*

***dbname**

All databases whose names end with the string *dbname*

dbname

All databases whose names contain the string *dbname*

dbstring1*dbstring2

All databases whose names contain the strings *dbstring1* and *dbstring2*

SPACENAM

Specifies the particular table spaces or indexes within the database that are to be started. If you use ACCESS(FORCE), you must use SPACENAM with a list of table space and index names.

Abbreviation: SPACE, SP

(*space-name* , ...)

Specifies the name of a table space or index space that is to be started. You can use a list of several names of table spaces and index spaces. Separate names in the list with commas.

You can specify *space-name* like *database-name* to designate:

- The name of a single table space or index space
- A range of names
- A partial name, including a beginning or ending pattern-matching character (*)
- Two strings separated by a pattern-matching character (*)
- Any combination of the previous items in this list, with the following exceptions. Consecutive pattern-matching characters (*) are not allowed, and you cannot specify two pattern-matching characters (*) in the middle of a keyword string.

You cannot use a partial name or a range of names with the ACCESS(FORCE) option.

(*)

Starts all table spaces and index spaces in the specified database. You cannot use (*) with ACCESS(FORCE).

spacename and *spacestring* can have any of the forms in the following list (where *spacename1* and *spacename2* represent any 1- to 8-character string, and *spacename* represents any 1- to 7-character string):

Form Displays the status of

spacename1:spacename2

All table spaces or index spaces whose names, in Unicode, are greater than or equal to *spacename1* and less than or equal to *spacename2*

spacename*

All table spaces or index spaces whose names begin with the string *spacename*

****spacename***

All table spaces or index spaces whose names end with the string *spacename*

****spacename****

All table spaces or index spaces whose names contain the string *spacename*

****spacestring1*spacestring2****

All table spaces or index spaces whose names contain the strings *spacestring1* and *spacestring2*

PART (*integer* , ...)

Specifies the partition number of one or more partitions, within the specified table space or index, that are to be started. The start or stop state of other partitions does not change.

The specified *integer* must identify a valid partition number for the corresponding space name and database name. If you specify nonvalid partition numbers, you receive an error message for each nonvalid number, but all other valid partitions that you specified are started.

integer can be written to designate one of the following specifications:

- A list of one or more partitions
- A range of all partition numbers that are greater than or equal to *integer1* and less than or equal to *integer2*
- A combination of lists and ranges

The PART option is valid with partitioned table spaces, partitioned indexes, and nonpartitioned type 2 indexes of partitioned table spaces. If you specify PART with a nonpartitioned table space or index on a nonpartitioned table space, you receive an error message, and the nonpartitioned space is not started.

CLONE

Starts clone objects. In the absence of the CLONE keyword, base table objects are started and the clone table objects are not processed. If you specify the CLONE keyword then only clone objects are processed.

ACCESS

Specifies whether the objects that are started are in read/write, read only, or utility only status. Also forces access to objects that are in unavailable status.

Abbreviation: ACC

(RW)

Allows programs to read from and write to the specified databases, table spaces, indexes, or partitions.

(RO)

Allows programs to only read from the specified databases, table spaces, indexes, or partitions. Any programs attempting to write to the specified objects will not succeed. Do not use this option for a database for declared temporary tables (databases created with the AS TEMP option).

(UT)

Allows only DB2 online utilities and the SQL DROP statement to access the specified databases, table spaces, indexes, or partitions.

(RREPL)

Allows programs only read access to the specified databases, table spaces, indexes, or partitions, unless those programs were identified as replication programs. Attempts by non-replication programs to write to the specified objects fail. Most DB2 utilities are allowed on the specified objects.

If ACCESS(RREPL) is specified in DB2 10 conversion mode or DB2 10 enabling-new-function mode, DB2 converts ACCESS(RREPL) to ACCESS(RO), and does not issue a message.

Restriction: Do not use ACCESS(RREPL) for a database for declared temporary tables (a database that was created with the AS TEMP option).

(FORCE)

Resets any indications that a table space, index, or partition is unavailable because of pages in the logical page list, pending-deferred restarts, write-error ranges, read-only accesses, or utility controls. FORCE also resets the CHECK-pending, COPY-pending, and RECOVER-pending states. Full access to the data is forced. FORCE cannot be used to reset the restart-pending (RESTP) state.

When using ACCESS(FORCE), you must use a single database name, the SPACENAM option, and an explicit list of table space and index names. You cannot use any range or combination of pattern-matching characters (*), including DATABASE (*) or SPACENAM (*).

A utility-restrictive state is reset (and the utility is terminated) only if all of the target objects are reset with this command. To identify which objects are target objects of the utility, use the DISPLAY DATABASE command, or run the DIAGNOSE utility with the DISPLAY SYSUTIL option. The DIAGNOSE utility should be used only under the direction of IBM Software Support.

Note: ACCESS(FORCE) will not successfully complete if the object you are trying to force was placed in a utility-read-only (UTRO), utility-read-write (UTRW), or utility-utility (UTUT) state by a utility running in a previous release of DB2. If this situation is encountered, DB2 issues message DSNIO41I. To reset the restrictive state, you must terminate the utility using the release of DB2 in which it was started.

A table space or index space that is started with ACCESS(FORCE) might be in an inconsistent state.

Usage notes

Data sets offline: Disk packs that contain partitions, table spaces, or indexes, do not necessarily need to be online when a database is started. Packs must, however, be online when partitions, table spaces, or indexes are first referred to. If they are not online, an error in opening occurs.

Table spaces and indexes explicitly stopped: If table spaces and indexes are stopped explicitly (using the STOP DATABASE command with the SPACENAM option), they must be started explicitly. Starting the database does not start table spaces or indexes that have been explicitly stopped.

Effect on objects marked with GRECP or with LPL entries: If a table space, partition, or index is in the group buffer pool RECOVER pending (GRECP) status, or if it has pages in the logical page list (LPL), the START DATABASE command begins recovery of the object. You must specify the SPACENAM option and ACCESS (RW) or (RO).

This recovery operation is performed even if SPACENAM specifies an object that is already started.

If the object is stopped when the command is issued, then the START DATABASE command both starts the object and clears the GRECP or LPL status. If the GRECP or LPL recovery action cannot complete, the object is still started.

If any table space or index space that is required to check command authority is unavailable, Installation SYSADM or Installation SYSOPR authority will be required to issue the START DATABASE command.

When recovering objects that are in GRECP or LPL status, avoid using pattern-matching characters (*) for both the database name and the space name. Multiple START DATABASE(*dbname*) SPACENAM(*) commands running in parallel should complete faster than one START DATABASE(*) SPACENAM(*) command.

If you use pattern-matching characters (*) for both the database name and space name, you must have DBMAINT authority and ensure that the catalog and directory databases have already been explicitly started in the following order:

- START DATABASE(DSNDB01) SPACENAM(*)
- START DATABASE(DSNDB06) SPACENAM(*)

Although not recommended, you can start an object using START DATABASE ACCESS(FORCE). That deletes all LPL and write error page range entries without recovering the pages. It also clears the GRECP status.

When a table space or partition is placed in the LPL because undo processing is needed for a NOT LOGGED table space, the **-START DATABASE** command does not remove the table space or partition from the LPL.

When starting a LOB table space defined as LOG NO and either in GRECP or having pages in the LPL, the LOB table space will be placed in the AUXW state and the LOB will be invalidated if DB2 detects that log records required for LPL recovery are missing due to the LOG NO attribute.

Use of ACCESS(FORCE): The ACCESS(FORCE) option is intended to be used when data has been restored to a previous level after an error, by DSN1COPY, or by a program that is not DB2 for z/OS, and the exception states resulting from the error still exist and cannot be reset. When using ACCESS(FORCE), it is up to the user to ensure the consistency of data with respect to DB2.

If an application process requests a transaction lock on a table space that is in a restrictive status (RECP) or has a required index in a restrictive status, DB2 acquires the lock. DB2 does not detect the status until the application tries to access the table space or index, when the application receives an error message indicating that the resource is not available (SQLCODE -904). After receiving this message, the application should release the lock, either by committing or rolling back (if the value of the RELEASE option is COMMIT) or by ending (if the value of RELEASE is DEALLOCATE). If you issue the command START DATABASE ACCESS(FORCE) for either the table space or the index space while the lock is in effect, the command fails.

If an object has retained locks (that is, a member of a DB2 data sharing group has failed and the locks it held on the object are retained in the lock structure), START DATABASE ACCESS (FORCE) is not allowed.

START DATABASE ACCESS(FORCE) does not execute if postponed abort or indoubt units of recovery exist. If you attempt to issue the START DATABASE ACCESS(FORCE) command in this situation, the command fails. FORCE cannot be used to reset the restart pending (RESTP) state.

Restricted mode (RO or UT): When a START DATABASE command for a restricted mode (RO and UT) takes effect depends on whether applications are started after the START DATABASE command has completed, or whether applications are executing at the time the command is issued. For applications that are started after START DATABASE has completed, access restrictions are effective immediately. For applications that are executing at the time START DATABASE is issued, the access restrictions take effect when a subsequent claim is requested or the application is allowed to run to completion. Whether the application is interrupted by the START DATABASE command depends on various factors. These factors include the ACCESS mode that is specified on the START DATABASE command, the type of drain activity, if any, on the table space or partition, and whether any cursors are being held on the table space or partition.

Do not start table spaces or index spaces for defined temporary tables with RO or UT access. You can start a temporary file database with UT access to accommodate the REPAIR DBD utility.

If the table space, index, or partition must be accessed in a mode that is incompatible with the ACCESS type currently in effect, DB2 issues a resource-unavailable message.

For shared-owner databases, a STOP DATABASE command must be issued to quiesce a database or table space prior to issuing the START DATABASE command.

| *Starting a table space partition in PRO restricted status:* When a table space that
| is in Persistent Read Only (PRO) restricted status is started, the partition remains
| in PRO restricted status.

Communications database or resource limit facility: If the communications database (CDB) or resource limit facility (RLF) is currently being used by any member of the data sharing group, any attempt to start either active database or table space with ACCESS(UT) fails.

Synchronous processing completion: Message DSN9022I indicates that synchronous processing has completed successfully.

Asynchronous processing completion: Recovery of objects in GRECP status or with pages on the LPL is performed asynchronously. Message DSNI022I is issued periodically to give you the progress of the recovery. The starting of databases, table spaces, or indexes (a synchronous task) often completes before the recovery operation starts. Therefore, when DB2 issues message DSN9022I, which indicates that synchronous processing has completed, the recovery of objects might not be complete.

Message DSNI006I is issued in response to the START DATABASE command when the object (table space or index space) that is identified by TYPE and NAME has group buffer pool recovery pending (GRECP) or logical page list (LPL) status, and recovery was triggered. The START DATABASE command does not complete until the asynchronous task of recovery completes.

Message DSNI021I indicates that asynchronous processing for an object has completed. You can issue the command DISPLAY DATABASE to determine whether the recovery operation for all objects is complete. If recovery is complete, the output from the command shows either a RW or a RO status without LPL or GRECP.

Starting a LOB table space: The **START DATABASE** command can be used to start LOB table spaces and indexes on auxiliary tables. LOB table spaces are started independently of the base table space with which the LOB table space is associated.

Examples

Example 1: Start table space DSN8S81E in database DSN8D81A. Recover the table space if it is in GRECP status or recover the pages on the LPL if one exists.

```
-START DATABASE (DSN8D81A) SPACENAM (DSN8S81E)
```

Example 2: Start all databases (except DSNDB01, DSNDB06, and work file databases) for which you have authority. Recovery for any objects with GRECP or LPL status is not performed.

```
-START DATABASE (*)
```

Example 3: Start the third and fourth partitions of table space DSN8S81E in database DSN8D81A for read-only access. Recover the partitions if they are in GRECP status or recover the pages on the LPL if one exists.

```
-START DATABASE (DSN8D81A) SPACENAM (DSN8S81E) PART (3,4) ACCESS (RO)
```

Example 4: Start all table spaces that begin with "T" and end with the string "IQUA03" in database DBIQUA01 for read and write access.

```
-START DATABASE (DBIQUA01) SPACENAM (T*IQUA03) ACCESS (RW)
```

This command produces output that is similar to the following output:

```
DSN9022I - DSNTDDIS 'START DATABASE' NORMAL COMPLETION
```

Example 5: Start clone objects.

```
-START DATABASE (MYDB*) SPACENAM (MYDB*SP) CLONE
```

Chapter 74. -START DB2 (DB2)

The DB2 command START DB2 initializes the DB2 subsystem. When the operation is complete, the DB2 subsystem is active and available to TSO applications and to other subsystems (for example, IMS and CICS).

The effect of restarting the system can be controlled by a conditional restart control record, which you create by using the DSNJU003 (change log inventory) utility. For more details about the effects, see "Usage notes" on page 437.

Abbreviation: -STA DB2

Environment

This command can be issued only from a z/OS console. The name of the DB2 subsystem is determined by the command prefix. For example, -START indicates that the DB2 subsystem to be started is the one with '-' as the command prefix.

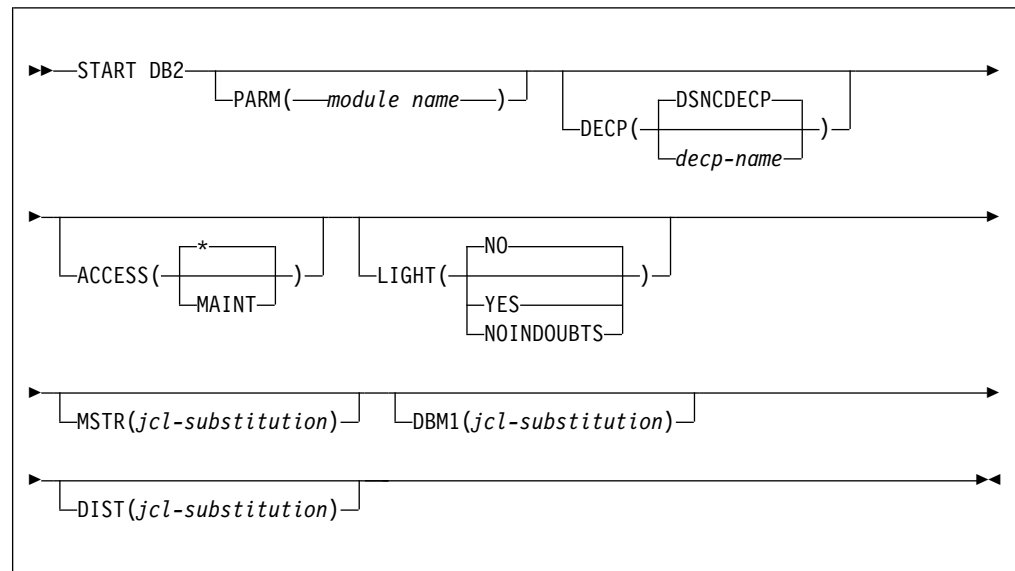
The command is rejected if the DB2 subsystem is already active. The restart recovery status of DB2 resources is determined from the prior DB2 shutdown status.

Data sharing scope: Member

Authorization

None is required. However, the command can be executed only from a z/OS console with the START command capability.

Syntax



Option descriptions

None of the following options are required.

PARM (*module-name*)

Specifies the load module that contains the DB2 subsystem parameters.

The default is the name of the parameter module that was specified on panel DSNTIPO when the installation CLIST was run. The default can also be changed if you update ZPARM(*default-module-name*) in the *ssnm*MSTR DB2 subsystem startup procedure.

DECP

Specifies the name of the load module that contains DB2 application parameter defaults.

decpr-name is the name of a module that is provided by the installation. The default name is DSNHDECP. If the specified module is not found or cannot be loaded, an error is issued, and the DB2 subsystem does not start.

ACCESS

Specifies whether access to DB2 is to be general or restricted.

Abbreviation: ACC

(*)

Makes access general; all authorized users can connect to DB2.

The **default** is ACCESS(*).

(MAINT)

Prohibits access to any authorization IDs other than installation SYSADM, installation SYSOPR, and SECADM.

For data sharing, ACCESS(MAINT) restricts access on only the DB2 member on which you execute this command. Other members of the data sharing group are unaffected.

LIGHT

Specifies whether a light restart is to be performed in a data sharing environment.

(NO)

A light restart is not performed.

(YES)

Specifies that a light restart is to be performed. DB2 starts with reduced storage, waits for resolution of indoubt units of recovery, and terminates normally after freeing retained locks.

(NOINDOUBTS)

Specifies that DB2, during a light restart, does not wait for indoubt units of recovery to resolve before it terminates.

MSTR (*jcl-substitution*)

Gives parameters and values to be substituted in the EXEC statement of the JCL that executes the startup procedure for the system services address space.

DBM1 (*jcl-substitution*)

Gives parameters and values to be substituted in the EXEC statement of the JCL that executes the startup procedure for the database services address space.

DIST (*jcl-substitution*)

Gives parameters and values to be substituted in the EXEC statement of the JCL that executes the startup procedure for the distributed services address space.

(*jcl-substitution*)

One or more character strings of the form *keyword = value* , enclosed between apostrophes. If you use more than one character string, separate each string with a comma and enclose the entire list between a single pair of apostrophes.

Recommendation: Omit the keyword and use the parameters that are provided in the startup procedure.

Usage notes

Command prefix: If your installation has more than one DB2 subsystem, you must define more than one command prefix.

Conditional restart: A conditional restart control record can prevent a complete restart and specify current status rebuild only. In that case, the following actions occur during restart:

- Log records are processed to the extent that is determined by the conditional restart control record.
- The following values are displayed:
 - The relative byte address (RBA) of the start of the active log
 - The RBA of the checkpoint record
 - The status counts for units of recovery
 - The display table for restart unit of work elements
- The restart operation terminates with an abend.

Light restart with ARM: To enable a light restart in an ARM environment, you must code an ARM policy for DB2 and IRLM.

The following example shows an ARM policy for DB2, where the element name is the DB2 data sharing group name and member name concatenated. For example, DSNDB0GDB1G.

```
ELEMENT(elementname)
```

```
RESTART_METHOD(SYSTEM,STC,'cmdprfx STA DB2,LIGHT(YES)')
```

The following example shows an ARM policy for IRLM, where the element name is the IRLM group name and the ID concatenated. For example, DXRDB0GDJ1G001.

```
ELEMENT(elementname)
```

```
RESTART_METHOD(SYSTEM,STC,'cmdprfx S irlmproc')
```

The element name that DB2 uses is the DB2 data sharing group name and member name concatenated. For example, DSNDB0GDB1G.F

Endless wait during start: The start operation might begin and fail to complete, if the system services address space starts and the database services address space cannot start. If a seemingly endless wait occurs, cancel the system services address space from the console, and check both startup procedures for JCL errors.

Starting members of a data sharing group: To start members of a data sharing group, you must enter a START DB2 command for each subsystem in the group. If it is the first startup of the group, you must start the originating member (the first DB2 that was installed) first.

Examples

Example 1: Start the DB2 subsystem.

```
-START DB2
```

Example 2: Start the DB2 subsystem, and provide a new value for the REGION parameter in the startup procedure for the system services address space.

```
-START DB2 MSTR('REGION=6000K')
```

Example 3: Start the DB2 subsystem. Assuming that the EXEC statement of the JCL that executes the startup procedure for the system services address space uses the symbol RGN, provide a value for that symbol.

```
-START DB2 MSTR('RGN=6000K')
```

Example 4: DB2 subsystems DB1G and DB2G are members of a data sharing group. Both were installed with a command prefix scope of STARTED. Start DB1G and DB2G by routing the appropriate commands to the z/OS system on which they are to be started, MVS1 and MVS2.

```
ROUTE MVS1,-DB1G START DB2  
ROUTE MVS2,-DB2G START DB2
```

Example 5: Start the DB2 subsystem, then provide the parameter module and a value for the DECP option. Enter either DSNCDECP or another *decp-name*

```
-START DB2 PARM(VA1AZNS) DECP(DSNHDVA1)
```

Chapter 75. -START DDF (DB2)

The DB2 command START DDF starts the distributed data facility (DDF) if it is not already started.

Abbreviation: -STA DDF

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



```
▶▶—START DDF—▶▶
```

Usage note

The START DDF command activates the DDF interface to VTAM and TCP/IP. When this command is issued after STOP DDF MODE(SUSPEND), suspended threads are resumed and DDF activity continues.

Example

Start the distributed data facility.

```
-START DDF
```

Chapter 76. -START FUNCTION SPECIFIC (DB2)

The DB2 command START FUNCTION SPECIFIC starts an external function that is stopped. Built-in functions or user-defined functions that are sourced on another function cannot be started with this command.

On successful completion of the command, queued requests for the specified functions begin executing. The abend counts for those functions are set to zero.

You do not need to issue the START FUNCTION SPECIFIC command when defining a new function to DB2. DB2 automatically starts the new function on the first SQL statement that invokes the new function.

Historical statistics in the DISPLAY FUNCTION SPECIFIC report (MAXQUE, TIMEOUT) are reset each time a START FUNCTION SPECIFIC command is issued for a given function.

Abbreviation: -STA FUNC SPEC

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel, an IMS or CICS terminal, or a program that uses the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the value of the SCOPE option.

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities for each function:

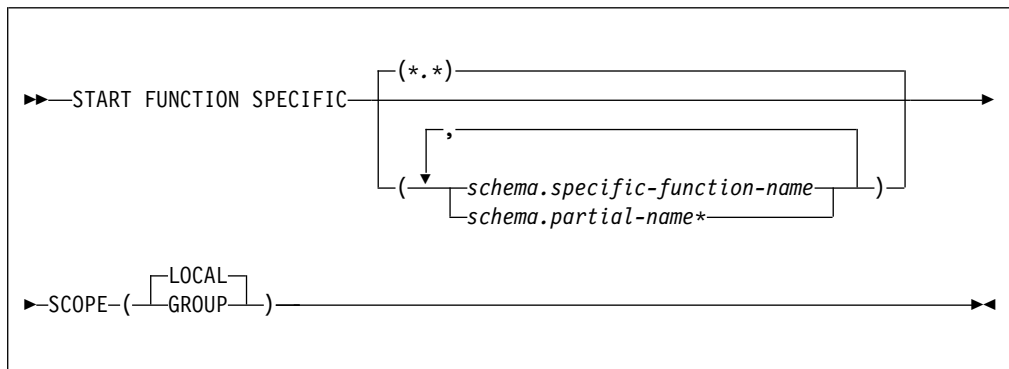
- | • Ownership of the function
- | • RECOVER privilege
- | • System DBADM authority
- | • SYSOPR authority
- | • SYSCTRL authority
- | • SYSADM authority

If you specify START FUNCTION SPECIFIC *.* or *schema.partial-name **, the privilege set of the process must include one of the following authorities:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

* (asterisk) (*.*)

Starts all functions in all schemas. This is the default.

(*schema.specific-function-name*)

Starts the specific function name in the schema. You cannot specify a function name in the same way that you do in SQL; you must use the specific name. If a specific name was not specified on the CREATE FUNCTION statement, query SYSIBM.SYSROUTINES for the correct specific name:

```
SELECT SPECIFICNAME, PARM_COUNT
FROM SYSIBM.SYSROUTINES
WHERE NAME='function_name'
AND SCHEMA='schema_name';
```

For overloaded functions, this query can return multiple rows.

(*schema.partial-name **)

Starts all functions or a set of functions in the specified schema. The specific names of all functions in the set begin with *partial-name* and can end with any string, including the empty string. For example, *schema1.ABC** starts all functions with specific names that begin with ABC in *schema1*.

SCOPE

Specifies the scope of the command.

(LOCAL)

Specifies that the command applies only to the current member.

(GROUP)

Specifies that the command applies to all members of the data sharing group.

Usage notes

Language Environment in the WLM-established stored procedure address space :

The START FUNCTION SPECIFIC command does not refresh the Language Environment in the WLM-established stored procedure address space. You must issue the WLM command. For example, if you need to refresh the Language Environment to get new copies of user-defined function load modules, issue the following WLM command:

```
VARY WLM, APPL ENV=applenv, REFRESH
```


| *Considerations for SQL functions:* The START FUNCTION SPECIFIC command
| affects all versions of the SQL functions that you specify in the command.

Examples

Example 1: Start all functions.

```
-START FUNCTION SPECIFIC
```

Output that is similar to the following output is generated:

```
DSNX973I - DSNX9ST2 START FUNCTION SPECIFIC SUCCESSFUL FOR *.*  
DSN9022I - DSNX9COM '-START FUNC' NORMAL COMPLETION
```

Example 2: Start functions USERFN1 and USERFN2. If any requests are queued for these functions, the functions are executed.

```
-START FUNCTION SPECIFIC(PAYROLL.USERFN1,PAYROLL.USERFN2)
```

Output that is similar to the following output is generated:

```
DSNX973I - DSNX9ST2 START FUNCTION SPECIFIC SUCCESSFUL FOR  
PAYROLL.USERFN1  
DSNX973I - DSNX9ST2 START FUNCTION SPECIFIC SUCCESSFUL FOR  
PAYROLL.USERFN2  
DSN9022I - DSNX9COM '-START FUNC' NORMAL COMPLETION
```

Chapter 77. START irlmproc (z/OS IRLM)

The `START irlmproc` command starts an IRLM component with a procedure that is defined by the installation. Symbolic parameters in the procedure can be overridden on the `START irlmproc` command.

Environment

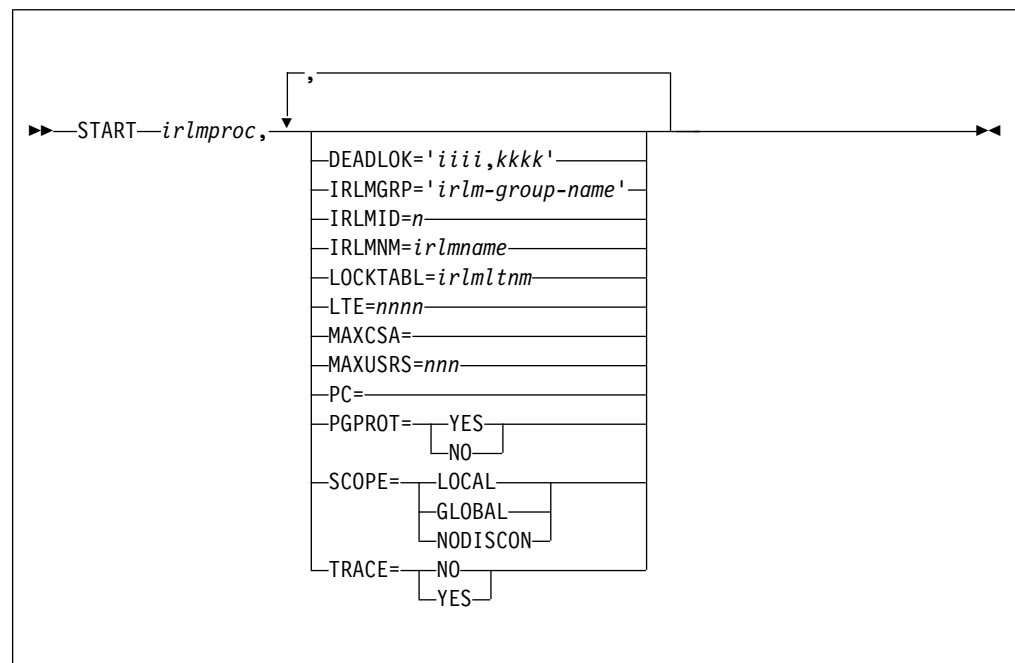
This command can be issued only from a z/OS console.

Data sharing scope: Member

Authorization

The command requires an appropriate level of operating system authority.

Syntax



Options must be separated by commas, with no spaces.

Option descriptions

irlmproc

Specifies the procedure name of the IRLM to be started.

None of the following options are required:

DEADLOK= ' iii,kkk '

Specifies the local deadlock-detection interval in seconds (*iii*), and the number of local cycles (*kkk*) that are to occur before a global detection is initiated.

iii

Is a one- to four-digit number. Values between 1 and 5 are interpreted as seconds. Values between 100 and 5000 are interpreted as milliseconds. Depending on the value that you enter, IRLM might substitute a smaller maximum value.

kkkk

Is a one- to four-digit number from 1 to 9999 that specifies the number of local deadlock cycles that must expire before global deadlock detection is performed. You can specify any value from 1 to 9999, but IRLM uses 1. The recommended value to specify is 1.

In a data sharing environment, IRLM synchronizes all of the DEADLOCK values in the group to the values specified on the most recent IRLM to join the group. The DEADLOCK values can be changed by starting a member with the values that you want. To reduce confusion, it is recommended that the installation specify the same value for DEADLOCK on all of its IRLM startup procedures and use the START *irlmproc* command to override this value only when the interval must be increased from its original value.

IRLMGRP= ' *irlm-group-name* '

Specifies the name of the cross system coupling facility (XCF) group, in a data sharing environment, to which the IRLM belongs as the lock manager for DBMSs that share the same data. All IRLMs in the same group must specify the same value for LOCKTABL and unique values for IRLMID.

The group name is used as the XCF group name. The name must not start with 'SYS' and must not be the same name specified for LOCKTABL.

In a non-data-sharing environment (SCOPE=LOCAL), IRLMGRP is ignored.

IRLMID= *n*

Specifies a decimal number that is used to distinguish between IRLMs in a data sharing group.

n can be either a one- to three-digit number from 1 to 255, or a printable character in quotation marks. Note that this IRLM ID does not relate directly to the limit of IRLM members that can be in the group. That limit is determined by the current hardware limits (currently 32).

When *n* is specified as a printable character, IRLM uses the EBCDIC value of the printable character as the IRLMID (such as X'C4'). The printable character must be surrounded by enough single quotes to permit IRLM to see it as a printable character. Because of the way that the operating system interprets quotes, single quotes must be on either side of the characters. For example, if you want to specify the printable character 'D', you must specify it here as IRLMID='D'.

A unique IRLMID must be specified for each IRLM in a group (IRLMs with the same value specified for the IRLMGRP option).

IRLMNM= *irlmname*

Specifies a 4-byte z/OS subsystem name assigned to this IRLM. (Although z/OS can accept names that are less than 4 bytes, IRLM requires a 4-byte name.)

LOCKTABL= *irlmltnm*

Specifies the lock table to be used by this group. This option is overridden by DB2; it is needed in an IMS environment.

In a non-data-sharing environment (SCOPE=LOCAL), LOCKTABL is ignored.

LTE= *nnnn*

Specifies the number of lock table entries that are required in the coupling facility (CF) lock structure in units of 1048576 entries. LTE= can have a value of blank, zero, or any exact power of two up to 2048 (inclusive). The number of lock table entries in the group is determined by the first IRLM to connect to the group during initial structure allocation or during REBUILD.

The LTE value is used in the following order:

1. The value that is specified using MODIFY irlmproc,SET,LTE= if the value is greater than zero.
2. The value from LTE= in the irlmproc if the value is greater than zero.
3. The value that is determined by the existing logic, which divides the XES structure size returned on the IXCQUERY call by two times LTE width. The result is rounded to the nearest power of two, which the existing logic uses for the value.

Note: The LTE width is determined by the MAXUSRS value.

If IRLM attempts to use a value from MODIFY irlmproc,SET,LTE= that is greater than the available storage in the structure size returned by XES IXCQUERY, the value for the LTE= in the irlmproc is used. If this value is greater than the available storage, IRLM uses the value that is determined by the existing logic.

Table 25. Some common values for lock table entries and the required lock table storage

For LTE=	Lock Table Storage needed for 2-byte entries	Lock Table Storage needed for 4-byte entries
8	16 MB	32 MB
16	32 MB	64 MB
32	64 MB	128 MB
64	128 MB	256 MB
128	256 MB	512 MB
256	512 MB	1024 MB

MAXCSA=

MAXCSA= is a required positional parameter but is currently unused.

MAXUSRS= *nnn*

Specifies the initial maximum number of members in the data sharing group, set by the IRLM which results in structure allocation. The specified value determines the size of each lock entry in the lock table portion of the lock structure, as shown in the following table.

Table 26. Effect of MAXUSRS on initial size of lock table entry

MAXUSRS	Initial size of lock entry
7 or less	2 bytes
≥ 8 and < 24	4 bytes
≥ 24 and < 33	8 bytes

mmn must be a one- to two-digit number from 1 to 32. The default is 7. The recommended value is 7 or less.

In a non-data-sharing environment (SCOPE=LOCAL), MAXUSRS is ignored.

PC=

PC= is a required positional parameter but is currently unused.

PGPROT=

Specifies whether the IRLM load modules that are resident in common storage are placed in z/OS page-protected storage.

YES The IRLM load modules that are resident in common storage are placed in z/OS page-protected storage.

NO The IRLM load modules that are resident in common storage are not placed in z/OS page-protected storage.

SCOPE=

Specifies whether the IRLM is to be used in a data sharing environment.

LOCAL

Specifies the IRLM is in a non-data-sharing environment and there is no intersystem sharing.

GLOBAL

Specifies the IRLM is in a data sharing environment and that intersystem sharing is to be performed.

NODISCON

Specifies that IRLM is in a data sharing environment and that intersystem sharing is to be performed. IRLM remains connected to the data sharing group even when no database management systems are identified to it. You must explicitly stop IRLM to bring it down.

If you specify the NODISCON option, there is less impact on other systems when a DB2 subsystem fails because the operating system is not required to perform certain recovery actions that it normally performs when IRLM comes down. Using the NODISCON option might allow DB2 to restart more quickly after a DB2 subsystem normally or abnormally terminates because it does not have to wait for IRLM to rejoin the IRLM data sharing group.

TRACE=

Specifies whether the IRLM is to capture traces in wrap-around IRLM buffers. Each buffer is reused when the previous buffer is filled. Traces are captured at IRLM startup. You should specify TRACE=YES in the irlmproc to place traces in wrap-around mode.

NO

Does not capture traces unless the TRACE CT command is issued.

YES

Captures traces in wrap-around buffers.

Examples

Example: This command starts the IRLM with a lock table storage size of 64 MB, assuming a width of 2-bytes for each lock table entry.

Enter the following command on the system console:

S irlmproc,LTE=32

If this value is correct, message DXR132I, which is displayed after successful connection to the lock structure, displays the value used by IRLM. If this value is incorrect, START will terminate with DXR116E CODE=24 and ABENDU2018. This value is only used if SCOPE=GLOBAL or SCOPE=NODISCON and has a default value calculated by IRLM.

Chapter 78. -START PROCEDURE (DB2)

The DB2 command START PROCEDURE activates the definition of a stored procedure that is stopped or refreshes one that is stored in the cache. You can qualify stored procedure names with a schema name.

On successful completion of the command, queued requests for the specified stored procedures begin to execute. The abend counts for the specified procedures are set to zero. DB2 resets the MAXQUE and TIMEOUT statistics to 0 each time that you execute the START PROCEDURE command.

You do not need to issue START PROCEDURE when you define a new stored procedure to DB2. DB2 automatically activates the new definition when it first receives an SQL CALL statement for the new procedure.

Abbreviation: -STA PROC

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the value of the SCOPE option.

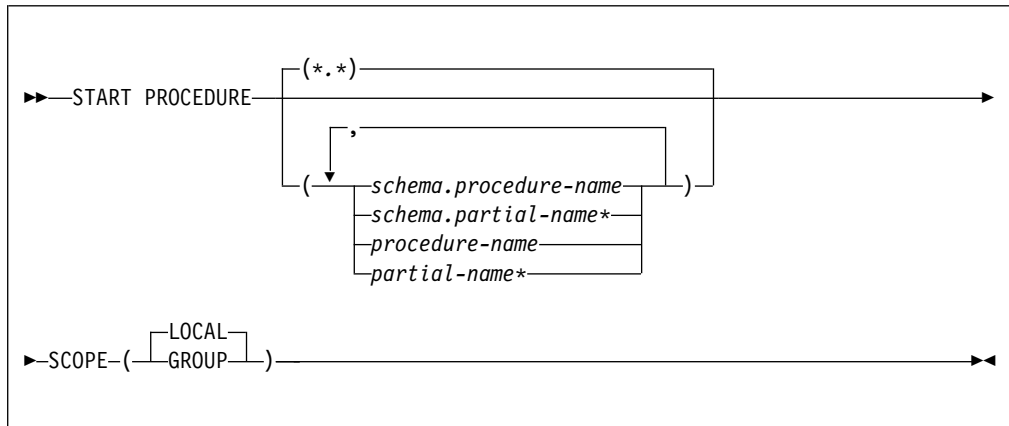
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- Ownership of the stored procedure
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

(*.*)

Marks all stored procedures in all schemas as available to be called.

(*schema.procedure-name*)

Starts the specified stored procedure in the specified schema.

(*schema.partial-name **)

Starts a set of stored procedures in the specified schema. The names of all procedures in the set begin with *partial-name* and can end with any string, including the empty string. For example, PAYROLL.ABC* starts all stored procedures with names that begin with ABC in the PAYROLL schema.

procedure-name

Marks one or more specific stored procedures as available to be called.

*partial-name **

Marks a set of stored procedures in the SYSPROC schema as available to be called. The names of all procedures in the set begin with *partial-name* and can end with any string, including the empty string. For example, ABC* starts all stored procedure names that begin with ABC in the SYSPROC schema.

SCOPE

Specifies the scope of the command.

(LOCAL)

Starts the specified stored procedures in only the local members.

(GROUP)

Starts the specified stored procedures in all members of the data sharing group.

Usage notes

Errors in a definition of a stored procedure: Errors are detected at create time for a stored procedure.

Considerations for native SQL procedures: The START PROCEDURE command affects all versions of the native SQL procedures that you specify in the command.

Examples

Example 1: Start all stored procedures.

```
-START PROCEDURE
```

This command produces output that is similar to the following output:

```
DSNX946I - DSNX9ST2 START PROCEDURE SUCCESSFUL FOR *.*  
DSN9022I - DSNX9COM '-START PROC' NORMAL COMPLETION
```

Example 2: Make the stored procedures USERPRC1 and USERPRC2 available to be called, and start any requests that are waiting for those procedures.

```
-START PROCEDURE(USERPRC1,USERPRC2)
```

This command produces output that is similar to the following output:

```
DSNX946I - DSNX9ST2 START PROCEDURE SUCCESSFUL FOR USERPRC1  
DSNX946I - DSNX9ST2 START PROCEDURE SUCCESSFUL FOR USERPRC2  
DSN9022I - DSNX9COM '-START PROC' NORMAL COMPLETION
```

Chapter 79. -START PROFILE (DB2)

The DB2 command START PROFILE loads or reloads the profile table into a data structure in memory.

If this data structure already exists, DB2 deletes it, and a new structure is created. The profile table must be loaded by issuing the command above explicitly. The table is not loaded when the database is initialized at the startup time. After loading the database, the functions specified in the profile become active. Only the rows in the profile table with column PROFILE_ENABLED='Y' are activated.

Abbreviation: -STA PROFILE

Environment

This command can be issued from the z/OS console, through a batch job or the instrumentation facility interface (IFI).

Data sharing scope: Member

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SQLADM authority
- System DBADM authority
- SYSOPR authority
- SYSTRM authority
- SYSADM authority

Syntax

```
▶▶—START PROFILE—◀◀
```

Examples


Example 1: This command is required to load the profile table into memory.


```
-START PROFILE
```


Related concepts:


 Profiles for monitoring and controlling DB2 for z/OS subsystems (DB2 Performance)

Related tasks:


 Using profiles to monitor and optimize DB2 for z/OS subsystems (DB2 Performance)

 [Optimizing subsystem parameters for SQL statements by using profiles \(DB2 Performance\)](#)

 [Monitoring threads and connections by using profiles \(DB2 Performance\)](#)

 [Modeling a production environment on a test subsystem \(DB2 Performance\)](#)

Related reference:

 [Profile tables \(DB2 Performance\)](#)

Chapter 80. -START RLIMIT (DB2)

The DB2 command START RLIMIT starts the resource limit facility (governor) and specifies a resource limit specification table for the facility to use.

You can issue START RLIMIT even if the resource limit facility is already active. However, you must specify a resource limit table that is not already active. Only the resource limit table that you identify in the command applies for new threads. Existing threads continue to be subject to the limits in the table that was active at the time they were created.

Abbreviation: -STA RLIM

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

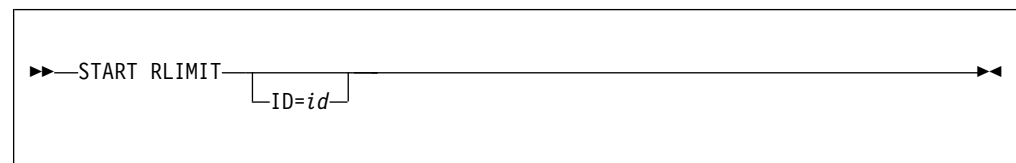
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option description

The following keyword is optional.

ID= *id*

Identifies the resource limit specification table for the governor to use.

id is the one or two identification character that is specified when the table is created.

The full name of the table is *authid.DSNRLSTid* or *authid.DSNRLMTid*, where *authid* is the value that is specified in field RESOURCE AUTHID on installation panel DSNTIPP.

The **default** ID is specified by the value of the RLFTBL subsystem parameter.

Example

Start the resource limit facility.

```
-START RLIMIT ID=01
```

Related tasks:

[↗](#) Setting limits for system resource usage by using the resource limit facility (DB2 Performance)

[↗](#) Starting and stopping resource limit tables (DB2 Performance)

[↗](#) Limiting resource usage for packages (DB2 Performance)

[↗](#) Limiting resource usage by client information (DB2 Performance)

Related reference:

```
-STOP RLIMIT (DB2)
```

```
-DISPLAY RLIMIT (DB2)
```

[↗](#) DSNRLSTxx resource limit tables (DB2 Performance)

[↗](#) DSNRLMTxx resource limit tables (DB2 Performance)

[↗](#) DSNTIPP: Protection panel (DB2 Installation and Migration)

[↗](#) RLST NAME SUFFIX field (RLFTBL subsystem parameter) (DB2 Installation and Migration)

Chapter 81. -START TRACE (DB2)

The DB2 command `START TRACE` starts DB2 traces.

An additional option for this command and additional values for a few other options exist. This additional information is intended for service and use under the direction of IBM Software Support.

Abbreviation: -STA TRA

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the value of the `SCOPE` option.

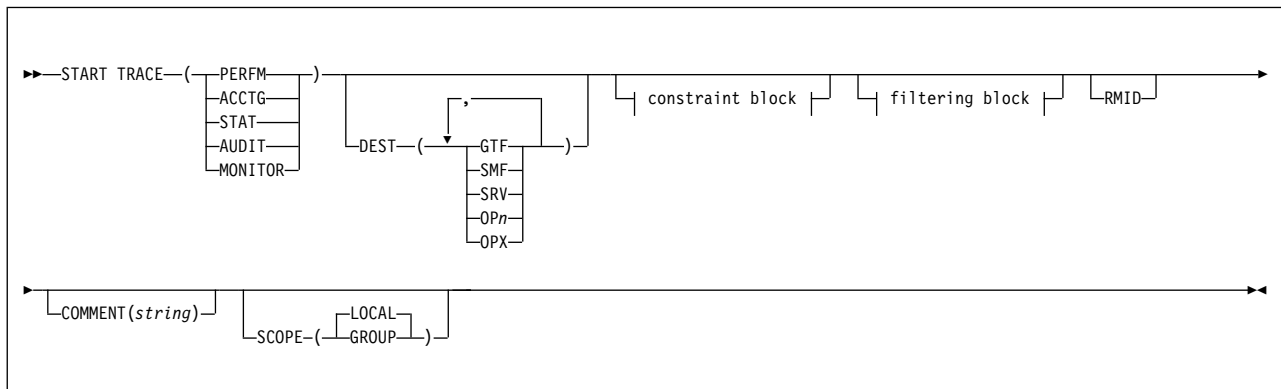
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

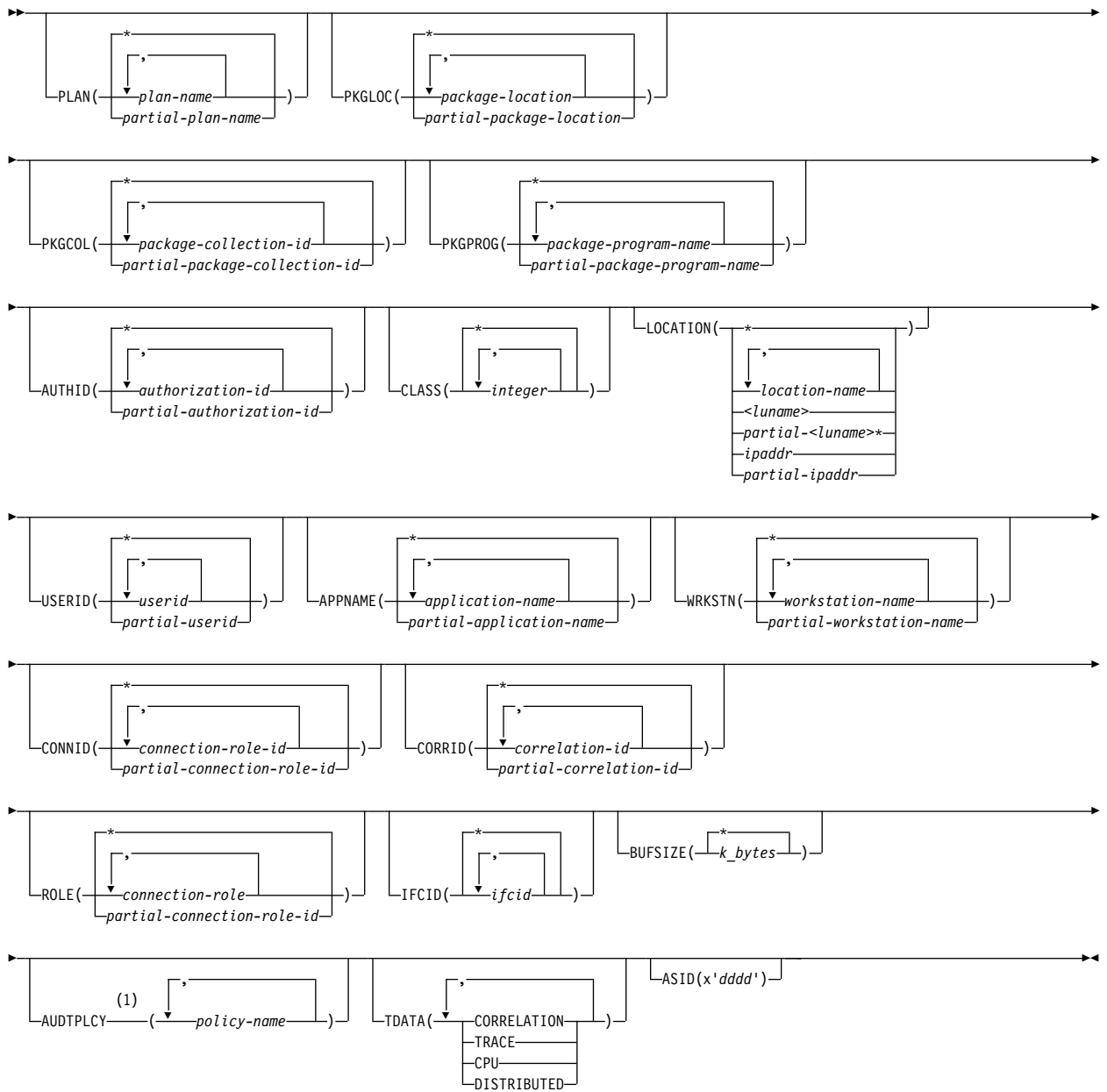
- TRACE privilege
- SQLADM authority
- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority
- SECADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax

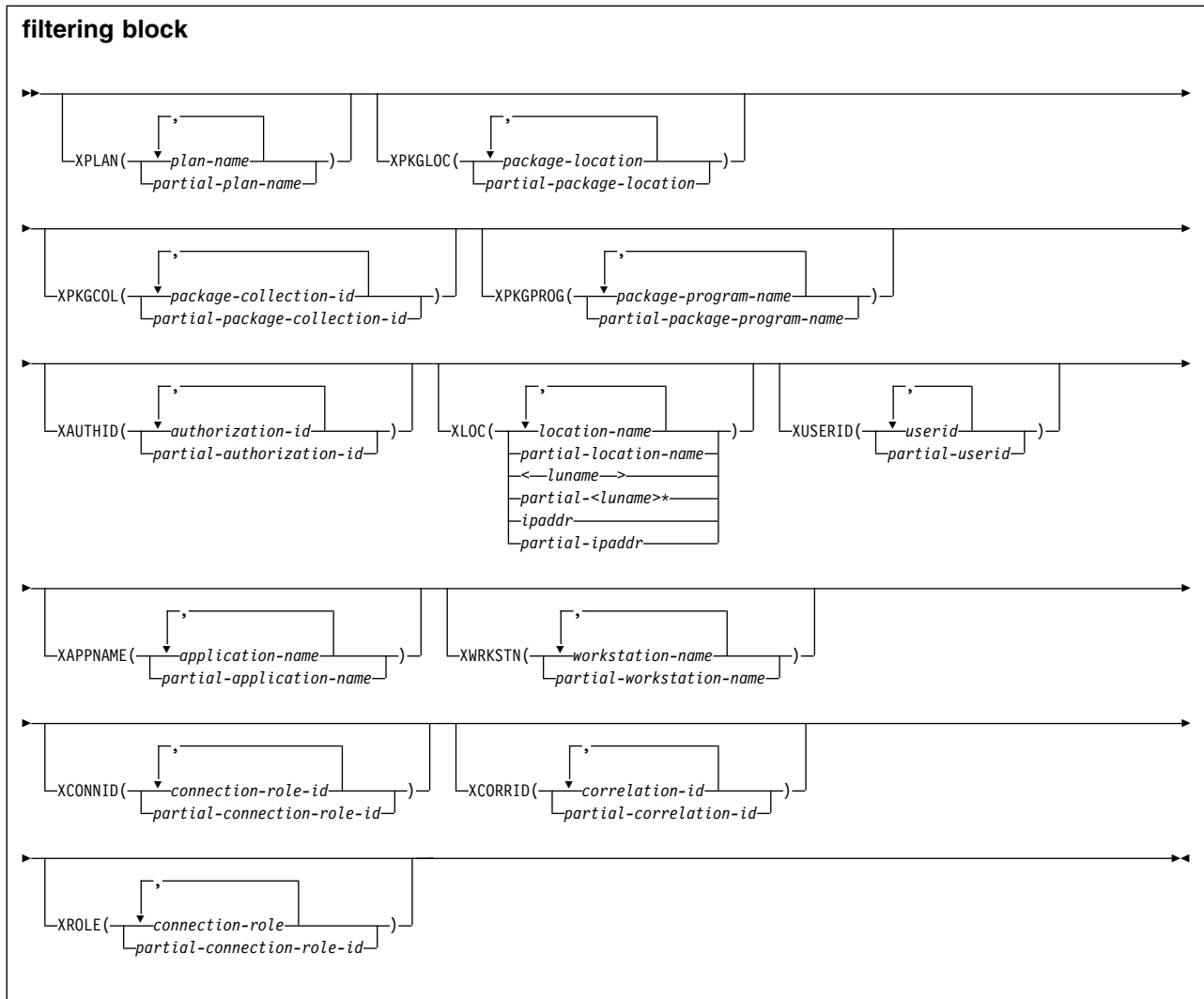


constraint block



Notes:

- 1 You cannot specify CLASS or IFCID with AUDTPLCY. AUDTPLCY applies to trace type AUDIT.



Option descriptions

You must specify a trace type.

The options PERFM, ACCTG, STAT, AUDIT, and MONITOR identify the type of trace that is started.

(PERFM)

The *performance trace* is intended for performance analysis and tuning. This trace includes records of specific events in the system, including events related to distributed data processing. The data can be used for program, resource, user, and subsystem-related tuning.

Abbreviation: P

(ACCTG)

The *accounting trace* records transaction-level data that is written when the processing for a transaction is completed. It provides data that enables you to conduct DB2 capacity planning and to tune application programs.

Abbreviation: A

(**STAT**)

The *statistics trace* collects statistical data that is broadcast by various components of DB2 at certain time intervals. You can specify intervals for statistics collection during installation.

Abbreviation: S

LOCATION cannot be specified when you choose a statistics trace.

(**AUDIT**)

The *audit trace* collects information about DB2 security controls and can be used to ensure that data access is allowed only for authorized purposes.

Abbreviation: AU

(**MONITOR**)

The *monitor trace* enables attached monitor programs to access DB2 trace data through calls to the instrumentation facility interface (IFI). Monitor programs can access the trace data asynchronously through an OP x buffer by issuing READA requests, or synchronously in the monitor return area by issuing READS requests.

Abbreviation: MON

SCOPE

Specifies the scope of the command in a data sharing group.

(**LOCAL**)

Specify to start traces on the local member only.

(**GROUP**)

Specify to start traces on all members of the data sharing group.

ASID(x' *ddd* $'$)

Specifies the address space for which trace data is collected.

ddd is a four-byte hexadecimal address space ID (ASID).

COMMENT(*string*)

Gives a comment that is reproduced in the trace output (except in the resident trace tables). This option can be used to record why the command was issued.

string is any character string; it must be enclosed between apostrophes if it includes a blank, comma, or special character.

RMID

Specifies resource manager identifier. You can specify up to 8 valid RMIDs, which are one or two digit identifiers. You cannot specify RMID for accounting or statistic traces.

DEST

Specifies where the trace output is to be recorded. You can use more than one value, but do not use the same value twice. If you do not specify a value, the trace output is sent to the default destination shown in the following table.

If the specified destination is not active or becomes inactive after you issue the START TRACE command, you receive message DSNW133I, which indicates that the trace data is lost. This applies for destinations GTF, SRV, and SMF. You also receive this message for destinations OP n and OP x if START TRACE is not issued by an application program.

Abbreviation: D

The allowable values and the default value depend on the type of trace started, as shown in the following table.

Table 27. Allowable destinations for each trace type

Type	GTF	SMF	SRV	OP _n	OPX
PERFM	Default	Allowed	Allowed	Allowed	Allowed
ACCTG	Allowed	Default	Allowed	Allowed	Allowed
STAT	Allowed	Default	Allowed	Allowed	Allowed
AUDIT	Allowed	Default	Allowed	Allowed	Allowed
MONITOR	Allowed	Allowed	Allowed	Allowed	Default

The meaning of each value is as follows:

GTF

The z/OS generalized trace facility (GTF). The record identifier for records from DB2 is X'0FB9'.

SMF

The system management facility. The SMF record type of DB2 trace records depends on the IFCID record, as follows:

IFCID record

SMF record type

1 (System Services Statistics)

0100

2 (Database Services Statistics)

0100

3 (Agent Accounting)

0101

202 (Dynamic System Parameters)

0100

225 (System Storage[®] Summary Statistics)

0100

230 (Data Sharing Global Statistics)

0100

239 (AGENT ACCOUNTING OVERFLOW)

0101

All Others

0102

SRV

An exit to a user-written routine. For instructions and an example of how to write such a routine, see the macro DSNWVSER in library *prefix.SDSNMACS*.

OP_n

A specific destination.

n can be an integer from 1 to 8.

OPX

A generic destination which uses the first free OP_n slot.

Only applications that start a trace to an OP_n buffer can read that buffer.

All traces to an OPX destination must be stopped before the buffer is marked as not in use. Traces that are started to an OPX buffer that was formerly in use write over the storage any previous traces had set.

The constraint and filtering blocks

The constraint and filtering blocks place optional limits on the kinds of data that are collected by the trace. The filtering options are exclusionary equivalents to the corresponding constraint options.

Only the CLASS constraint option can be specified for starting a statistics trace. Any constraint option or filter option can be specified for starting all other trace types. However, DB2 does not use any filters or constraints that you specify when you start a trace for any of the following IFCIDs:

1	2	4	5	104	105	107	124	129	147
148	149	150	185	186	199	202	217	225	230
254	306	316	317	365	373	401	402		

A START TRACE command can contain multiple constraint options, multiple filtering options, or a combination of both. A constraint or a filtering option can contain multiple values. However, the command must not contain multiple constraint options that each contain multiple values. A single constraint option that has multiple values can be specified with:

- Multiple other constraint options, each of which has a single value
- Multiple filtering options, each of which has a single value or multiples values

An error message is issued if the START TRACE command contains two or more constraint options that each have multiple options.

The meaning of each option is as follows. Filtering options are described with their corresponding constraint options.

PLAN(*plan-name* , ...) or XPLAN(*plan-name* , ...)

Introduces a list of specific plans for which trace information is gathered. Use PLAN to constrain the trace to the specified plans or XPLAN to exclude the specified plans. You cannot use this option for a STAT trace.

The **default** is PLAN(*).

(*)

Starts a trace for all plans.

plan-name

Is the name of an application plan. You can use up to eight names; a separate trace is started for each name. If you use more than one name, you can use only one value for AUTHID and LOCATION.

PKGLOC or XPKGLOC

Specifies the location name where the package is bound. Use PKGLOC to constrain the trace to the specified locations or XPKGLOC to exclude the specified locations.

PKGCOL or XPKGCOL

Specifies the package collection name. Use PKGCOL to constrain the trace to the specified collections or XPKGCOL to exclude the specified collections.

| **PKGPROG or XPKGPROG**

| Specifies the DBRM or program name. Use PKGPROG to constrain the trace to
| the specified programs or XPKGPROG to exclude the specified programs.

| **AUTHID(*authorization-id* , ...) or XAUTHID(*authorization-id* , ...)**

| Introduces a list of specific authorization IDs for which trace information is
| gathered. Use AUTHID to constrain the trace to the specified authorization IDs
| or XAUTHID to exclude the specified authorization IDs. The authorization IDs
| specified must be the primary authorization IDs. You cannot use this option for
| a STAT trace.

| The **default** is AUTHID(*_*).

| (*_*)

| Starts a trace for all authorization IDs.

| *authorization-id*

| Specifies an authorization ID. You can use up to eight identifiers; a
| separate trace is started for each identifier. If you use more than one
| identifier, you can use only one value for PLAN and LOCATION.

| **LOCATION(*location-name* , ...) or XLOC(*location-name* , ...)**

| Specifies a list of location names for which trace information is gathered. Use
| LOCATION to constrain the trace to the specified locations or XLOC to
| exclude the specified locations. The use of the LOCATION or XLOC option
| precludes tracing threads that have no distributed data relationship.
| LOCATION or XLOC cannot be specified when you want to start a statistics
| trace.

| *location-name*

| Identifies the DB2 subsystems whose distributed threads you want to trace.
| Activates the DB2 trace for the remote TCP/IP or SNA location that you
| specify by *location-name*.

| You can specify up to eight locations; a separate trace is started for each
| one. You can specify only one location if you use more than one plan name
| or authorization ID.

| <*luname*>

| Activates the DB2 trace for the remote clients that are connected to DDF
| through the remote SNA LU name that you specified in *luname*.

| *ipaddr*

| Activates the DB2 trace for the remote clients that are connected to DDF
| through the remote TCP/IP host. *ipaddr* is the IP address.

| The format of the IP address depends on whether the TCP/IP stack has
| been configured as an IPv4-only stack or a dual-mode stack. An IPv4 stack
| supports only the IPv4 protocol. A dual-mode stack supports IPv4 and
| IPv6 protocols.

- For an IPv4-only stack, you enter the IP address in IPv4 format:

| *x.x.x.x*

- For a dual-mode stack:

| – You enter an IPv6 address in IPv6 format:

| *y:y:y:y:y:y*

| – You enter an IPv4 address in dual IPv6 and IPv4 format, with ::FFFF:
| as the IPv6 portion:

| *::FFFF:x.x.x.x*

(*)

Indicates that you want to start trace events that occur under distributed threads regardless of which location they are connected to. Specifying the local location name is equivalent to specifying LOCATION(*).

Clients other than DB2 for z/OS: DB2 for z/OS does not receive a location name from clients that are not DB2 for z/OS subsystems. To start a trace for a client that is not a DB2 for z/OS subsystem, enter its LUNAME or IP address. Enclose the LUNAME by the less-than (<) and greater-than (>) symbols. Enter the IP address in the form *nnn.nnn.nnn.nnn*. For example, to start a trace for a client with the LUNAME of LULA, enter the following command:

```
-START TRACE (PERFM) CLASS (*) LOCATION (<LULA>)
```

To start a trace for a client with the IP address of 123.34.101.98, enter the following command:

```
-START TRACE (PERFM) CLASS (*) LOCATION (::FFFF:123.34.101.98)
```

USERID or XUSERID

Specifies the user ID. Use **USERID** to constrain the trace to the specified user IDs or **XUSERID** to exclude the specified user IDs. You can specify multiple values and wildcard values as described in Usage notes.

APPNAME or XAPPNAME

Specifies the application name. Use **APPNAME** to constrain the trace to the specified applications or **XAPPNAME** to exclude the specified applications. You can specify multiple values and wildcard values as described in Usage notes.

WRKSTN or XWRKSTN

Specifies the workstation name. Use **WRKSTN** to constrain the trace to the specified workstations or **XWRKSTN** to exclude the specified workstations. You can specify multiple values and wildcard values as described in Usage notes.

CONNID or XCONNID

Specifies the connection ID. Use **CONNID** to constrain the trace to the specified connections or **XCONNID** to exclude the specified connections.

The **CONNID** or **XCONNID** value is one of the following values:

CONNID value	Type of connection to DB2
BATCH	Batch
TSO	TSO
DB2CALL	QMF™
UTILITY	DB2 utility
<i>subsystem-id</i>	DB2 internal
<i>cics-connection-name</i>	CICS
<i>ims-connection-name</i>	IMS
RRSAF	RRSAF
<i>connection-name</i>	A connection from a remote DB2 for z/OS requester. This value is the connection name of the thread at the requesting location.
SERVER	A connection from a remote requester that is not DB2 for z/OS.

CORRID or XCORRID

Specifies the correlation ID. Use CORRID to constrain the trace to the specified correlation IDs or XCORRID to exclude the specified correlation IDs.

ROLE or XROLE

Specifies the connection roles. Use ROLE to constrain the trace to the specified roles or XROLE to exclude the specified roles.

CLASS(integer , ...)

Introduces a list of classes of data gathered. What classes are allowable, and their meaning, depends on the type of trace started.

Abbreviation: C

When the CLASS option is omitted, the *default classes* within the trace type are activated. The default classes are identified in the Description column of the following tables.

(*)

Starts a trace for all classes of the trace type.

integer

Any number in the following table. You can use any number of classes that are allowed for the type of trace started.

Accounting trace (ACCTG)

Table 28. Classes for DB2 accounting trace

Class	Description of class	Activated IFCIDs
1	Standard accounting data. This default class is also activated when you omit the CLASS keyword from the START TRACE command when you start the accounting trace.	0003, 0106, 0200, 0239
2	Entry or exit from DB2 event signaling.	0232
3	Elapsed wait time in DB2.	0006-0009, 0032, 0033, 0044, 0045, 0117, 0118, 0127, 0128, 0170, 0171, 0174, 0175, 0213-0216, 0226, 0227, 0242, 0243, 0321, 0322, 0329, 0378, 0379
4	Installation-defined accounting record.	0151
5	Time spent processing IFI requests.	0187
6	Reserved.	
7	Package-level accounting in-DB2 time.	0200, 0232, 0239, 0240
8	Package-level accounting wait time in DB2.	0006-0009, 0032, 0033, 0044, 0045, 0117, 0118, 0127, 0128, 0170, 0171, 0174, 0175, 0213-0216, 0226, 0227, 0239, 0241-0243, 0321, 0322, 0378, 0379
10	Package detail.	0239
	One of the following traces must also be activated before the IFCID 0239 records are written:	
	• Accounting class 7	
	• Accounting class 8	
	• Monitor class 7	
	• Monitor class 8	
11-29	Reserved.	
30 - 32	Available for local use.	

Audit trace (AUDIT)

Table 29. Classes for DB2 audit trace

Class	Description of class	Activated IFCIDs
1	Access attempts denied due to inadequate authorization. This default class is also activated when you omit the CLASS keyword from the START TRACE command when you start the audit trace.	0140
2	Explicit GRANT and REVOKE.	0141
3	CREATE, ALTER, and DROP operations against audited tables.	0142
4	First change of audited object.	0143
5	First read of audited object.	0144
6	Bind time information about SQL statements that involve audited objects.	0145
7	Assignment or change of authorization ID.	0055, 0083, 0087, 0169, 0319
8	Utilities.	0023, 0024, 0025, 0219, 0220
9	Installation-defined audit record.	146
10	Trusted context information.	0269, 0270
11	Audits of successful access.	0361 ¹
12 - 29	Reserved.	
30 - 32	Available for local use.	

Notes:

1. If IFCID 0361 is started through START TRACE, all successful access is traced. If IFCID 0361 is started because audit policy category SYSADMIN is on, only successful access using the SYSADMIN administrative authority is traced. If IFCID 0361 is started because audit policy category DBADMIN is on, only successful access using the DBADMIN administrative authority is traced.

Statistics trace (STAT)

Table 30. Classes for DB2 statistics trace

Class	Description of class	Activated IFCIDs
1	Information about system services, database statistics, statistics for the DBM1 address space, and information about the system parameters that were in effect when the trace was started. This default class is also activated when you omit the CLASS keyword from the START TRACE command when you start the statistics trace.	0001, 0002, 0105, 0106, 0202, 0225
2	Installation-defined statistics record	0152
3	Deadlock, lock escalation, group buffer pool, data set extension information, and indications of long-running uncommitted reads, and active log space shortages.	0172, 0196, 0250, 0258, 0261, 0262, 0313, 0330, 0337
4	DB2 exceptional conditions.	0173, 0191-0195, 0203-0210, 0235, 0236, 0238, 0267, 0268
5	DB2 data sharing statistics record.	0230
6	Storage statistics for the DB2 subsystem.	0225
7	DRDA location statistics.	0365

Table 30. Classes for DB2 statistics trace (continued)

Class	Description of class	Activated IFCIDs
8	Data set I/O statistics.	0199
9	Aggregate CPU and wait time statistics by connection type.	0369
10 - 29	Reserved.	
30 - 32	Available for local use.	

Performance trace (PERFM)

Table 31. Classes for DB2 performance trace

Class	Description of class	Activated IFCIDs
1	Background events. This default class is also activated when you omit the CLASS keyword from the START TRACE command when you start the performance trace.	0001, 0002, 0031, 0042, 0043, 0076-0079, 0102, 0103, 0105-0107, 0153
2	Subsystem events. This default class is also activated when you omit the CLASS keyword from the START TRACE command when you start the performance trace.	0003, 0068-0075, 0080-0089, 0106, 0174, 0175
3	SQL events. This default class is also activated when you omit the CLASS keyword from the START TRACE command when you start the performance trace.	0022, 0053, 0055, 0058-0066, 0092, 0095-0097, 0106, 0112, 0173, 0177, 0233, 0237, 0250, 0272, 0273, 0325
4	Reads to and writes from the buffer and EDM pools.	0006-0010, 0029-0030, 0105-0107, 0127, 0128, 0226, 0227, 0321, 0322
5	Write to log; archive log.	0032-0041, 0104, 0106, 0114-0120, 0228, 0229
6	Summary lock information.	0020, 0044, 0045, 0105-0107, 0172, 0196, 0213, 0214, 0218, 0337
7	Detailed lock information.	0021, 0105-0107, 0223
8	Data scanning detail.	0013-0018, 0105-0107, 0125, 0221, 0222, 0231, 0305, 0311, 0363
9	Sort detail.	0026-0028, 0095-0096, 0106
10	BIND, commands, and utilities detail.	0023-0025, 0090, 0091, 0105-0107, 0108-0111, 0201, 0256
11	Execution unit switch and latch contentions.	0046-0052, 0056, 0057, 0093, 0094, 0106, 0113
12	Storage manager.	0098-0101, 0106
13	Edit and validation exits.	0011, 0012, 0019, 0105-0107
14	Entry from and exit to an application.	0067, 0106, 0121, 0122
15	Installation-defined performance record.	0154
16	Distributed processing.	0157-0163, 0167, 0183
17	Claim and drain information.	211-216
18	Event-based console messages.	0197
19	Data set open and close activity.	0370, 0371
20	Data sharing coherency summary.	0249-0251, 0256-0257, 0261, 0262, 0267, 0268
21	Data sharing coherency detail.	0255, 0259, 0263
22	Authorization exit parameters.	0314

Table 31. Classes for DB2 performance trace (continued)

Class	Description of class	Activated IFCIDs
23	Language environment runtime diagnostics.	0327
24	Stored procedure detail.	0380, 0499
25-29	Reserved.	
30 - 32	Available for local use.	

Monitor trace (MONITOR)

Table 32. Classes for DB2 monitor types

Class	Description of class	Activated IFCIDs
1	Standard accounting data. This default class is also activated when you omit the CLASS keyword from the START TRACE command when you start the monitor trace.	0200
2	Entry or exit from DB2 event signaling.	0232
3	DB2 wait time for I/O, locks; resource usage information.	0006-0009, 0032, 0033,0044, 0045, 0117, 0118, 0127, 0128, 0170, 0171, 0174, 0175, 0213,0 214, 0215, 0216, 0226, 0227, 0242, 0243, 0321, 0322, 0378, 0379
4	Installation-defined monitor record.	0155
5	Time spent processing IFI requests.	0187
6	Changes to tables created with DATA CAPTURE CHANGES.	0185
7	Entry or exit from DB2 event signaling for package accounting. The data traces the amount of time an agent spent in DB2 to process each package.	0200, 0232, 0240
8	Wait time for a package.	0006-0009, 0032, 0033, 0044, 0045, 0051, 0052, 0056, 0057, 0117, 0118, 0127, 0128, 0170,171,174, 175, 213-216, 226, 227, 239, 241-243, 321, 322, 0378, 0379
9	Enables statement level accounting.	0124
10	Package detail for buffer manager, lock manager and SQL statistics. One of the following traces must also be activated before the IFCID 0239 records are written: <ul style="list-style-type: none"> • Accounting class 7 • Accounting class 8 • Monitor class 7 • Monitor class 8 	0239
11-28	Reserved.	
29	Controls the subsystem-wide collection of statistics for SQL statements.	0316, 0318, 0400, 0401
30 - 32	Available for local use.	

IFCID (ifcid , ...)

Specifies which other IFCIDs (trace events), in addition to those IFCIDs contained in the classes specified in the CLASS option, are to be started. To start only those IFCIDs specified in the IFCID option, use trace classes 30-32.

These classes have no predefined IFCIDs and are available for a location to use. See Example 1 for an example of activating only those trace events specified in the IFCID option.

If you do not specify the IFCID option, only those IFCIDs contained in the activated trace classes are started.

The maximum number of IFCIDs is 156. The range of values that are valid for the IFCID option is 0001 through 0511, with the exceptions of: 0004, 0005, 0185, 0187, 0217, 0232, 0234, 0240, 0241, and 0362. These exceptions are invalid values for the IFCID option. IFCIDs 4 and 5 are always automatically active. IFCID 0362 is automatically started if you specify the AUDTPLCY option. Some of the other invalid IFCIDs can be activated only by certain trace classes. The invalid values for the IFCID option that can be started only by trace classes are:

To start...

Start...

IFCID 0185

monitor trace class 6

IFCID 0232

monitor trace class 2 or 7, or accounting trace class 2 or 7

IFCID 0240

monitor trace class 7 or accounting trace 7

IFCID 0241

monitor trace class 8 or accounting trace 8

The **default** is **IFCID(*)**.

BUFSIZE (*k_bytes* , ...)

Specifies the size of an IFC managed buffer that receives the trace data. You can specify this option only if you specified an OP *n* destination.

k_bytes can range from 256 KB to 65536 KB. The number must be evenly divisible by 4. If you specify a value outside of this range, the range limit closest to the specified value is used. To allocate a buffer size of 256 KB, you would specify BUFSIZE(256).

The **default** is **BUFSIZE (*)**, which is the size set when DB2 was installed.

AUDTPLCY(*policy-name* , ...)

Introduces a list of up to eight audit policy names for which trace information is gathered. This option starts the IFCIDs that correspond to the categories that are specified in the listed audit policies, and starts a trace for IFCID 0362.

AUDTPLCY applies to trace type AUDIT. You cannot specify AUDTPLCY with CLASS or IFCID.

TDATA

Specifies the product section headers to be placed into the product section of each trace record. If you do not specify TDATA, then the type of trace determines the type of product section header. The product section of a trace record can contain multiple headers.

All IFC records have a standard IFC header. The correlation header is added for accounting, performance, audit, and monitor records. The trace header is added for serviceability records.

CORRELATION

Places a correlation header on the record.

Abbreviation: COR

TRACE

Places a trace header on the record.

Abbreviation: TRA

CPU

Places a CPU header on the record. The CPU header contains the current processor time for the z/OS TCB or SRB executing.

DISTRIBUTED

Places a distributed header on the record.

Abbreviation: DIST

Usage notes

Audit policies: Up to 32 audit policies can be active concurrently. If you specify multiple audit policies to start, and some of those policies do not start successfully, warning message DSNW196I is returned. The remaining audit policies are started.

Number of traces: If you use one or no values for PLAN, AUTHID, or LOCATION, the START TRACE command starts a single trace. If you use multiple values for PLAN, AUTHID, or LOCATION, the command starts a trace for each plan, authorization ID, or location. Up to 32 traces can be active concurrently. If a START TRACE command is entered from the console or from the DB2I panels to an OP *n* or an OPX destination, message DSNW133I is issued to indicate trace data lost.

Using the options PLAN, AUTHID, or LOCATION when starting monitor trace class 1 has no effect on the amount of data returned on IFI READS requests.

Using the options PLAN, AUTHID, or LOCATION has no effect when starting either accounting or monitor trace classes 2, 5, or 7.

Stopping and starting DB2: If DB2 is stopped and started after you have started a trace, the trace is not restarted automatically.

Specifying SCOPE (GROUP): When you issue START TRACE with SCOPE(GROUP), DB2 issues a START TRACE command on each member of the data sharing group. The data goes to the destination as it is defined for each member of the data sharing group. If you want to gather trace data for all members of the data sharing group in one place, use a monitor program with IFI READA or READS calls to collect the data.

If a trace is started with SCOPE(GROUP), and a new member joins the data sharing group after the trace is started, the new member also writes the trace data that is specified by the START TRACE command.

Starting a trace with SCOPE(GROUP) can generate large amounts of trace data, so you might need to increase the size of the return area in your monitor program to hold the extra data.

*Tracing threads using the * wildcard in a partial name or address:* In a partial name or address, you can use the wildcard suffix, "*" to filter threads. For example, if you specify "-START TRACE PLAN (A,B,C*)", DB2 will trace, and then return A, B, CDE, CDEFG, CDEFGH, and so on. It will trace threads "A", "B" and all threads starting with "C".

You cannot include the wildcard character in the middle of a partial name or address.

Tracing threads using the positional, () wildcard in a partial name or address: In a partial name or address, you can use the positional wildcard, which is represented by the, “_” character, to trace threads when you want the wildcard in the middle, or when you want to trace threads of a specific length. For example, if you specify “-START TRACE PLAN (A_C), all threads will be traced that are three characters that have “A” as the first character, and “C” as the third. This command would return “ABC”, “ADC”, “AEC”, and so on. If you want to trace the thread “A_C” then you can specify “-START TRACE PLAN (A/_C). The “/” before the “_” tells DB2 to trace for the underscore in the search, rather than treating it as a wildcard. The same logic applies if you are trying to trace a thread that includes a “/” or “*” character. Because the character “/” is an escape character, if you are trying to trace a thread that has an “/” character in it, you can specify, for example, “-START TRACE PLAN (A//C)” to trace the thread “A/C”. You can also specify “-START TRACE PLAN (A/*C) to trace the thread “A*C”.

Tracing multiple threads at once using wildcards: You also have the option of tracing multiple threads based on multiple trace qualifications. For example, you can specify, “-START TRACE PLAN (A*, B*, C*) to simultaneously trace ALL threads for plan that start with “A”, “B”, and “C”. The wildcard character, “*” will trace all threads. You can specify more complex combinations such as, “-START TRACE PLAN (A_B*, C*, AND C/_D*)”, which will return all threads that:

- Begin with “A”, have a one character wild card as the second character in the thread, have a “B” as the third character in the thread, and end with any type or number of characters (ADBIOF, AOBTYJDP)
- begin with “C”, and end with any combination of characters (CDE, CGHKO)
- begin with “C_D” and end with any type of character combination (C_DEFGH, C_DLMNOP)

All of the possible thread combinations listed above will be returned with the command above.

Specifying filtering criteria for threads: You can control the set of threads for which trace records are written by setting specific filtering criteria in the -START TRACE command. The following rules apply to **all** trace filters:

- DB2 applies trace filters when an external trace record is written. The state of a thread at that time dictates whether a thread is written in its entirety or eliminated completely.
- Trace filters do not alter the contents of a trace record.
- Trace classes or IFCIDs that are not associated with specific trace records, such as accounting classes 2, 3, 7, 8, and 10, or IFCID 0318, are not affected by filtering. For example, the following command does not result in filtering because no external records are written by accounting classes 7 or 8:
-START TRACE(ACCTG) CLASS(7,8) DEST(SMF) PKGPROG(ABC)
- DB2 allows only one filtering option to have more than one value in a -START TRACE command. For example, the following command is valid:
-START TRACE PLAN(A,B) USERID(B) WRKSTN(E)
The following command is not valid:
-START TRACE PLAN(A,B) USERID(A,B) WRKSTN(E)

Filtering threads using exclude functionality: When you specify an “X” with any constraint keyword, such as XPLAN, when you are filtering threads, you are using

the exclude functionality for the `-START TRACE` command. You have the option of excluding specific types of threads when you are running trace commands. You can use the "X" character to exclude specific combinations of characters when you are running a trace. For example, you can specify this command to trace all threads except "A":

```
-START TRACE XPLAN(A)
```

In this instance, b, bcd, bcde, or cd might be returned.

You can also exclude multiple types of threads from your trace. For example, you can specify this command to start a trace for all threads **except** threads for plans that start with "A", with any combination of characters following "A", and all those characters starting with "B", with any combination of characters following "B":

```
-START TRACE XPLAN(A*, B*)
```

Specifying XPLAN (*) excludes all threads from your search, and is not allowed. You also cannot use the * wildcard in the middle of trace criteria with exclude functionality, such as: `"-START TRACE XPLAN (A*C)."` You can, however, specify this command to return all threads **except** those for plans that start with "A", any **two** characters next, a "C" in the fourth space, and any characters at the end.

```
-START TRACE XPLAN (A_ _ C *)
```

You can start two traces at once, in order to help you optimize your tracing capabilities. For example, you can specify this command:

```
-START TRACE XPLAN (A, B, C) USERID(D)
```

This command tells DB2 to start tracing threads for all plans **except** plans "A", "B", or "C", and only where the user ID = "D".

Combining trace qualifiers: You can customize the threads you trace by commanding DB2 to trace specific threads, while excluding other specific threads. For example, you can specify, `"-START TRACE USERID (A,B) XPLAN (C)"`. This criteria only traces threads where the user ID is equal to "A" or "B", and plan is **not** equal to "C". In this example, a thread where the user id is "A" and the plan is equal to "D" would pass, and be traced, but a thread where the user ID is "A" and plan is "C" would not pass, and would not be traced.

You can introduce multiple wildcards into your start trace commands to add more customization to your traces. For example, you can specify `"-START TRACE PLAN (C*) USERID (Z, X) XPLAN (C, D, E)"`. In this example, for the thread to be traced, the plan must begin with C, the user ID must be equal to Z or to X, and the plan cannot be C, D, or E. So a plan of CB, with a user ID of Z would pass, and the thread would be traced, but plan C with a user ID of X would fail because the command specifies not to trace threads where the plan is "C", without additional characters in the thread.

Trace destination precedence: If an IFCID is associated with a class, and you specify that IFCID in the IFCID keyword, the destination for the class takes precedence. This rule affects IFCIDs in accounting classes 2, 3, 5, 7, 8, and monitoring classes 1, 2, 3, 5, 7, 8 because these classes have preset destinations for the IFCIDs.

For example, the following command `"-START TRACE(ACCTG) CLASS (1,2,3) IFCID(6,7) DEST SMF,"` will not write IFCIDs 6 and 7 to SMF because they are part

of accounting class 3, which has a preset destination. To write IFCIDs 6 and 7 to SMF, you need to start the trace as follows:

```
-START TRACE(ACCTG) CLASS(1,2,3) DEST(SMF)
-START TRACE(ACCTG) CLASS(30) IFCID(6,7) DEST(SMF)
```

Examples

Example: Start a performance trace for threads with remote activity to location USIBMSTODB21. Only activate IFCIDs 0044 (lock suspends) and 0045 (lock resumes). Trace class 30 is available for installation use.

```
-START TRACE (PERFM)
  DEST(GTF)
  LOCATION(USIBMSTODB21)
  CLASS(30)
  IFCID(44,45)
```

Example: Start an accounting trace for plan DSN8BC81. Write records to SMF (that will happen by default). Include a comment to identify the trace.

```
-START TRACE (ACCTG)
  PLAN (DSN8BC81)
  COMMENT ('ACCTG TRACE FOR DSN8BC81')
```

Example: Start the statistics trace. Write records to SMF (by default).

```
-START TRACE=S
```

Example: Start monitor tracing (usually done by an application program). Write records to OPX (by default).

```
-START TRACE(MON)
```

Example: Start monitor tracing (usually done by an application program) on the data sharing group. Write records to OPX (by default).

```
-START TRACE(MON) SCOPE(GROUP)
```

Example: Use the PKGPROG option to write performance trace records only for threads that are executing package ABC during the SQL event that causes the trace record to be externalized.

```
-START TRACE(PERFM) CLASS(3) DEST(SMF) PKGPROG(ABC)
```

Example: Start an accounting trace to activate package-level accounting and to collect data that is externalized by IFCID 0003 and IFCID 0239. Use the PKGPROG option to externalize accounting information that is written when accounting records for package ABC are written.

```
-START TRACE(ACCTG) CLASS(1,2,3,7,8) DEST(SMF) PKGPROG(ABC)
```

The externalized records contain information for **all** packages that are executed during the accounting interval.


Example: Provide IFC exclude filtering for correlation ID.


```
START TRACE (A) XCORRID (*)
- 10.46.05 STC00051 DSNW150I ) EXCLUDE FOR ALL CORRID VALUES IS NOT
- ALLOWED
- 10.46.05 STC00051 DSN9023I ) DSNWVCM1 '-START TRACE' ABNORMAL COMPLETION
```

Example: Start an audit trace using audit policy AUDITADMIN.


```
-STA TRACE(AUDIT) DEST(GTF) AUDTPLCY(AUDITADMIN)
```

Related concepts:

 [DB2 trace \(DB2 Performance\)](#)

 [DB2 trace output \(DB2 Performance\)](#)

Related tasks:

 [Minimizing the processing cost of DB2 traces \(DB2 Performance\)](#)

Related reference:

-STOP TRACE (DB2)

-MODIFY TRACE (DB2)

-DISPLAY TRACE (DB2)

Chapter 82. -STOP ACCEL (DB2)

The DB2 command STOP ACCEL causes the DB2 subsystem to stop using the specified accelerator servers.

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program that uses the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the SCOPE option.

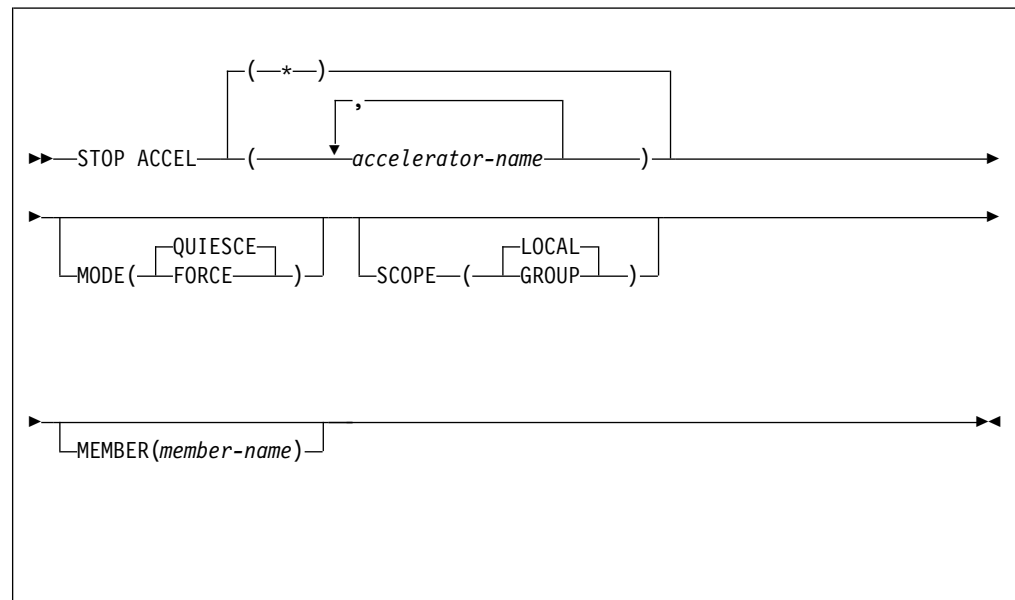
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization by using primary and secondary authorization IDs.

Syntax



Option descriptions

(*accelerator-name*)

The accelerator server name. This option limits the stop command to the specified accelerator server.

(*) Indicates that the stop command applies to all accelerator servers.

MODE

Indicates whether currently executing active accelerator threads are allowed to complete. Valid values are:

(QUIESCE)

Allows active accelerator threads to complete normally and terminates only inactive accelerator threads.

(FORCE)

Terminates all currently executing accelerator threads. DB2 reverts to baseline processing for the originating threads.

SCOPE

Specifies the scope of the command. In a non-data-sharing environment, the option is ignored. Valid values are:

(LOCAL)

Stops the current data sharing member from using the specified accelerator servers. This is the default.

(GROUP)

Stops the data sharing group from using the specified accelerator servers.

MEMBER

Stops specific members of the data sharing group from using the specified accelerator servers.

Usage notes

If you specify both SCOPE(GLOBAL) and MEMBER(*member-name*), the command will be executed only on the specified members of the data sharing group.

Example


The following command is used to stop the current data sharing member from using all accelerators, and to terminate only inactive accelerator threads.

```
-STOP ACCEL(*) MODE(QUIESCE) SCOPE(LOCAL)
```

The output is similar to the following output:

```
DSNX821I ) DSNX8STO STOP ACCELERATOR SUCCESSFUL FOR BLINK1
DSNX821I ) DSNX8STO STOP ACCELERATOR SUCCESSFUL FOR BLINK2
DSNX821I ) DSNX8STO STOP ACCELERATOR SUCCESSFUL FOR BLINK3
DSNX829I ) DSNX8STO ALL ACCELERATORS STOPPED
DSN9022I ) DSNX8CMD '-STOP ACCEL' NORMAL COMPLETION
```

Related reference:

 Support Portal: IBM DB2 Analytics Accelerator for z/OS

Chapter 83. STOP admtproc (z/OS)

The STOP *admtproc* command stops the administrative task scheduler that is specified in the *admtproc* parameter.

The command should only be issued to bring down the administrative task scheduler for maintenance or to prepare for an IPL. To stop the administrative task scheduler for other purposes, issue the command: modify admtproc, appl=shutdown.

Environment


This command can be issued only from a z/OS console.

Data sharing scope: Member

Authorization

The command requires an appropriate level of operating system authority.

Syntax



```
▶—STOP—admtproc—◄◄
```

Option descriptions

admtproc

Specifies the procedure name of the administrative task scheduler task that you want to stop.

Examples

Example: This command stops the *admtproc* scheduler.

Enter the following command on the system console:

```
stop admtproc
```

Chapter 84. /STOP (IMS)

The IMS /STOP command (with the SUBSYS parameter) prevents application programs from accessing external subsystem resources.

The following is only a partial description of the /STOP command.

Environment

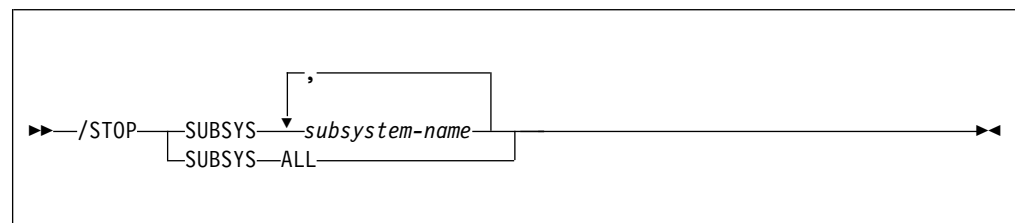
This command can be issued only from an IMS terminal.

Data sharing scope: Member

Authorization

This command requires an appropriate level of IMS authority.

Syntax



Option descriptions

SUBSYS

Specifies whether connection is to be stopped for one or more names of external subsystems presently connected to IMS, or for all of them.

subsystem-name , ...

Specifies one or more names of external subsystems whose connection to IMS is to be stopped.

ALL

Indicates that connection is to be stopped for all external subsystems presently connected to IMS.

Usage note

When to use /STOP: The /STOP command allows application programs currently accessing external resources to complete normally. When all applications have terminated, the connection to the external subsystem is also terminated. A /START command must be issued to reestablish the connection.

The /STOP command can also be used to stop the subsystem connection in order to change the specifications in the external subsystem's PROCLIB member entry. The /START command then refreshes the copy in main storage of the PROCLIB entry with the modified entry.

Chapter 85. -STOP DATABASE (DB2)

The DB2 command STOP DATABASE makes the specified objects unavailable for applications and closes their data sets.

The objects that can be designated are:

- Databases
- Table spaces
- Index spaces
- Physical partitions of partitioned table spaces or index spaces (including index spaces that contains data-partitioned secondary indexes)
- Logical partitions of nonpartitioned secondary indexes

When used to stop a logical partition of a secondary index, the command does not close any data sets that are associated with the index.

In a data sharing environment, the command applies to every member of the data sharing group. If a GBP-dependent object is stopped with the command STOP DATABASE, DB2 performs the necessary processing to make the object no longer GBP-dependent.

Abbreviation: -STO DB

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- STOPDB privilege
- DBMAINT authority
- DBCTRL authority
- DBADM authority
- System DBADM authority
- SYSCTRL authority
- SYSADM authority

Error messages are produced for those specified databases for which this set does not have the STOPDB privilege.

For implicitly created databases, the database privilege or authority can be held on the implicitly created database or on DSNDB04. If the STOP DATABASE command is issued on specific table spaces or index spaces in an implicitly created database, ownership of the table spaces is sufficient to stop them. This means that the owner

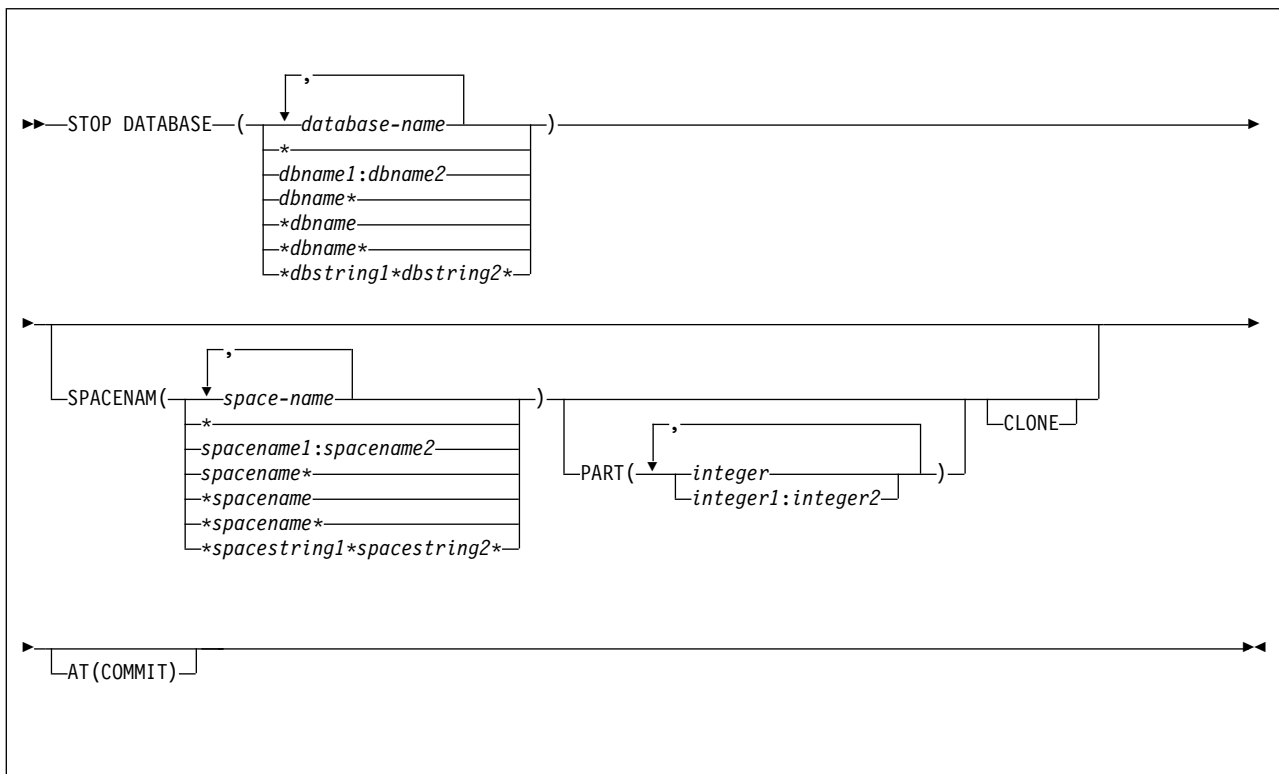
can display information about an implicitly created table space or index space if the command explicitly specifies that table space or index space name.

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

When data definition control is active, installation SYSOPR or installation SYSADM authority is required to stop the database, a table space, or an index space that contains a registration table or index.

Database DSNDB06 contains the table spaces and index spaces that are required to check authorization. If you stop any table space or index space that is required for the START DATABASE authorization check, installation SYSADM authority is required to restart it.

Syntax



Option descriptions

One of the following two options is required.

(*database-name* , ...)

Specifies the names of the database, or database for the table spaces or index spaces to stop. If you use more than one name, separate names in the list by commas.

(*)

Stops all databases for which the privilege set of the process has at least DBMAINT authority or STOPDB privilege.

However, DSNDB01, DSNDB06, and work file databases, such as DSNDB07, can be stopped only by specifying them explicitly (for example, STOP DATABASE(DSNDB01)).

dbname and *dbstring* can have any of the forms in the following list (where *dbname1* and *dbname2* represent any strings of from 1 to 8 characters, and *dbname* represents any string of from 1 to 7 characters):

Form Stops...

dbname1:dbname2

All databases whose names, in UNICODE, collate greater than or equal to *dbname1* and less than or equal to *dbname2*

dbname*

All databases whose names begin with the string *dbname*

***dbname**

All databases whose names end with the string *dbname*

dbname

All databases whose names contain the string *dbname*

dbstring1*dbstring2

All databases whose names contain the strings *dbstring1* and *dbstring2*

SPACENAM(*space-name* , ...)

Indicates names of table spaces or indexes within the specified database to stop.

Abbreviation: SPACE, SP

space-name

Is the name of one or more table spaces or index spaces to stop.

You can write *space-name* like *database-name* to designate:

- The name of a single table space or index space
- A range of names
- A partial name, including a beginning or ending pattern-matching character (*), pattern-matching character between two strings, or any combination of these uses. Consecutive pattern-matching characters (*) are not allowed, and you cannot specify two pattern-matching characters in the middle of a keyword string.

See the following section for instructions on how to start a table space or index space again.

(*)

Stops all table spaces and indexes of the specified database.

spacename and *spacestring* can have any of the forms in the following list (where *spacename1* and *spacename2* represent any strings of from 1 to 8 characters, and *spacename* represents any string of from 1 to 7 characters):

Form Displays the status of...

spacename1:spacename2

All table spaces or index spaces whose names, in UNICODE, collate greater than or equal to *spacename1* and less than or equal to *spacename2*

spacename*

All table spaces or index spaces whose names begin with the string *spacename*

***spacename**

All table spaces or index spaces whose names end with the string *spacename*

spacename

All table spaces or index spaces whose names contain the string *spacename*

spacestring1*spacestring2

All table spaces or index spaces whose names contain the strings *spacestring1* and *spacestring2*

PART (*integer* , ...)

Indicates the partition number of one or more partitions, within the specified table space or index, that are to be stopped. The START or STOP state of other partitions does not change.

The *integer* specified must identify a valid partition number for the corresponding space name and database name. If you specify nonvalid partition numbers, you receive an error message for each nonvalid number, but all valid partitions that you specified are stopped.

integer can be written to designate one of the following specifications:

- A list of one or more partitions
- A range of all partition numbers that collate greater than or equal to *integer1* and less than or equal to *integer2*
- A combination of lists and ranges

PART is valid with partitioned table spaces, partitioned indexes, and nonpartitioned type 2 indexes of partitioned table spaces. If you specify PART with a nonpartitioned table space or index on a nonpartitioned table space, you receive an error message, and the nonpartitioned space is not stopped. When a logical partition is stopped, the index is not closed. A nonpartitioning index must be stopped without the use of PART to close the index.

CLONE

Stops clone objects. In the absence of the CLONE keyword, base table objects are stopped and clone table objects are not processed. If you specify the CLONE keyword then only clone objects will be processed.

AT(COMMIT)

Marks the specified object as being in STOP status to prevent access from new requesters. Currently running applications are allowed to continue access until their next commit. After commit, further access by the committing application is prohibited. The object is actually stopped and put in STOP status when all jobs release their claims on it and all utilities release their drain locks on it. Specify AT(COMMIT) to break in on threads that are bound with RELEASE(DEALLOCATE), especially in situations where there is high thread reuse.

The option is ignored for declared temporary databases and table spaces within it.

Usage notes

Explicitly stopped databases: If table spaces and indexes are stopped explicitly (using the STOP DATABASE command with the SPACENAM option), they must be started explicitly using the START DATABASE command. Starting the database does not start table spaces or indexes that have been stopped explicitly.

Stopped table spaces, indexes, and partitions: Table spaces, indexes, and partitions are physically closed when the STOP DATABASE command is issued, except for logical partitions of a nonpartitioning index of a partitioned table space. Index spaces for declared temporary tables cannot be stopped or started.

Operation in TSO, z/OS, and batch: When the STOP DATABASE command is issued from a TSO or a z/OS console, the command operates asynchronously to keep the terminal free. When the command is issued from a batch job, it operates synchronously in case later steps depend on the database being stopped. The STOP DATABASE command drains work in progress on the database before stopping it. If it cannot get the drain locks on the first request, it repeatedly tries again. The command fails if it times out more than 15 times trying to get the locks or if a serious deadlock situation occurs.

Ensuring that all databases are stopped: When the STOP DATABASE command is processing asynchronously, message DSN9022I might be issued before the command completes. Message DSNT736I is issued to indicate that the asynchronous processing of the STOP DATABASE command is complete.

Use the DISPLAY DATABASE command to check the stopped status of table spaces and indexes in a database. A status of STOPP indicates that the object is in the process of being stopped. A status of STOP indicates that the stop has completed and the object is in a stopped state. An object is not stopped until all currently active threads accessing the object are quiesced.

An object might remain in the STOP pending (STOPP) status if the STOP DATABASE command does not successfully complete processing.

Stopping the communication database and the resource limit database: If the communication database (CDB) and the resource limit database (RLST) are active, they cannot be stopped. Those databases are active when created and are activated by DB2.

| **Stopping the SYSCONTX catalog table space or indexes on tables in the**
| **SYSCONTX catalog table space:** If trusted contexts are in use when you stop
| SYSCONTX or the associated indexes, you can continue to use any trusted contexts
| that are already defined.

Stopping DSNDB01: If you try to stop the DSNDB01 database while an application plan or package is executing, you might receive a time out because of locking contention on DSNDB01. This is most likely to occur when an application plan or package is executing for the first time since DB2 was started, or if the skeleton cursor table (SKCT) for the plan or the skeleton package table (SKPT) for the package was swapped out of the EDM pool.

Table space in a restrictive status: If an application process requests a transaction lock on a table space that is in a restrictive status (RECP) or has a required index in a restrictive status, DB2 acquires the lock and does not detect the status until the application tries to access the table space or index. The application then receives

SQLCODE -904 (“resource not available”) and should release the lock, either by committing or rolling back (if the value of the RELEASE option is COMMIT) or by ending (if the value of RELEASE is DEALLOCATE). If you issue the command STOP DATABASE for either the table space or the index space while a transaction lock is in effect, the command is suspended. It repeatedly tries to get the locks needed to drain the work in progress before stopping the database. If the command times out more than 15 times trying to get the locks, it fails.

Stopping a table space partition in PRO restricted status: When a table space that is in Persistent Read Only (PRO) restricted status is stopped, the partition remains in PRO restricted status.

After a disk failure: Issuing the STOP DATABASE command before interrupting the I/O interface between the failed device and DB2 can result in incomplete I/O requests. To prevent this hang situation, create an interruption either by forcing the device offline using the z/OS command VARY with the FORCE option, or by setting the I/O timing interval for the device before any failures. You can set the I/O timing interval through the IECIOSxx z/OS parmlib member or by issuing the z/OS command:

```
SETIOS MIH,DEV=dddd,IOTIMING=mm:ss
```

Stopping a LOB table space: The STOP DATABASE command can be used to stop LOB table spaces and indexes on auxiliary tables. LOB table spaces are stopped independently of the base table space with which the LOB table space is associated.

The following table summarizes the locking used by the STOP DATABASE command.

Table 33. Locking used by the STOP DATABASE command

Command	Table space type		Locks acquired
STOP AT COMMIT	Partitioned	PART	IX mass delete lock. Drain-all on partitions specified.
			IX mass delete lock. Drain-all on all partitions.
	Nonpartitioned		IX mass delete lock. Drain-all on table space.
STOP	Partitioned	PART	X-lock partitions specified. Drain-all on partitions specified.
			X-lock all partitions. Drain-all on all partitions.
	Nonpartitioned		X-lock table space. Drain-all on table space.

Examples

Example 1: Stop table space DSN8S81E in database DSN8D81A and close the data sets that belong to that table space.

```
-STOP DATABASE(DSN8D81A) SPACENAM(DSN8S81E)
```

Example 2: Stop all databases (except DSNDB01, DSNDB06, and work file databases)

```
-STOP DATABASE(*)
```

Example 3: Stop all databases (except DSNDB01, DSNDB06, and work file databases) when all jobs release their claims and all utilities release their drain locks.

```
-STOP DATABASE(*) AT(COMMIT)
```

Example 4: Stop the first partition of XEMP2, a nonpartitioning index of a partitioned table space in database DSN8D81A. Partition 1 is logically stopped and cannot be accessed by applications; however, no data sets are closed because parts of a nonpartitioning index are not associated with separate physical data sets.

```
-STOP DATABASE(DSN8D81A) SPACENAM(XEMP2) PART(1)
```

Example 5: Stop all table spaces with names that begin with "T" and end with the "IQUA03" string in database DSN8D81A.

```
-STOP DATABASE(DSN8D81A) SPACENAM(T*IQUA03)
```

Output similar to the following output indicates that the command completed successfully:

```
DSN9022I - DSNTDDIS 'STOP DATABASE' NORMAL COMPLETION  
DSNT736I - ASYNCHRONOUS STOP DATABASE COMMAND HAS  
COMPLETED FOR COMMAND: STOP DB(DSN8D81A) SPACE(T*IQUA03)
```

Example 6: Stop clone objects.

```
-STOP DATABASE(MYDB*) SPACENAM(MYDB*SP) CLONE
```

Chapter 86. -STOP DB2 (DB2)

The DB2 command STOP DB2 stops the DB2 subsystem.

Abbreviation: -STO DB2

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

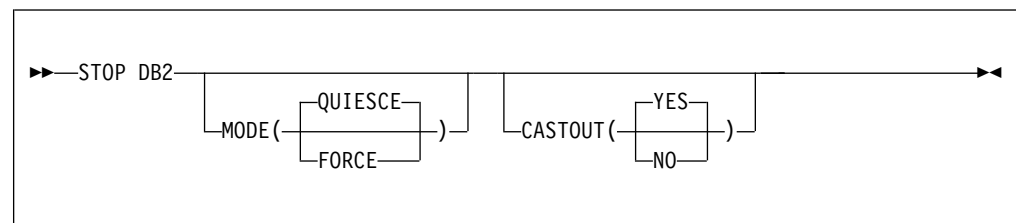
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

- STOPALL privilege
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

MODE

Indicates whether currently executing programs will be allowed to complete. For the effects of this option on distributed threads see the description of the MODE option of -STOP DDF (DB2).

(QUIESCE)

Allows currently executing programs to complete processing. No new program is allowed to start. If a utility starts a subtask after -STOP DB2 MODE (QUIESCE) has been issued, message DSNU006I is issued and the subtask and utility might end abnormally.

(FORCE)

Terminates currently executing programs, including utilities. No new program is allowed to start. MODE(FORCE) probably causes indoubt situations. Some tasks, such as stored procedures tasks and DB2 service

tasks, terminate abnormally. When they terminate abnormally, you might see dumps and messages from these failures.

CASTOUT

Specifies whether the DB2 member performs castout processing for the page sets or partitions for which the member was last updated. The CASTOUT option only applies in a data sharing environment.

YES

Allow group buffer pool castout processing.

NO Skip group buffer pool castout processing.

Usage notes

MODE(QUIESCE): If MODE(QUIESCE) is used, all connected address spaces must terminate all connections before the DB2 subsystem stops. The system operator can tell whether any connections remain by using the DISPLAY THREAD command, and can cancel them by using the DB2 CANCEL command or z/OS commands.

MODE(FORCE): A forced stop does not cause an immediate abend. If a connected task is executing outside DB2, DB2 posts an exit routine to stop the task from accessing DB2. If a task is executing in DB2, it stops when the next “suspend” or “execution unit switch” occurs. In some cases, the delay before stopping can be significant.

CASTOUT(NO): Consider using CASTOUT(NO) when shutting down a DB2 data sharing member for maintenance, because the option can speed shutdown processing in a data sharing environment. If you are shutting down multiple members of a data sharing group with CASTOUT(NO), some changed data might reside in the group buffer pools after the members have shut down. Therefore, if you want consistent data on disk (for example, you are shutting down all members to create a copy of the database to send offsite), do not use CASTOUT(NO).

With CASTOUT(NO), the DB2 member shuts down with QC status, as displayed by the DISPLAY GROUP command, which indicates that the member quiesced with some castout processing not completed. A retained page set or partition P-lock is held in IX state for each object for which the DB2 member was the last updater. Also, group buffer pool connections enter a failed-persistent state.

Example

Example 1: Stop the DB2 subsystem. Allow currently active programs to complete. Do not allow new programs to identify to DB2.

```
-STOP DB2 MODE (QUIESCE)
```

Example 2: Stop a member of a data sharing group for maintenance.

```
-STOP DB2 MODE (QUIESCE) CASTOUT(NO)
```

Chapter 87. -STOP DDF (DB2)

The DB2 command STOP DDF stops the distributed data facility (DDF) if it has already been started; use this command to terminate the DDF interface to VTAM or TCP/IP.

Abbreviation: -STO DDF

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

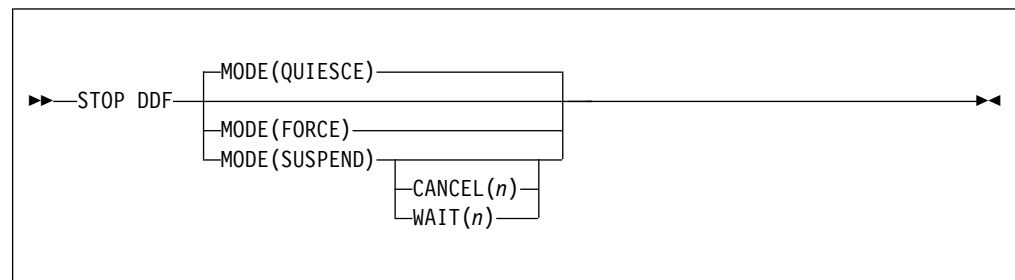
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SYSOPR authority
- SYSTRM authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

MODE

Indicates whether currently executing active distributed threads are allowed to complete.

(QUIESCE)

Allows active distributed threads that are using DDF to complete normally and terminates only inactive distributed threads. If DDF THREADS ACTIVE was specified during DB2 installation, all DDF threads are active threads.

(FORCE)

Terminates all currently executing distributed threads.

Some tasks, such as stored procedures tasks and DB2 service tasks, terminate abnormally. When they terminate abnormally, you might see dumps and messages resulting from these failures.

(SUSPEND)

Suspends all DDF threads by:

- Keeping inactive DDF threads inactive until a subsequent START DDF command is issued
- Terminating all DDF pool threads
- Preventing inbound DDF work from starting

MODE(SUSPEND) is intended to be used at a DB2 DRDA server when locking conflicts exist between CREATE, ALTER, DROP, GRANT, or REVOKE operations and client access to data. Requests that normally cause work to be dispatched (including requests for new connections) are queued. Outbound DDF processing is not affected by this command.

CANCEL (n)

Cancels all active DDF database access threads if suspend processing does not complete in *n* seconds. The range of *n* is 0 to 9999.

WAIT (n)

Resumes DDF processing if suspend processing does not complete in *n* seconds. The range of *n* is 0 to 9999.

Usage notes

MODE(QUIESCE): If MODE(QUIESCE) is used, all distributed activity must complete before DDF stops. The operator can tell whether any distributed threads remain by using DISPLAY THREAD with the LOCATION option. To cancel distributed threads that are preventing DDF from stopping, see for CANCEL THREAD, or use STOP DDF MODE(FORCE).

MODE(QUIESCE) forces any inactive threads to terminate. A requesting system that is using two-phase commit on an inactive thread might report the terminated thread as indoubt at the system that issued STOP DDF. The thread is not actually indoubt (no commit or rollback is pending), and the condition is resolved when DDF is restarted.

MODE(FORCE): If MODE(FORCE) is used, the DB2 connection to VTAM or TCP/IP terminates. The termination forces all VTAM or TCP/IP requests to complete immediately, indicating that a communications error has occurred and DDF has stopped. A forced stop might take as long as three minutes to complete.

If any applications are updating remote servers that use two-phase commit, MODE(FORCE) might result in indoubt threads at each server.

MODE(SUSPEND): If MODE(SUSPEND) completes successfully, additional database resources, which are not inbound DDF work, might still be held. Cancel these additional resources with CANCEL THREAD as described in .

The following table summarizes the actions that DB2 takes when START DDF, STOP DDF, START DB2, and STOP DB2 commands are issued with different DDF states.

Table 34. The result of commands on the DDF status

DDF status	START DDF command	STOP DB2 or STOP DDF command without MODE(FORCE)	STOP DB2 or STOP DDF command with MODE(FORCE)	STOP DDF command with MODE(SUSPEND)
Starting	DSNL003I	DSNL003I	DSNL003I	DSNL003I
Started	DSNL001I	DDF stops	DDF forced stop	DDF suspends
Stopping	DSNL005I	DSNL005I	DSNL005I	DSNL005I
Stopped	DDF starts	DSNL002I	DSNL002I	DSNL002I
Suspending	DDF resumes	DDF stops	DDF forced stop	DSNL069I
Suspended	DDF resumes	DDF stops	DDF forced stop	DSNL065I

Examples

Example 1: Stop the distributed data facility (MODE QUIESCE).

```
-STOP DDF
```

Example 2: Stop the distributed data facility (MODE FORCE).

```
-STOP DDF MODE(FORCE)
```

Example 3: Suspend distributed data facility activity (MODE SUSPEND). If command processing continues after 600 seconds, cancel any remaining DDF threads.

```
-STOP DDF MODE(SUSPEND) CANCEL(600)
```

Chapter 88. -STOP FUNCTION SPECIFIC (DB2)

The DB2 command STOP FUNCTION SPECIFIC prevents DB2 from accepting SQL statements with invocations of the specified functions.

This command does not prevent SQL statements with invocations of the functions from running if they have already been queued or scheduled by DB2. You cannot use this command to stop built-in functions or user-defined functions that are sourced on another function.

DB2 implicitly issues the command STOP FUNCTION SPECIFIC ACTION(REJECT) for any function that exceeds the maximum abend count. That count is set by the MAX ABEND COUNT field of installation panel DSNTIPX.

Abbreviation: -STO FUNC SPEC

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the value of the SCOPE option.

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities for each function:

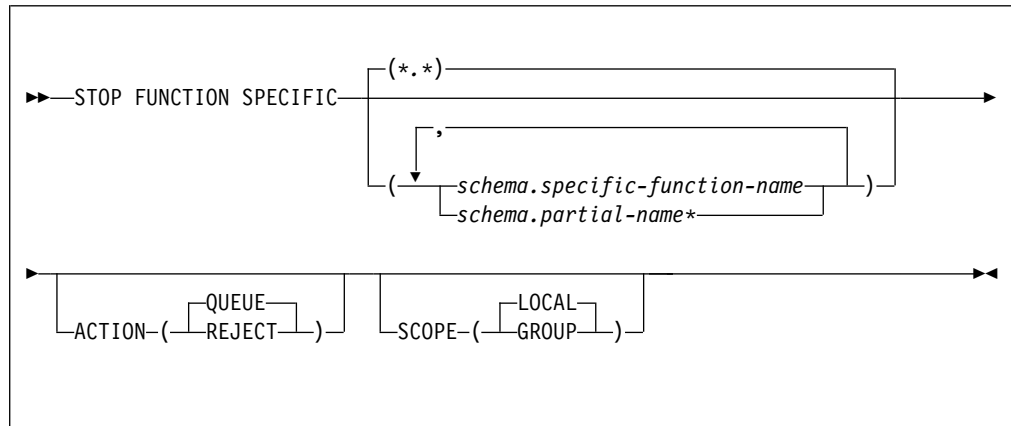
- Ownership of the function
- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

If you specify STOP FUNCTION SPECIFIC *.* or *schema.partial-name **, the privilege set of the process must include one of the following authorities:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

(*.*)

Stops access to all functions, including functions that DB2 applications have not yet accessed.

If no functions are named, all functions are stopped.

schema.specific-function-name

Stops one specific function name. You cannot specify a function name as you can in SQL; you must use the specific name. If a specific name was not specified on the CREATE FUNCTION statement, query SYSIBM.SYSROUTINES for the correct specific name:

```
SELECT SPECIFICNAME, PARM_COUNT
FROM SYSIBM.SYSROUTINES
WHERE NAME='function_name'
AND SCHEMA='schema_name';
```

For overloaded functions, this query can return multiple rows.

*schema.partial-name **

Stops a set of functions in the specified schema. The specific names of all functions in the set begin with *partial-name* and can end with any string, including the empty string. For example, *schema1.ABC** stops all functions with specific names that begin with ABC in schema1.

ACTION

Indicates what to do with an SQL statement that invokes the function while the function is stopped. If you issue STOP FUNCTION SPECIFIC more than once for a given function, the action that is taken is determined by the ACTION option on the most recent command.

(QUEUE)

Queues the request until either of the following conditions is true:

- The wait exceeds the installation timeout value.
- You issue START FUNCTION SPECIFIC command for the function.

(REJECT)

Rejects the request.

SCOPE

Specifies the scope of the command.

(LOCAL)

Specify to stop the function on the local member only.

(GROUP)

Specify to stop the function on all members of the data sharing group.

Usage notes

Limitations of STOP FUNCTION SPECIFIC : STOP FUNCTION SPECIFIC is only applicable to external functions that run in the WLM application environment or non-inline SQL functions. STOP FUNCTION SPECIFIC cannot stop a built-in function, an inline SQL function, or a user-defined function that is sourced on another function.

Permanently disabling a function : A stopped function does not remain stopped if DB2 is stopped and restarted. To disable a function permanently, you can:

- Use ALTER FUNCTION to change the LOADMOD name to a nonexistent z/OS load module
- Rename or delete the z/OS load module

Considerations for SQL functions: The STOP FUNCTION SPECIFIC command affects all versions of the SQL functions that you specify in the command.

Examples

Example 1: Stop access to all functions. While the STOP FUNCTION SPECIFIC command is in effect, DB2 queues all attempts to execute functions.

```
-STOP FUNCTION SPECIFIC ACTION(QUEUE)
```

This command produces output similar to the following output:

```
DSNX974I - DSNX9SP2 STOP FUNCTION SPECIFIC SUCCESSFUL FOR *.*
DSN9022I - DSNX9COM '-STOP FUNC' NORMAL COMPLETION
```

Example 2: Stop access to all functions. While the STOP FUNCTION SPECIFIC command is in effect, DB2 rejects attempts to execute functions.

```
-STOP FUNCTION SPECIFIC ACTION(REJECT)
```

This command produces output similar to the following output:

```
DSNX974I - DSNX9SP2 STOP FUNCTION SPECIFIC SUCCESSFUL FOR *.*
DSN9022I - DSNX9COM '-STOP FUNC' NORMAL COMPLETION
```

Example 3: Stop functions PAYROLL.USERFN1 and PAYROLL.USERFN3. While the STOP FUNCTION SPECIFIC command is in effect, DB2 queues all attempts to execute functions.

```
-STOP FUNCTION SPECIFIC(PAYROLL.USERFN1,PAYROLL.USERFN3)
```

This command produces output similar to the following output:

```
DSNX974I - DSNX9SP2 STOP FUNCTION SPECIFIC SUCCESSFUL
FOR PAYROLL.USERFN1
```

```
DSNX974I - DSNX9SP2 STOP FUNCTION SPECIFIC SUCCESSFUL
FOR PAYROLL.USERFN3
```

```
DSN9022I - DSNX9COM '-STOP FUNC' NORMAL COMPLETION
```

Example 4: Stop functions PAYROLL.USERFN1 and PAYROLL.USERFN3. While the STOP FUNCTION SPECIFIC command is in effect, DB2 rejects attempts to execute either of these functions.

```
-STOP FUNCTION SPECIFIC(PAYROLL.USERFN1,PAYROLL.USERFN3) ACTION(REJECT)
```

This command produces output similar to the following output:

```
DSNX974I - DSNX9SP2 STOP FUNCTION SPECIFIC SUCCESSFUL  
FOR PAYROLL.USERFN1
```

```
DSNX974I - DSNX9SP2 STOP FUNCTION SPECIFIC SUCCESSFUL  
FOR PAYROLL.USERFN3
```

```
DSN9022I - DSNX9COM '-STOP FUNC' NORMAL COMPLETION
```

Chapter 89. STOP irlmproc (z/OS IRLM)

The STOP *irlmproc* command shuts IRLM down normally. The command is rejected if any active DB2 subsystems are currently identified to IRLM.

Abbreviation: P

Environment

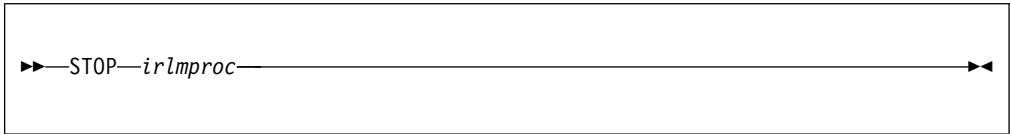
This command can be issued only from a z/OS console.

Data sharing scope: Member

Authorization

The command requires an appropriate level of operating system authority.

Syntax



```
▶▶—STOP—irlmproc—————▶▶
```

Option description

irlmproc

Identifies the procedure name for the IRLM to be stopped.

Usage note

Terminating the IRLM: If IRLM does not shut down normally, issue the MODIFY *irlmproc*,ABEND command to terminate the IRLM abnormally:

```
F irlmproc,ABEND,DUMP
```

If outstanding DB2 requests are in process and IRLM does not terminate, use the z/OS CANCEL command. If all other means of removing the subsystem fail, issue the z/OS FORCE CANCEL command.

Example

Enter the following command on the system console:

```
P KRLM1
```

IRLM outputs the following responses on system console:

```
DXR165I IR21 TERMINATED VIA IRLM MODIFY COMMAND  
DXR121I IR21 END-OF-TASK CLEANUP SUCCESSFUL - HI-CSA      325K
```

In a data sharing environment: You cannot issue the STOP *irlmproc* command to IRLM in a data sharing group until no DB2 subsystems are identified to that IRLM and the IRLM issues the following messages:

```
DXR136I IR21 HAS DISCONNECTED FROM THE DATA SHARING GROUP
```

Any members that are still active in the group issue:

```
DXR137I JR21 GROUP STATUS CHANGED. IR21 233 HAS BEEN DISCONNECTED  
FROM THE DATA SHARING GROUP
```

Chapter 90. -STOP PROCEDURE (DB2)

The DB2 command STOP PROCEDURE prevents DB2 from accepting SQL CALL statements for one or more stored procedures.

You can qualify stored procedure names with a schema name. This command does not prevent CALL statements from running if they have already been queued or scheduled by DB2.

DB2 implicitly issues the command STOP PROCEDURE ACTION(REJECT) for any stored procedure that exceeds the maximum abend count. That count is set by the MAX ABEND COUNT field of installation panel DSNTIPX.

Abbreviation: -STO PROC

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the value of the SCOPE option.

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

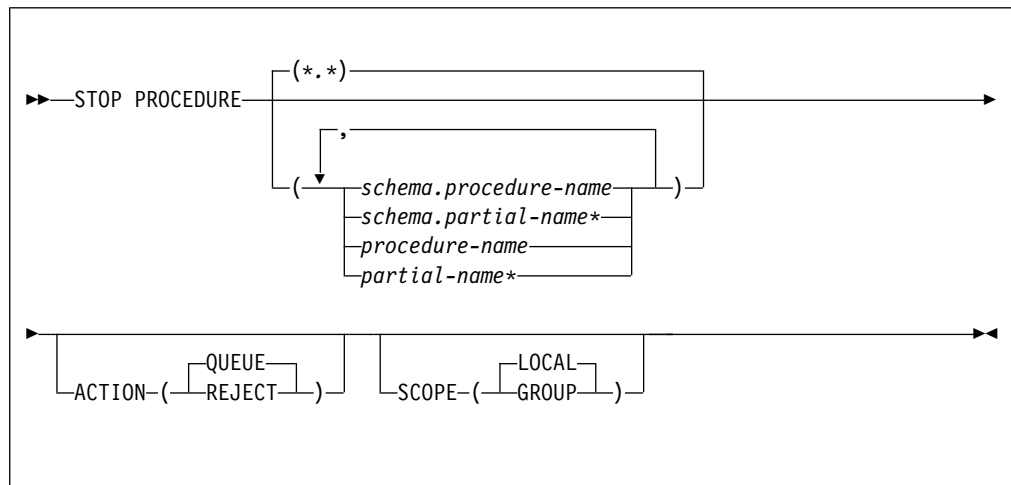
- Ownership of the stored procedure
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

If you specify STOP PROCEDURE *.* or *schema.partial-name **, the privilege set of the process must include one of the following authorities:

- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax



Option descriptions

(*.*)

Stops access to all stored procedures in all schemas, including procedure definitions that have not yet been accessed by DB2 applications.

(schema.procedure-name)

Identifies the fully-qualified procedure name that is to be stopped.

(schema.partial-name *)

Stops a set of stored procedures in the specified schema. The names of all procedures in the set begin with *partial-name* and can end with any string, including the empty string. For example, PAYROLL.* stops all stored procedures in the PAYROLL schema.

procedure-name

Identifies one or more specific stored procedure names to be stopped. The procedure name is implicitly qualified with the SYSPROC schema name.

*partial-name **

Stops a set of stored procedures within the SYSPROC schema. The names of all procedures in the set begin with *partial-name* and can end with any string, including the empty string. For example, ABC* stops all stored procedures with names that begin with ABC.

ACTION

Indicates what to do with a CALL statement that is received while the procedure is stopped. If STOP PROCEDURE is issued more than once for a given procedure, the action taken is determined by the ACTION option on the most recent command.

(QUEUE)

Queues the request until either:

- The wait exceeds the installation timeout value, or
- The stored procedure is started by the command START PROCEDURE.

(REJECT)

Rejects the request

SCOPE

Specifies the scope of the command.

(LOCAL)

Specify to stop the procedure on the local member only.

(GROUP)

Specify to stop the procedure on all members of the data sharing group.

Usage notes

Permanently disabling a stored procedure: A stopped procedure does not remain stopped if DB2 is stopped and restarted. To disable a stored procedure permanently, you can:

- Drop the procedure using the DROP PROCEDURE statement.
- Use an ALTER PROCEDURE statement.
- Rename or delete the z/OS load module.

Considerations for native SQL procedures: The STOP PROCEDURE command affects all versions of the native SQL procedures that you specify in the command.

Examples

Example 1: Stop access to all stored procedures, and terminate the DB2 stored procedures address space. While the STOP PROCEDURE command is in effect, attempts to execute stored procedures are queued.

```
-STOP PROCEDURE ACTION(QUEUE)
```

```
DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR *.*
```

```
DSN9022I - DSNX9COM '-STOP PROC' NORMAL COMPLETION
```

Example 2: Stop access to all stored procedures, and terminate the DB2 stored procedures address space. While the STOP PROCEDURE command is in effect, attempts to execute stored procedures are rejected.

```
-STOP PROCEDURE ACTION(REJECT)
```

```
DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR *.*
```

```
DSN9022I - DSNX9COM '-STOP PROC' NORMAL COMPLETION
```

Example 3: Stop stored procedures USERPRC1 and USERPRC3. While the STOP PROCEDURE command is in effect, attempts to execute these stored procedure are queued.

```
-STOP PROCEDURE(USERPRC1,USERPRC3)
```

```
DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR USERPRC1
```

```
DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR USERPRC3
```

```
DSN9022I - DSNX9COM '-STOP PROC' NORMAL COMPLETION
```

Example 4: Stop stored procedures USERPRC1 and USERPRC3. While the STOP PROCEDURE command is in effect, attempts to execute these stored procedure are rejected.

```
-STOP PROCEDURE(USERPRC1,USERPRC3) ACTION(REJECT)
```

```
DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR USERPRC1
```

```
DSNX947I - DSNX9SP2 STOP PROCEDURE SUCCESSFUL FOR USERPRC3
```

```
DSN9022I - DSNX9COM '-STOP PROC' NORMAL COMPLETION
```

Chapter 91. -STOP PROFILE (DB2)

The DB2 command STOP PROFILE is used to stop or disable the profile function.

Environment

This command can be issued from the z/OS console, through a batch job or the instrumentation facility interface (IFI).

Data sharing scope: Member

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SQLADM authority
- System DBADM authority
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

Syntax

▶—STOP PROFILE—▶

Examples

Example 1: This command is used to stop or disable the profile function.

```
-STOP PROFILE
```

Related concepts:

➡ Profiles for monitoring and controlling DB2 for z/OS subsystems (DB2 Performance)

Related tasks:

➡ Creating profiles (DB2 Performance)

Related reference:

-START PROFILE (DB2)

➡ Profile tables (DB2 Performance)

Chapter 92. -STOP RLIMIT (DB2)

The DB2 command STOP RLIMIT stops the resource limit facility.

All previously limited SQL statements run without limits. However, the processing time continues to accumulate for statements that continue to run after the STOP RLIMIT command is issued.

Abbreviation: -STO RLIM

Environment

This command can be issued from a z/OS console, a DSN session under TSO, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Member

Authorization

To execute this command, you must use a privilege set of the process that includes one of the following authorities:

- SYSOPR authority
- SYSCtrl authority
- SYSADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax

```
▶▶—STOP RLIMIT—————▶▶
```

Example

Stop the resource limit facility.

```
-STOP RLIMIT
```

Related tasks:

[Starting and stopping resource limit tables \(DB2 Performance\)](#)

[Specifying and changing resource limits \(DB2 Performance\)](#)

Related reference:

-START RLIMIT (DB2)

-DISPLAY RLIMIT (DB2)

Chapter 93. -STOP TRACE (DB2)

The DB2 command STOP TRACE stops tracing.

One additional option to this command and additional values for a few other options exist. This additional information is intended for service and use under the direction of IBM Software Support.

Abbreviation: -STO TRA

Environment

This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group or local, depending on the value of the SCOPE option.

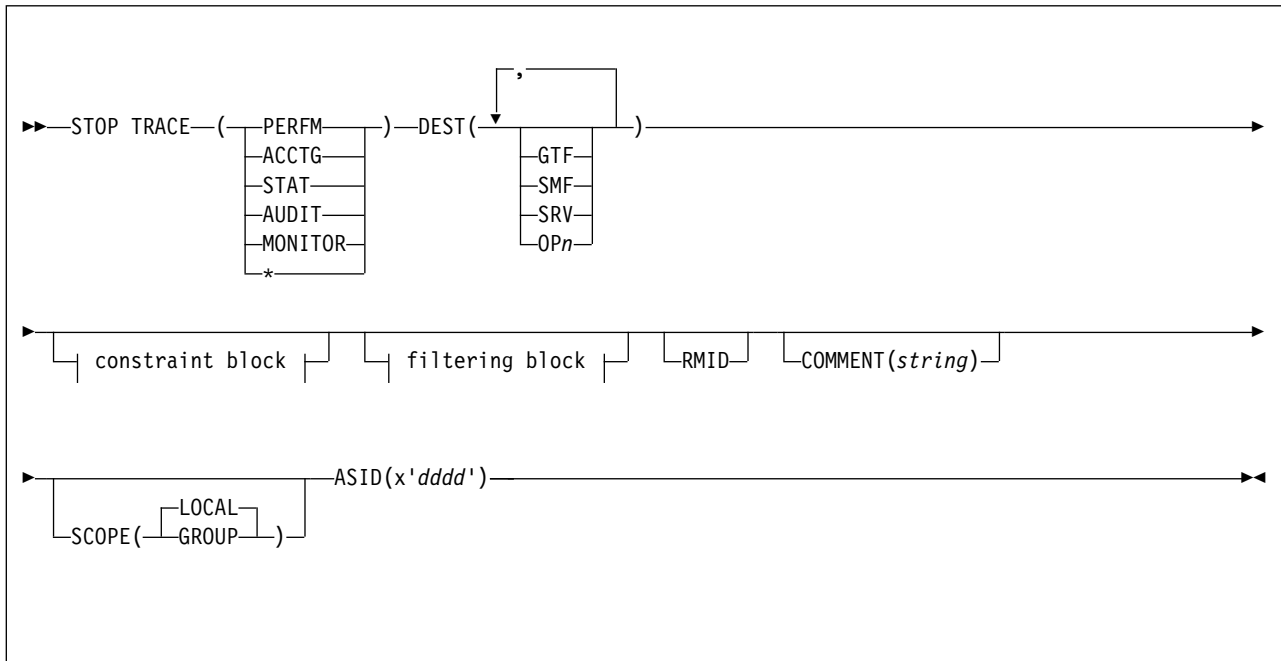
Authorization

To execute this command, you must use a privilege set of the process that includes one of the following privileges or authorities:

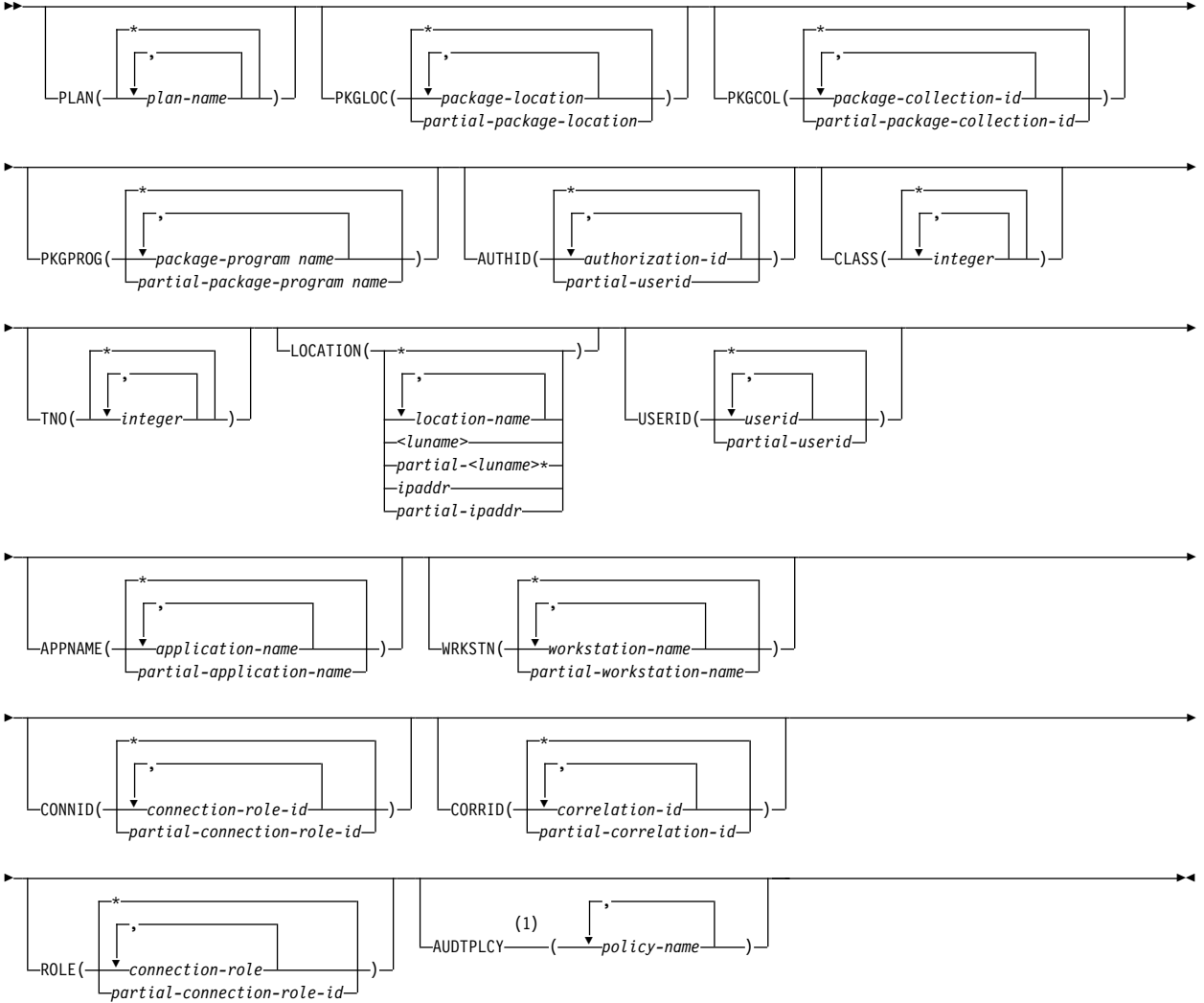
- TRACE privilege
- SQLADM authority
- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority
- SECADM authority

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

Syntax

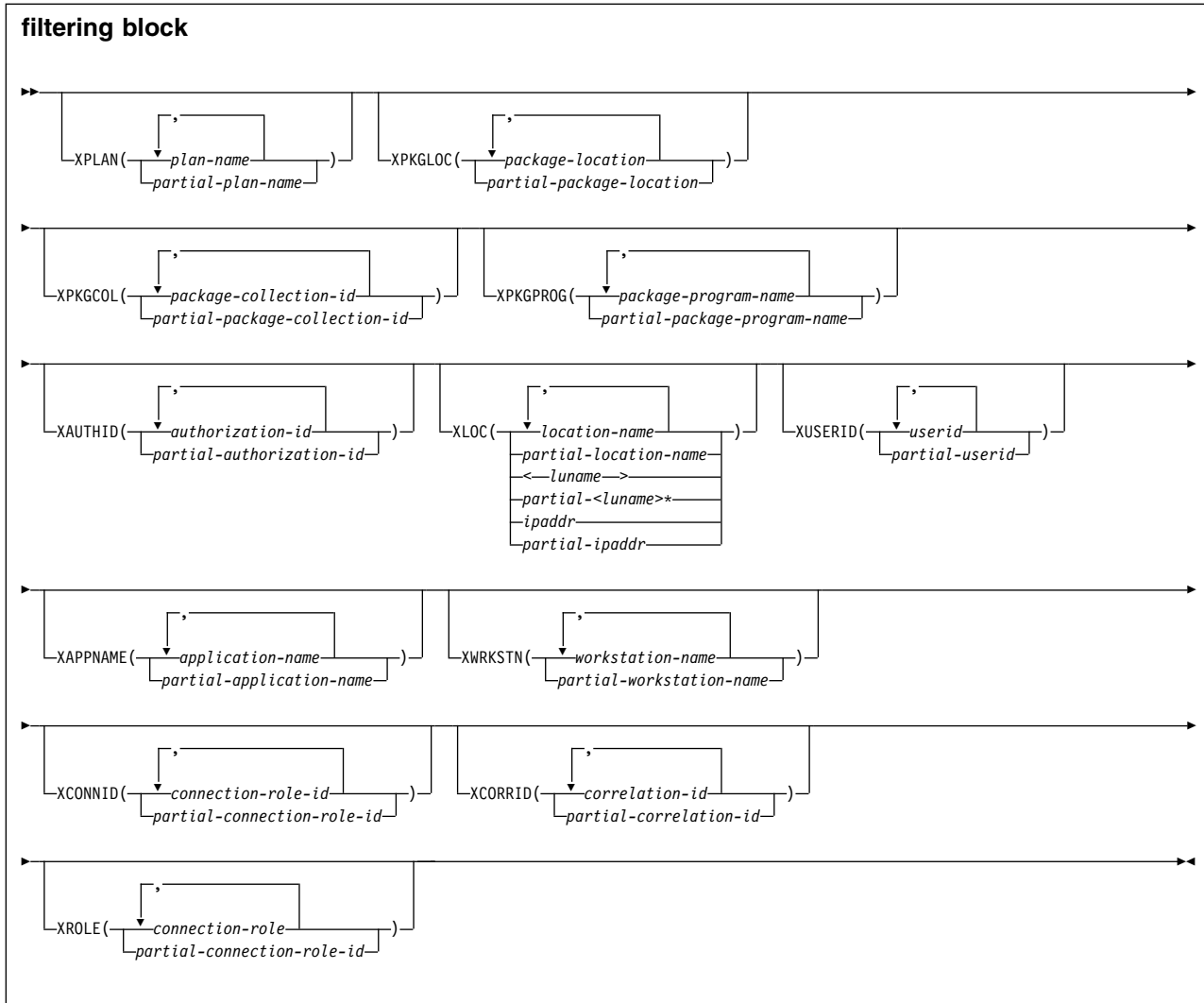


constraint block



Notes:

- 1 You cannot specify CLASS or IFCID with AUDTPLCY. AUDTPLCY applies to trace type AUDIT.



Option descriptions

For additional descriptions of each of the following trace types, see Chapter 81, “-START TRACE (DB2),” on page 459.

(PERFM)

Specify to stop a trace that is intended for performance analysis and tuning.

Abbreviation: P

(ACCTG)

Specify to stop an accounting trace.

Abbreviation: A

(STAT)

Specify to stop a trace that collects statistical data. The LOCATION option cannot be specified when you choose a statistics trace.

Abbreviation: S

(AUDIT)

Specify to stop a trace that collects audit data from various components of DB2.

Abbreviation: AU

(MONITOR)

Specify to stop a trace that collects monitor data.

Abbreviation: MON

(*)

Specify to stop all trace activity. See “Usage notes” on page 518 for information about using STOP TRACE (*) with traces that use monitor trace class 6.

SCOPE

Specifies the scope of the command.

(LOCAL)

Stops the trace only on the local DB2 subsystem.

(GROUP)

Stops the trace on all members of a data sharing group.

ASID(x'ddd')

Specifies that the trace for an address space is to be stopped.

ddd is a four-byte hexadecimal address space ID (ASID).

RMID

Specifies resource manager identifier. You can specify up to 8 valid RMIDs, which are one or two digit identifiers. You cannot specify RMID for accounting or statistic traces.

COMMENT (string)

Gives a comment that is reproduced in the trace output record for the STOP TRACE command (except in the resident trace tables).

string is any SQL string; it must be enclosed between apostrophes if it includes a blank, comma, or special character.

DEST

Limits stopping to traces started for particular destinations. You can use more than one value, but do not use the same value twice. If you do not specify a value for DEST, DB2 does not use destination to limit which traces to stop.

Abbreviation: D

Possible values and their meanings are:

Value Trace destination

GTF The generalized trace facility

SMF The System Management Facility

SRV An exit to a user-written routine

OP *n* A specific destination. *n* can be a value from 1 to 8

See -START TRACE (DB2) for a list of allowable destinations for each trace type.

CLASS(integer , ...)

Limits stopping to traces started for particular classes. For descriptions of the allowable classes, see -START TRACE (DB2). You cannot specify a class if you did not specify a trace type.

Abbreviation: C

The **default** is CLASS(), which does not limit the command.

TNO(integer , ...)

Limits stopping to particular traces, identified by their trace numbers (1 to 32,

01 to 09). You can use up to eight trace numbers. If you use more than one number, you can use only one value each for PLAN, AUTHID, and LOCATION.

The **default** is TNO(), which does not limit the command.

PLAN(*plan-name* , ...) or XPLAN(*plan-name* , ...)

Introduces a list of specific plans for which trace information is gathered. Use PLAN to constrain the trace to the specified plans or XPLAN to exclude the specified plans. You cannot use this option for a STAT trace.

The **default** is PLAN().

()

Stops a trace for all plans.

plan-name

The name of an application plan. You can use up to eight names. If you use more than one name, you can use only one value for AUTHID and LOCATION.

PKGLOC or XPKGLOC

Specifies the location name where the package is bound. Use PKGLOC to constrain the trace to the specified locations or XPKGLOC to exclude the specified locations.

PKGCOL or XPKGCOL

Specifies the package collection name. Use PKGCOL to constrain the trace to the specified collections or XPKGCOL to exclude the specified collections.

PKGPROG or XPKGPROG

Specifies the DBRM or program name. Use PKGPROG to constrain the trace to the specified programs or XPKGPROG to exclude the specified programs.

AUTHID(*authorization-id* , ...) or XAUTHID(*authorization-id* , ...)

Introduces a list of specific authorization IDs for which trace information is gathered. Use AUTHID to constrain the trace to the specified authorization IDs or XAUTHID to exclude the specified authorization IDs. The authorization IDs specified must be the primary authorization IDs. You cannot use this option for a STAT trace.

The **default** is AUTHID().

()

Stops a trace for all authorization IDs.

authorization-id

Specifies an authorization ID. You can use up to eight identifiers. If you use more than one identifier, you can use only one value for PLAN and LOCATION.

LOCATION(*location-name* , ...) or XLOC(*location-name* , ...)

Specifies a list of location names for which trace information is gathered. Use LOCATION to constrain the trace to the specified locations or XLOC to exclude the specified locations. The use of the LOCATION or XLOC option precludes tracing threads that have no distributed data relationship.

location-name

Identifies the DB2 subsystems whose distributed threads you want to trace. Activates the DB2 trace for the remote TCP/IP or SNA location that you specify by *location-name*.

You can specify up to eight locations. You can specify only one location if you use more than one plan name or authorization ID.

<luname>

Activates the DB2 trace for the remote clients that are connected to DDF through the remote SNA LU name that you specified in *luname*.

ipaddr

Activates the DB2 trace for the remote clients that are connected to DDF through the remote TCP/IP host. *ipaddr* is the IP address.

(*)

Indicates that you want to stop trace events that occur under distributed threads regardless of which location they are connected to. Specifying the local location name is equivalent to specifying LOCATION(*).

Clients other than DB2 for z/OS: DB2 for z/OS does not receive a location name from clients that are not DB2 for z/OS subsystems. To stop a trace for a client that is not a DB2 for z/OS subsystem, enter its LUNAME or IP address. Enclose the LUNAME by the less-than (<) and greater-than (>) symbols. Enter the IP address in the form *nnn.nnn.nnn.nnn*. For example, to stop a trace for a client with the LUNAME of LULA, enter the following command:

```
-START TRACE (*) LOCATION (<LULA>)
```

To stop a trace for a client with the IP address of 123.34.101.98, enter the following command:

```
-STOP TRACE (*) LOCATION (: :FFFF:123.34.101.98)
```

USERID or XUSERID

Specifies the user ID. Use **USERID** to constrain the trace to the specified user IDs or **XUSERID** to exclude the specified user IDs. You can specify multiple values and wildcard values as described in Usage notes.

APPNAME or XAPPNAME

Specifies the application name. Use **APPNAME** to constrain the trace to the specified applications or **XAPPNAME** to exclude the specified applications. You can specify multiple values and wildcard values as described in Usage notes.

WRKSTN or XWRKSTN

Specifies the workstation name. Use **WRKSTN** to constrain the trace to the specified workstations or **XWRKSTN** to exclude the specified workstations. You can specify multiple values and wildcard values as described in Usage notes.

CONNID or XCONNID

Specifies the connection ID. Use **CONNID** to constrain the trace to the specified connections or **XCONNID** to exclude the specified connections.

CORRID or XCORRID

Specifies the correlation ID. Use **CORRID** to constrain the trace to the specified correlation IDs or **XCORRID** to exclude the specified correlation IDs.

ROLE or XROLE

Specifies the connection roles. Use **ROLE** to constrain the trace to the specified roles or **XROLE** to exclude the specified roles.

AUDTPLCY

Stops the IFCIDs that correspond to the categories specified in the listed audit

policies. You can specify up to eight audit policy names. AUDTPLCY applies to trace type AUDIT. You cannot specify CLASS or IFCID with AUDTPLCY.

Usage notes

Stopping specific traces: Each option that you use, except TNO, limits the effect of the command to active traces that were started using the same option, either explicitly or by default, with exactly the same parameter values. For example, the following command stops only the active traces that were started using the options PERFM and CLASS (1,2):

```
-STOP TRACE (PERFM) CLASS (1,2)
```

This command does *not* stop, for example, any trace started using CLASS(1).

You must specify a trace type or an asterisk. For example, the following command stops all active traces:

```
-STOP TRACE (*)
```

Traces that use monitor trace class 6: When stopping trace classes, a special circumstance occurs if monitor trace class 6 is active. Monitor trace class 6 enables and disables data propagation. To avoid accidentally stopping this trace class, the commands STOP TRACE(*) and STOP TRACE(MON) CLASS(*) fail if monitor trace class 6 is active.

To stop monitor trace class 6, you must explicitly specify it as one of the arguments of the CLASS option of the STOP TRACE command, including any other monitor trace classes that were started with monitor trace class 6. For example, if monitor trace class 6 was started with the command STOP TRACE(MON) CLASS(1,3,6), the following command stops it:

```
-STOP TRACE(MON) CLASS(1,3,6)
```

In the case where monitor trace class 6 was started with the command STOP TRACE(MON) CLASS(*), you must explicitly specify all 32 monitor trace classes to have monitor trace class 6 stopped:

```
-STOP TRACE(MON) CLASS(1,2,3,4,5,6,...32)
```

However, if monitor trace class 6 is not active the STOP TRACE(*) command stops all active traces.

Traces started by a IFI/IFC program: Before you stop an active trace, ensure that an IFI application program or the IFC Selective Dump utility (DSN1SDMP) did not start the trace. If you stop a trace started by DSN1SDMP, the DSN1SDMP utility abnormally terminates.

Tracing threads using the * wildcard: You can use the wildcard suffix, "*" to filter threads. For example, if you specify "-STOP TRACE PLAN (A,B,C*)", DB2 will trace, and then return A, B, CDE, CDEFG, CDEFGH, and so on. It will trace threads "A", "B" and all threads starting with "C".

Tracing threads using the positional, () wildcard: You can use the positional wildcard, which is represented by the, "_" character, to trace threads when you want the wildcard in the middle, or when you want to trace threads of a specific length. For example, if you specify "-STOP TRACE PLAN (A_C)", all threads will be traced that are three characters that have "A" as the first character, and "C" as the third. This command would return "ABC", "ADC", "AEC", and so on. NOTE:

if you want to trace the thread "A_C" then you can specify "-STOP TRACE PLAN (A/_C). The "/" before the "_" tells DB2 to trace for the underscore in the search, rather than treating it as a wildcard. The same logic applies if you are trying to trace a thread that includes a "/" or "*" character. Because the character "/" is an escape character, if you are trying to trace a thread that has a "/" character in it, you can specify, for example, "-STOP TRACE PLAN (A//C)" to trace the thread "A/C". You can also specify "-STOP TRACE PLAN (A/*C) to trace the thread "A*C".

Tracing multiple threads at once using wildcards: You also have the option of tracing multiple threads based on multiple trace qualifications. For example, you can specify, "-STOP TRACE PLAN (A*, B*, C*)" to simultaneously trace ALL threads for plan that start with "A", "B", and "C". The wildcard character, "*" will trace all threads. You can specify more complex combinations such as, "-STOP TRACE PLAN (A_B*, C*, AND C/_D*)", which will return all threads that:

- Begin with "A", have a one character wild card as the second character in the thread, have a "B" as the third character in the thread, and end with any type or number of characters (ADBIOF, AOBTYJDP)
- Begin with "C", and end with any combination of characters (CDE, CGHKO)
- Begin with "C_D" and end with any type of character combination (C_DEFGH, C_DLMNOP)

All of the possible thread combinations listed above will be returned with the command above.

You have the ability to filter multiple threads at the same time, setting specific criteria for the trace. For example, you can specify "-STOP TRACE PLAN (A) USERID (B). This will trace the threads where the plan thread is A, and the user ID is B. When tracing threads, you can only specify more than one thread criteria for one filter per "-STOP TRACE" command. For example, you can specify "-STOP TRACE PLAN (A,B) USERID (B) WRKSTN (E)," but you cannot specify "-STOP TRACE PLAN (A, B) USER ID (A, B) WRKSTN (E) because, in this example, two of the filter qualifications have two elements defined to be traced, and DB2 only allows for one attribute to have more than one trace element to be defined per trace.

Filtering threads using exclude functionality: When you specify an "X" with any constraint keyword, such as "XPLAN", when you are filtering threads, you are using the exclude functionality for the -STOP TRACE command. You have the option of excluding specific types of threads when you are running trace commands. You can use the "X" character to exclude specific combinations of characters when you are running a trace. For example, you can specify "-STOP TRACE XPLAN(A), to trace all threads **except** "A". In this instance B, BCD, BCDE, or CD could possibly be returned.

You also have the option of excluding multiple types of threads from your trace. For example, you can specify, "-STOP TRACE XPLAN (A*, B*)" to trace all threads **except** those starting with "A", with any combination of characters following "A", and all those characters starting with "B", with any combination of characters following "B". Note: Specifying XPLAN (*) excludes all threads from your search, and is not allowed. You also cannot use the wildcard in the middle of a -STOP TRACE command with exclude functionality, such as, "-STOP TRACE XPLAN (A*C)." You can, however, specify "-STOP TRACE XPLAN (A_ _ C *)", which will return all threads **except** those starting with "A", a variety of **two** characters next, a "C" in the fourth space, and a variety of characters at the end. The wildcard symbol cannot be placed in the middle of trace criteria.

You have the ability to stop two traces at once, in order to help you optimize your tracing capabilities. For example, you can specify (-STOP TRACE XPLAN (A, B, C) USERID (D)). This tells DB2 to stop tracing all threads for plans **except** threads "A", "B", or "C", only where the user ID = "D".

Combining trace qualifiers: You can customize the threads you trace by commanding DB2 to trace specific threads, while excluding other specific threads. For example, you can specify, "-STOP TRACE USERID (A,B) XPLAN (C)". This criteria only traces threads where the user ID is equal to "A" or "B", and the plan is **not** equal to "C". In this example, a thread where the user ID is "A" and the plan is equal to "D" would pass, and be traced, but a thread where the user ID is "A" and plan is "C" would not pass, and would not be traced.

You can introduce wildcards into your stop trace commands to add more customization to your traces. For example, you can specify "-STOP TRACE PLAN (C*) USER ID (Z, X) XPLAN (C, D, E)". In this example, for the thread to be traced, the plan must begin with C, the user ID must be equal to Z or to X, and the plan cannot be C, D, or E. Therefore, a plan of CB, with a user ID of Z would pass, and the thread would be traced, but plan C with a user ID of X would fail because the command specifies not to trace threads where the plan is "C", without additional characters in the thread.

Examples

Example 1: Stop all traces that have the generalized trace facility as their only destination.

```
-STOP TRACE (*) DEST (GTF)
```

Example 2: Stop an accounting trace of all threads between the local and USIBMSTODB21 DB2 subsystems for plan DSN8BC81. Include a comment.

```
-STOP TRACE (ACCTG)
PLAN (DSN8BC81)
LOCATION (USIBMSTODB21)
COMMENT('ACCTG TRACE FOR DSN8BC81')
```

Example 3: Stop trace number 4.

```
-STOP TRACE (P) TNO(4)
```

Example 4: Stop all active traces of any type for USIBMSTODB22.

```
-STOP TRACE (*) LOCATION (USIBMSTODB22)
```

Example 5: Stop all performance traces.

```
-STOP TRACE=P
```

Example 6: Stop all monitor tracing.

```
-STOP TRACE(MON)
```

Example 7: Stop all monitor tracing in a data sharing group.

```
-STOP TRACE(MON) SCOPE(GROUP)
```

Example 8: Stop all audit tracing that uses audit policy AUDITADMIN.

```
-STO TRACE(AUDIT) DEST(GTF) AUDTPLCY(AUDITADMIN)
```

Related reference:

```
-DISPLAY TRACE (DB2)
```

|
|
|

-MODIFY TRACE (DB2)
-START TRACE (DB2)

Chapter 94. -TERM UTILITY (DB2)

The DB2 command TERM UTILITY terminates execution of a DB2 utility job step and releases all resources associated with the step.

When executing, a utility does not terminate until it checks to see that the TERM UTILITY command was issued. Active utilities perform this check periodically. If the utility is stopped, all its resources are released by the TERM UTILITY command. An active utility can be terminated only from the DB2 on which it is running. A stopped utility can be terminated from any active member of the data sharing group.

Abbreviation: -TER UTIL

Environment

This command can be issued from a z/OS console, a DSN session, DB2I panels DB2 COMMANDS and DB2 UTILITIES, an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

Data sharing scope: Group or member. The utility is implicitly of group scope when the utility is stopped.

Authorization

To execute this command, you must use the primary or some secondary authorization ID of the process that originally submitted the utility job, or you must use a privilege set of the process that includes one of the following authorities:

- DBMAINT authority
- DBCTRL authority
- DBADM authority
- SYSOPR authority
- System DBADM authority
- DATAACCESS authority
- SYSCTRL authority
- SYSADM authority

For utilities that run on objects in implicitly created databases, the database privilege or authority can be held on the implicitly created database or on DSNDB04. Ownership of the table spaces or index spaces on which the utility operates is also sufficient to terminate the utility. This means that the owner can display information about an implicitly created table space or index space if the command explicitly specifies that table space or index space name.

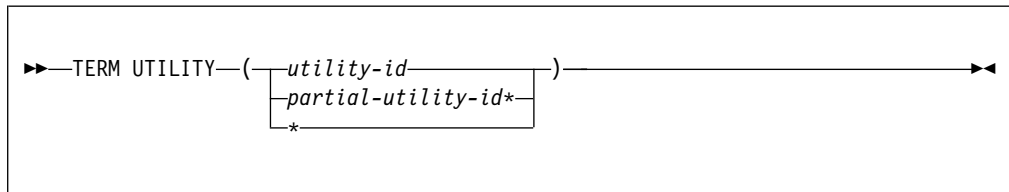
The utilities DIAGNOSE, REPORT, and STOSPACE can be terminated only by the job submitter or by a holder of SYSOPR, SYSCTRL, or SYSADM authority.

DB2 commands that are issued from a logged-on z/OS console or TSO SDSF can be checked by DB2 authorization using primary and secondary authorization IDs.

For users with DBMAINT, DBCTRL, or DBADM authority, the command takes effect only when DB2 can determine that the user has sufficient authority over each object that the utility job accesses.

Database DSNDB06 contains the table spaces and index spaces that are required to check authorization. If a table or index space that is required for authorization checking is affected by a utility that you need to terminate, installation SYSADM authority is required to terminate that utility.

Syntax



Option descriptions

One of the following parameters must be specified.

(*utility-id*)

Is the utility identifier, or the UID parameter used when creating the utility job step.

If *utility-id* was created by the DSNU CLIST by default, it has the form *tso-userid.control-file-name*.

If *utility-id* was created by default by the EXEC statement invoking DSNUTILB, then the token has the form *userid.jobname*.

If *utility-id* contains lowercase letters or special characters, it must be enclosed in single quotation marks (').

(*partial-utility-id**)

Terminates every utility job that begins with *partial-utility-id*. For example, TERM UTILITY(ABCD*) terminates every utility job step whose utility identifier begins with the letters ABCD. If you have a two-part utility ID, such as ABCD.EFGH, TERM UTILITY(ABCD*) also terminates that utility.

(*)

Terminates every utility job step known to DB2 for which you are authorized.

Usage notes

Restarting utilities: A terminated utility job step cannot be restarted. You must resubmit the step as a new utility job.

What happens to particular utilities: In some cases, terminating a utility job can leave work in an undesirable state, requiring special processing before the job can be resubmitted. The following list describes the effects of TERM UTILITY on jobs for each of the utilities:

Utility Special effects of the TERM UTILITY command

CATENFM

None.

CATMAINT

Places indexes in REBUILD-pending status.

CHECK DATA

Table spaces remain in CHECK-pending status.

CHECK INDEX

None.

CHECK LOB

Places LOB table spaces and indexes in the utility read-only (UTRO) state.

COPY Inserts "T" record in SYSIBM.SYSCOPY. When you run COPY, COPY does not allow an incremental image copy if the "T" record exists.

DIAGNOSE

None.

LOAD

Termination of LOAD

MERGECOPY

None.

MODIFY RECOVERY

None.

MODIFY STATISTICS

None.

QUIESCE

None.

REBUILD INDEX

Places the object that is being rebuilt in REBUILD-pending status.

RECOVER

Places the object that is being recovered in RECOVER-pending status.

REORG INDEX

Termination of REORG INDEX

REORG TABLESPACE

Termination of REORG TABLESPACE

REPAIR

None.

REPORT

None.

RUNSTATS

None.

STOSPACE

None.

UNLOAD

The output data set remains incomplete until you restart the utility job or delete the data set.

Examples


Example 1: Terminate all utility jobs for which you are authorized.

-TERM UTILITY (*)

Example 2: Terminate all utility jobs whose utility ID begins with SMITH.

```
-TERM UTILITY  
  (SMITH*)
```

Related tasks:

 [Terminating an online utility \(DB2 Utilities\)](#)

Chapter 95. /TRACE (IMS)

The IMS /TRACE command directs and controls the IMS capabilities for tracing internal IMS events. It also starts, stops, and defines the activity to be monitored by the IMS Monitor.

Abbreviation: /TRA

Environment

This command can be issued only from an IMS terminal.

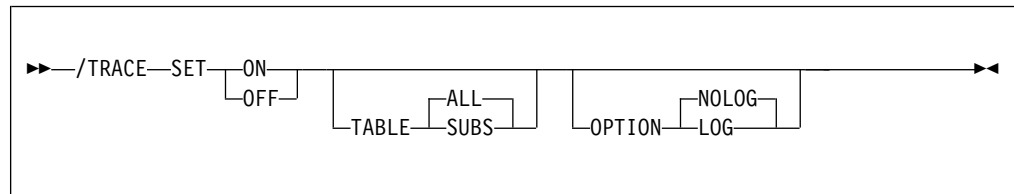
Data sharing scope: Member

Authorization

To enter this command, users must have passed the IMS security check.

The syntax diagram includes only those parameters that DB2 users need to know.

Syntax



Option descriptions

This topic provides information about the two parameters of the TRACE command that are especially important for DB2 users.

SUBS

Indicates that the external subsystem trace table (containing information about every interaction with DB2) is to be enabled or disabled. SET ON TABLE SUBS enables the DB2 trace facility, and SET OFF TABLE SUBS disables it.

If nothing is specified with the TABLE keyword, then the default is ALL; ALL includes SUBS, as well as other trace tables.

LOG

Specifies that traced data is to be written to the IMS system log. Because IMS has a tracing mechanism that writes trace entries to the IMS system log, it is important that DB2 users specify SET ON and TABLE OPTION LOG.

Otherwise, the trace information that IMS provides will not be available unless a control region dump occurs.

Examples

Example 1: This command starts IMS tracing and:

- Enables the DB2 trace

- Writes IMS trace tables to the IMS log before they wrap.

```
/TRACE SET ON TABLE SUBS OPTION LOG
```

Example 2: This command starts IMS tracing and:

- Enables all trace tables (including DB2 trace tables); (ALL is the default parameter for the TABLE keyword)
- Writes IMS trace tables to the IMS log before they wrap.

```
/TRACE SET ON TABLE ALL OPTION LOG
```

Chapter 96. TRACE CT (z/OS IRLM)

The z/OS command TRACE CT starts, stops, or modifies a diagnostic trace for the internal resource lock manager (IRLM) of DB2.

IRLM does not support all the options available on the TRACE command.

Environment

This command can be issued only from a z/OS console.

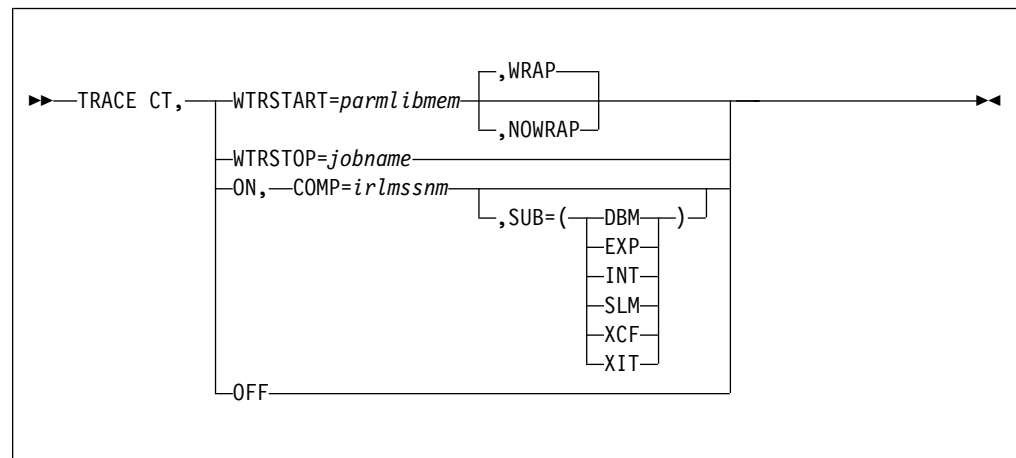
Data sharing scope: Member

Authorization

This command requires an appropriate level of operating system authority.

The syntax diagram and option descriptions for this command are purposely incomplete.

Syntax



Option descriptions

CT Specifies the component trace. (Do not use other trace options available on the z/OS TRACE command).

WTRSTART= *parmlibmem*

Identifies the member that contains source JCL. That JCL executes the CTRACE writer and defines the data set to which it writes the trace buffers. This member can be a procedure cataloged in SYS1.PROCLIB or a job.

WRAP

Specifies that when the system reaches the end of the group of data sets, it writes over the oldest data at the beginning of the first data set in the group. The system uses only the primary extents of the data sets.

NOWRAP

Specifies that the system stops writing to the data sets when they are all full. The system uses the primary and secondary extents of the data sets.

WTRSTOP= *jobname*

Stops the CTRACE writer for a trace that is running. The system also closes the data sets that the writer used.

jobname identifies the trace, either by:

- Member name, if the source JCL is a procedure
- Job name, if that appears on a JOB statement in the source JCL

ON Turns on the trace.**COMP=** *irlmssnm*

Gives the IRLM subsystem name.

SUB= *subname*

Specifies the type of sublevel trace. Traces INT, EXP, and XIT are ON by default. You cannot turn off traces INT and EXP. If you do not specify a subname on the TRACE command, the trace is performed on all subnames that you control. Specifying one subname restricts the traces to that trace plus the EXP and INT traces.

Use: To trace:

DBM Interactions with the identified DBMS

EXP Any exception condition

INT Member and group events outside normal locking activity

SLM Interactions with the z/OS locking component

XCF All interactions with z/OS cross-system coupling services

XIT Only asynchronous interactions with the z/OS locking component

OFF

Turns off the trace. If IRLM is connected to a CTRACE writer for the component trace, the system disconnects it.

Usage notes

Include the IRLM load module in the z/OS link list: This command uses z/OS component trace services. Include the IRLM load module DXRRL183, which contains a routine for stopping and starting, in the z/OS link list.

Displaying a trace: To display a trace, use the z/OS DISPLAY command:

```
D TRACE,COMP=IRLM
```

The z/OS DISPLAY TRACE command output is incorrect for IRLM unless you use TRACE CT commands to inform z/OS of the TRACE status. IRLM initializes its own traces and writes them in CTRACE format, but IRLM has no interface to z/OS to inform it of the status. If you want to know the true status of the traces without using TRACE CT commands to inform z/OS, use the MODIFY *irlmproc,STATUS,TRACE* command.

Monitoring a trace: To monitor a trace, use the z/OS MODIFY *irlmproc,STATUS,TRACE* command.

Setting the number of trace buffers: To set the number of trace buffers used by traces, use the z/OS MODIFY *irlmproc,SET* command.

Sample procedure for the CTRACE writer: This procedure identifies the data set to which the next sample procedure writes data. The external trace writer must be executed at the same or higher dispatch priority as IRLM. This allows the I/O to keep up with the filling of the trace buffers.

```
//CTWTR   PROC
//        EXEC PGM=ITTTTCWR
//TRCOUT01 DD  DSNAME=SYS1.WTR1,DISP=OLD
//TRCOUT02 DD  DSNAME=SYS1.WTR2,DISP=OLD
```

Sample procedure to start and stop a DBM trace to the CTRACE writer: These commands start and stop an IRLM DBM trace. The trace data is written to an external writer data set that is identified in procedure CTWTR.

```
TRACE CT,WTRSTART=CTWTR
TRACE CT,ON,COMP=IRLM,SUB=(DBM)
:
: (z/OS asks for a reply)
:
R 15,WTR=CTWTR,END
TRACE CT,OFF,COMP=IRLM,SUB=(DBM)
:
: (Wait to make sure trace buffers are externalized.)
TRACE CT,WTRSTOP=CTWTR
```

Sample procedure to start and stop traces in wrap-around mode: Traces captured in this procedure are saved in a limited number of buffers that are provided by IRLM. Each buffer is reused when the previous buffer is filled. To start the trace in this wrap-around mode, enter the following commands:

```
TRACE CT,ON,COMP=IRLM
:
: (z/OS asks for a reply)
:
R 15,END
:
TRACE CT,OFF,COMP=IRLM
```

Impact of setting TRACE CT ON: Each active subname type requires up to 0.7 MB of ECSA. Because IRLM initializes its own traces when it starts, the DISPLAY TRACE command shows that all traces are off. After you issue the TRACE ON command, the reports are accurate except for the two subname types, INT and EXT, which cannot be turned off.

Part 3. Appendixes

Information resources for DB2 10 for z/OS and related products

Information about DB2 10 for z/OS and products that you might use in conjunction with DB2 10 is available online in IBM Knowledge Center or on library websites.

Obtaining DB2 for z/OS publications

Current DB2 10 for z/OS publications are available from the following websites:

<http://www-01.ibm.com/support/docview.wss?uid=swg27019288>

Links to IBM Knowledge Center and the PDF version of each publication are provided.

DB2 for z/OS publications are also available for download from the IBM Publications Center (<http://www.ibm.com/shop/publications/order>).

In addition, books for DB2 for z/OS are available on a CD-ROM that is included with your product shipment:

- DB2 10 for z/OS Licensed Library Collection, LK5T-7390, in English. The CD-ROM contains the collection of books for DB2 10 for z/OS in PDF format. Periodically, IBM refreshes the books on subsequent editions of this CD-ROM.

Installable information center

You can download or order an installable version of the Information Management Software for z/OS Solutions Information Center, which includes information about DB2 10 for z/OS, QMF, IMS, and many DB2 Tools for z/OS products. You can install this information center on a local system or on an intranet server. For more information, see http://www-01.ibm.com/support/knowledgecenter/SSEPEK_11.0.0/com.ibm.db2z11.doc/src/alltoc/installabledzic.html.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785 US*

For license inquiries regarding double-byte (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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those 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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785 US*

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.

This information contains 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 actual people or business enterprises 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 shown below:

© *(your company name) (year)*.

Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. *(enter the year or years)*.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

This information is intended to help you to plan for and administer DB2 10 for z/OS. This information primarily documents General-use Programming Interface and Associated Guidance Information provided by DB2 10 for z/OS. This information also documents Product-sensitive Programming Interface and Associated Guidance Information provided by DB2 10 for z/OS.

General-use Programming Interface and Associated Guidance Information

General-use Programming Interfaces allow the customer to write programs that obtain the services of DB2 10 for z/OS.

Product-sensitive Programming Interface and Associated Guidance Information

Product-sensitive Programming Interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive Programming Interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs by the following markings:



Product-sensitive Programming Interface and Associated Guidance Information...



Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com)[®] are trademarks or registered marks 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: <http://www.ibm.com/legal/copytrade.shtml>.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Terms and conditions for product documentation

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.

Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at <http://www.ibm.com/privacy> and IBM's Online Privacy Statement at <http://www.ibm.com/privacy/details> the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at <http://www.ibm.com/software/info/product-privacy>.

Glossary

The glossary is available in IBM Knowledge Center.

See the Glossary topic for definitions of DB2 for z/OS terms.

Index

Special characters

* (asterisk)

DISPLAY THREAD command 257
FREE PACKAGE command 323
FREE QUERY command 328
REBIND PACKAGE command 140

. (asterisk)

DISPLAY PROCEDURE command 248
START PROCEDURE command 452
STOP FUNCTION SPECIFIC command 498
STOP PROCEDURE command 504

A

ABEND subcommand of DSN 285

description 55

accelerators

displaying 181

ACCESS DATABASE

command 57

ACCESS option

START DATABASE command 429

START DB2 command 436

accessibility

keyboard x

shortcut keys x

accounting

trace

displaying 271

starting 459

stopping 511

ACCTG option

DISPLAY TRACE command 274

MODIFY TRACE command 358

STOP TRACE command 514

ACQUIRE option

BIND PLAN subcommand 101

DSNH command 308

REBIND PLAN subcommand 101

ACTION option

BIND PACKAGE subcommand 102

BIND PLAN subcommand 102

DCLGEN subcommand 169

DSNH command 308

RECOVER INDOUBT command 380

STOP FUNCTION SPECIFIC command 498

STOP PROCEDURE command 504

ACTIVE option

DISPLAY BUFFERPOOL command 188

DISPLAY DATABASE command 207

ADD option

DCLGEN subcommand 169

DSNH command 311

ADD XMLSCHEMA DOCUMENT command 26

administrative task scheduler

commands

MODIFY admtproc, APPL=SHUTDOWN option 331

MODIFY admtproc, APPL=TRACE option 333

START admtproc 421

STOP admtproc 479

administrative task scheduler (*continued*)

starting

an administrative task scheduler 421

stopping

an administrative task scheduler 479

Administrative task scheduler commands 9

ADVISORY option of DISPLAY DATABASE command 209

AFTER option of DISPLAY DATABASE command 207

ALL keyword of MODIFY irlmproc,DIAG command 343

ALLD option of MODIFY irlmproc,STATUS command of z/OS 351

ALLI option of MODIFY irlmproc,STATUS command of z/OS 352

ALTER BUFFERPOOL command

description 61

example 66

Option descriptions 62

ALTER GROUPBUFFERPOOL command

description 67

example 70

Option descriptions 67

ALTER UTILITY command

description 71

example 73

ambiguous cursor 110

APCOMPARE option

BIND PACKAGE subcommand 103

REBIND PACKAGE subcommand 103

REBIND TRIGGER PACKAGE subcommand 103

APOST option

DCLGEN subcommand 171

DSNH command 300

APOSTSQL option of DSNH command 306

application plan

binding 91

deleting 325

rebinding, changing plans 367

application program

START command 423

testing 285

application program, preparing for DSNH CLIST

processing 289

APPNAME option

DISPLAY TRACE command 273

START TRACE command 466

STOP TRACE command 514

APRETAINDUP option

REBIND PACKAGE subcommand 104

REBIND TRIGGER PACKAGE subcommand 104

APREUSE option

BIND PACKAGE subcommand 105

REBIND PACKAGE subcommand 105

REBIND TRIGGER PACKAGE subcommand 105

ARCHIVE LOG command

description 75

option descriptions 76

ASMLIB option of DSNH command 295

ASMLOAD option of DSNH command 295

asterisk (*.*)

DISPLAY PROCEDURE command 248

START PROCEDURE command 452

- asterisk (*:*) (*continued*)
 - STOP FUNCTION SPECIFIC command 498
 - STOP PROCEDURE command 504
- asterisk (*)
 - DISPLAY THREAD command 257
 - FREE PACKAGE command 323
 - FREE QUERY command 328
 - REBIND PACKAGE command 140
- ASUSER option
 - DSN command 286
- AT option of DCLGEN subcommand 169
- AT(COMMIT) option of STOP DATABASE command 486
- AUDIT option
 - DISPLAY TRACE command 274
 - MODIFY TRACE command 358
 - STOP TRACE command 514
- audit trace
 - displaying 271
 - starting 459
 - stopping 511
- AUDTPLY option
 - START TRACE command 277
 - STOP TRACE command 517
- AUDTPLY option of START TRACE command 471
- AUTHID option
 - DISPLAY TRACE command 273
 - START TRACE command 464
 - STOP TRACE command 514
- authorization IDs
 - commands 3
- AUTOREC option of ALTER GROUPBUFFERPOOL command 68

B

- BDBRMLIB option of DSNH command 309
- BDMEM option of DSNH command 308
- BIND command 28, 30, 45
- BIND PACKAGE subcommand of DSN
 - description 81
 - example 89
 - option descriptions 101, 154
- BIND PLAN subcommand of DSN
 - description 91
 - example 94
 - option descriptions 101, 154
- BIND QUERY command
 - option descriptions 98
- BIND QUERY subcommand of DSN
 - description 97
 - example 99
- binding
 - DSNH processing 289
 - initiating 81, 91
 - options for 101, 154
- binding DB2 for z/OS packages 157
- BLIB option of DSNH command 308
- BMEM option of DSNH command 308
- BnLIB option of DSNH command 308
- BOTH option
 - SET LOG command 404
- BSDS (bootstrap data set), recovery 377
- buffer pool
 - active and inactive 61, 187
 - altering attributes 61
 - displaying current status 187
 - parallel sequential steal threshold (VPSEQT) 62

- BUFSIZE option of START TRACE command 471

C

- C option of DCLGEN subcommand 170
- CACHESIZE option
 - BIND PLAN subcommand 106
 - DSNH command 309
 - REBIND PLAN subcommand 106
- CANCEL OFFLOAD option of ARCHIVE LOG command 77
- CANCEL option of RECOVER POSTPONED command 383
- CANCEL THREAD command
 - description 159
 - example 164
- canceling threads, description 159
- CASTOUT option of STOP DB2 command 492
- CATENFM utility, effects of TERM command 524
- CATMAINT utility, effects of TERM command 525
- CCLINK option of DSNH command 295
- CCLLIB option of DSNH command 295
- CCLOAD option of DSNH command 295
- CCMSGS option of DSNH command 295
- CCOLIB option of DSNH command 295
- CCPLIB option of DSNH command 295
- CCPMSGS option of DSNH command 295
- CCSID option of DSNH command 295
- CCSLIB option of DSNH command 295
- CHANGE command of IMS
 - description 165
 - example 166
- CHECK DATA utility, effects of TERM command 525
- CHECK INDEX utility, effects of TERM command 525
- CHECK LOB utility, effects of TERM command 525
- CHKTIME option
 - SET LOG command 404
- CICS
 - option of BIND and REBIND subcommands 122
 - option of DSNH command 309
 - translation step in DSNH processing 289
- CICS commands 8
- CICSCOB option of DSNH command 296
- CICSLIB option of DSNH command 296
- CICSOPT option of DSNH command 296
- CICSPLIB option of DSNH command 296
- CICSPRE option of DSNH command 296
- CICSVER option of DSNH command 296
- CICSXLAT option of DSNH command 296
- CLAIMERS option of DISPLAY DATABASE command 205
- CLASS option
 - DISPLAY TRACE command 275
 - IFCIDs activated by trace class 467
 - MODIFY TRACE command 358
 - START TRACE command 467
 - STOP TRACE command 515
- CLASST option of ALTER GROUPBUFFERPOOL command 69
- CLIB option of DSNH command 296
- CLONE option
 - START DATABASE command 428
 - STOP DATABASE command 486
- CnLIB option of DSNH command 296
- COB2CICS option of DSNH command 297
- COB2LIB option of DSNH command 297
- COB2LOAD option of DSNH command 297
- COBICOMP option of DSNH command 296
- COBILINK option of DSNH command 296
- COBIPLNK option of DSNH command 296

- COBIPMSG option of DSNH command 297
- COBLIB option of DSNH command 297
- COBLOAD option of DSNH command 297
- COBSOM option of DSNH command 297
- code, return 287
- collection, package
 - BIND PACKAGE subcommand 139
 - REBIND PACKAGE subcommand 139
 - REBIND TRIGGER PACKAGE subcommand 375
- COLLID option
 - REBIND PLAN subcommand 107
- COLSUFFIX option of DCLGEN subcommand 171
- column name, as a field name 171
- comma option of DSNH command 299
- command continuation character 15
- command line processor 26, 28, 30, 31, 42, 44, 45, 46
 - CHANGE ISOLATION command 29
 - CHANGE MAXCOLUMNWIDTH command 29
 - CHANGE MAXLINESFROMSELECT command 30
 - command for stopping 48
 - CONNECT command 32
 - DESCRIBE CALL command 33
 - DESCRIBE TABLE command 35
 - DISCONNECT command 36
 - DISPLAY RESULT SETTINGS command 36
 - ECHO command 36
 - help command 37
 - LIST COMMAND OPTIONS command 38
 - LIST TABLES command 38
 - properties file 39
 - ROLLBACK command 44
 - running 25
 - TERMINATE command 48
 - tutorial 21
 - UPDATE COMMAND OPTIONS command 49
 - z/OS UNIX System Services command 48
- command output 19
- command prefix
 - description 11
 - multiple subsystems 437
 - part of a command 11
- command recognition character (CRC) 11
- commands 335
 - ACCESS DATABASE 57
 - authorization IDs 3
 - privileges 3
- commands, scope 17
- comment
 - DCLGEN subcommand output 172
 - DSN subcommands 285
- COMMENT option
 - DISPLAY TRACE command 274
 - MODIFY TRACE command 359
 - START TRACE command 462
 - STOP TRACE command 515
- commit point, terminating utility 523
- COMP option of TRACE CT command 530
- COMPILE option of DSNH command 297
- COMPLETE XMLSCHEMA command 31
- CONCURRENTACCESSRESOLUTION option
 - BIND PACKAGE command 107
 - BIND PLAN command 107
 - REBIND PACKAGE command 107
 - REBIND PLAN command 107
 - REBIND TRIGGER PACKAGE command 107
- conditional restart
 - control record, effect on restart 437
- CONNECT option of DSN command 297
- connection
 - DB2
 - GROUP option of DSN command 286
 - RETRY option of DSN command 286
 - displaying
 - connection information 255
 - group buffer pool 235
 - IRLM subsystem status 353
 - status 177
 - terminating 481
- CONNID option
 - DISPLAY TRACE command 273
 - START TRACE command 466
 - STOP TRACE command 514
- CONNLIST option of DISPLAY GROUPBUFFERPOOL command 235
- CONTROL option of DSNH command 297
- COPTION option of DSNH command 297
- COPY option
 - BIND PACKAGE subcommand 108
 - DSNH command 311
 - SET LOG command 405
- COPY utility, effects of TERM command 525
- COPYVER option
 - BIND PACKAGE subcommand 108, 115
 - DSNH command 311
- correlation ID
 - recovering threads 380
- CORRELATION option of START TRACE command 471
- CORRID option
 - DISPLAY TRACE command 273
 - START TRACE command 467
 - STOP TRACE command 514
- COUNT option of SET ARCHIVE command 400
- CP option of RUN subcommand 395
- CPP option of DCLGEN subcommand 170
- CPPCLASS option of DSNH command 298
- CPPCLLIB option of DSNH command 298
- CPPCSLIB option of DSNH command 298
- CPPLINK option of DSNH command 298
- CPPLLIB option of DSNH command 298
- CPPPMMSG option of DSNH command 298
- CPPSLIB option of DSNH command 298
- CPPUTIL option of DSNH command 298
- CPU option of START TRACE command 472
- CRC (command recognition character), description 11
- CURRENTDATA option
 - BIND PACKAGE subcommand 110
 - BIND PLAN subcommand 110
 - DSNH command 309, 311
 - REBIND PACKAGE subcommand 110
 - REBIND PLAN subcommand 110
 - REBIND TRIGGER PACKAGE subcommand 110
- CURRENTSERVER option
 - BIND PLAN subcommand 111
 - DSNH command 309
 - REBIND PLAN subcommand 111
- cursor, ambiguous 110
- CYLINDER option of DSNH command 306

D

- data sharing
 - delays, diagnosing 343
 - displaying
 - archive log information 185

- data sharing (*continued*)
 - displaying (*continued*)
 - information about groups 229
 - status of members 229
 - identifying members with utility jobs 282
 - scope of commands 17
 - starting members 438
- database
 - displaying status 201
 - reserved names 487
 - starting 425
 - stopping 483
- DATE
 - option of DSNH command 298
- DB2 commands 335
 - command names 11
 - commands
 - ACCESS DATABASE 57
 - ALTER BUFFERPOOL 61
 - ALTER GROUPBUFFERPOOL 67
 - ALTER UTILITY 71
 - ARCHIVE LOG 75
 - CANCEL THREAD 159
 - DISPLAY ACCEL 181
 - DISPLAY ARCHIVE 185
 - DISPLAY BUFFERPOOL 187
 - DISPLAY DATABASE 201
 - DISPLAY DDF 221
 - DISPLAY GROUP 229
 - DISPLAY GROUPBUFFERPOOL 233
 - DISPLAY LOCATION 241
 - DISPLAY LOG 245
 - DISPLAY PROCEDURE 247
 - DISPLAY PROFILE 251
 - DISPLAY RLIMIT 253
 - DISPLAY THREAD 255
 - DISPLAY TRACE 271
 - DISPLAY UTILITY 281
 - MODIFY TRACE 357
 - RECOVER BSDS 377
 - RECOVER INDOUBT 379, 382
 - RECOVER POSTPONED 383
 - REFRESH DB2,EARLY 387
 - RESET INDOUBT 391
 - SET SYSPARM 409
 - START ACCEL 417
 - START DATABASE 425
 - START DB2 435
 - START DDF 439
 - START FUNCTION SPECIFIC 441
 - START PROCEDURE 451
 - START PROFILE 455
 - START RLIMIT 457
 - START TRACE 459
 - STOP ACCEL 477
 - STOP DATABASE 483
 - STOP DB2 491
 - STOP DDF 493
 - STOP FUNCTION SPECIFIC 497
 - STOP PROCEDURE 503
 - STOP PROFILE 507
 - STOP RLIMIT 509
 - STOP TRACE 511
 - TERM UTILITY 523
 - completion messages 7
 - description of 11
 - DISPLAY FUNCTION SPECIFIC 225
- DB2 commands (*continued*)
 - entering from supported environments 7
 - scope 17
 - separator 11
- DB2 for z/OS packages
 - binding from IBM Data Server clients and drivers 157
- DB2 precompiler 11
- DBM1 option of START DB2 command 436
- DBPROTOCOL option
 - BIND PACKAGE subcommand 112
 - BIND PLAN subcommand 112
 - DSNH command 309
 - REBIND PACKAGE subcommand 112
 - REBIND PLAN subcommand 112
- DBRM (database request module)
 - BIND PLAN subcommand 136
- DBRMLIB option of DSNH command 299
- DCLGEN subcommand of DSN 172
 - declaring an indicator variable array 172
 - description 167
 - example 174
 - forming field names 171
 - option descriptions 168
- DDF 335
- DDF (distributed data facility), displaying 221
- DDF (distributed data facility), modifying 335
- DEADLINE option of ALTER UTILITY command 72
- DEADLOCK option of START irlmproc command 445
- DECARTH option of DSNH command 299
- DECIMAL
 - option of DSNH command 299
- DECP option 436
- DEFAULT option of SET ARCHIVE command 400
- DEFER
 - option of BIND PLAN subcommand 113
- DEFER option
 - BIND PACKAGE subcommand 113
 - BIND PLAN subcommand 113
 - DSNH command 309
 - REBIND PACKAGE subcommand 113
 - REBIND PLAN subcommand 113
- degree of parallel processing 114
- DEGREE option
 - BIND PACKAGE subcommand 114
 - BIND PLAN subcommand 114
 - DSNH command 309
 - REBIND PACKAGE subcommand 114
 - REBIND PLAN subcommand 114
- DELAY
 - keyword of MODIFY irlmproc,DIAG command 343
- DELAY option of ALTER UTILITY command 73
- deleting, IMS units of recovery 165
- DELIMIT option of DSNH command 300
- DEPLOY option
 - BIND PACKAGE subcommand 115
- deregistering IRLM 342
- description 335
- DEST option
 - DISPLAY TRACE command 274
 - START TRACE command 462
 - STOP TRACE command 515
- DETAIL option
 - DISPLAY BUFFERPOOL command 188
 - DISPLAY GROUP command 229
 - DISPLAY LOCATION command 242
 - DISPLAY THREAD command 260
 - DISPLAY TRACE command 274

DIAG keyword of MODIFY irlmproc,DIAG command 343
 DIAGNOSE utility, TERM command effects 525
 diagnostic dumps, IRLM 343
 disability x
 DISABLE option
 BIND PACKAGE subcommand 122
 BIND PLAN subcommand 122
 DSNH command 309
 REBIND PACKAGE subcommand 122
 REBIND PLAN subcommand 122
 disabling a function permanently 499
 DISCONNECT option
 BIND PLAN subcommand 116
 DSNH command 309
 REBIND PLAN subcommand 116
 DISPLAY ACCEL command
 description 181
 example 182
 Option descriptions 181
 DISPLAY ARCHIVE command 185
 DISPLAY BUFFERPOOL command
 description 187
 Option descriptions 188
 Output 190
 DISPLAY command of IMS
 description 177
 example 179
 Option descriptions 177
 DISPLAY DATABASE command
 description 201
 example 211
 Option descriptions 204
 DISPLAY DDF command
 description 221
 example 222
 Option descriptions 221
 DISPLAY FUNCTION SPECIFIC command
 description 225
 Examples 227
 Output 227
 DISPLAY GROUP command
 description 229
 Examples 230
 DISPLAY GROUPBUFFERPOOL command
 description 233
 option descriptions 234
 summary report example 235
 DISPLAY LOCATION command
 description 241
 example 242
 option descriptions 242
 DISPLAY LOG command
 description 245
 example 246
 DISPLAY NET command of VTAM 161
 DISPLAY PROCEDURE command
 description 247
 example 249
 Option descriptions 248
 Output 249
 DISPLAY PROFILE command
 example 251
 DISPLAY RLIMIT command 253
 DISPLAY THREAD command
 description 255
 example 262
 Option descriptions 257
 DISPLAY THREAD command (*continued*)
 Output 262
 DISPLAY TRACE command
 description 271
 example 279
 Option descriptions 273
 DISPLAY UTILITY command
 description 281
 example 282
 option descriptions 281
 displaying
 information about
 accelerators 181
 archive logs 185
 communications database and resource limit facility 210
 data sharing group 229
 data-partitioned secondary indexes 210
 DB2 functions 225
 DB2 threads 255
 DDF 221
 logical partitions 210
 logs 245
 resource limit facility (governor) 253
 restricted objects 209
 stored procedures 247
 threads with remote locations 241
 trace activity 271
 status of
 buffer pools 187
 DB2 databases 201
 DB2 utilities 281
 group buffer pools 233
 DIST option of START DB2 command 437
 DISTRIBUTED option of START TRACE command 472
 DLIBATCH option
 BIND and REBIND subcommands 123
 DSNH command 309
 DSN command
 ASUSER option 286
 GROUP option 286
 TEST option 286
 DSN command of TSO
 abbreviations 15
 description 285
 example 288
 Option descriptions 286
 parsing subcommands 15
 return codes 287
 subcommands
 ABEND 55, 285
 BIND PACKAGE 81
 BIND PLAN 91
 BIND QUERY 97
 DCLGEN 167
 END 319
 FREE PACKAGE 321
 FREE PLAN 325
 FREE QUERY 327
 REBIND PACKAGE 361
 REBIND PLAN 367
 REBIND TRIGGER PACKAGE 373
 RUN 395
 SPUFI 413
 DSN commands 5
 DSN subcommands 5
 dsnb461i 197

dsnb462i 197
 DSNDB01 database, authority needed to start 426
 DSNDB06 database, authority needed to start 426
 DSNH command of TSO
 data set names 294
 description 289
 example 314
 option descriptions 291
 DSNZPARM
 option of START DB2 command 436
 DUMP option
 CANCEL THREAD command 160
 MODIFY irlmproc,ABEND command 341
 dump, IRLM diagnostic 343
 DWQT option of ALTER BUFFERPOOL command 63
 DYNAMICRULES option
 BIND PACKAGE subcommand 116
 DSNH command 309
 REBIND PACKAGE subcommand 116

E

EARLY
 option of REFRESH DB2 command 388
 EARLY option of START DB2 command 388
 ENABLE option
 BIND PACKAGE subcommand 122
 BIND PLAN subcommand 122
 DSNH command 309
 REBIND PACKAGE subcommand 122
 REBIND PLAN subcommand 122
 ENCODING option
 BIND PACKAGE subcommand 124
 BIND PLAN subcommand 124
 REBIND PACKAGE subcommand 124
 REBIND PLAN subcommand 124
 END subcommand of DSN
 description 319
 Example 319
 ENTRY option of DSNH command 300
 escape character
 APOST option of DCLGEN subcommand 171
 QUOTE option of DCLGEN subcommand 171
 establishing connections between IMS and a subsystem 423
 EXPLAIN option
 BIND PACKAGE subcommand 125
 BIND PLAN subcommand 125
 DSNH command 309, 312
 ONLY option 125
 REBIND PACKAGE subcommand 125
 REBIND PLAN subcommand 125
 REBIND TRIGGER PACKAGE subcommand 125
 extended MCS consoles, DB2 support of 7
 EXTENDEDINDICATOR option
 BIND PACKAGE subcommand 127
 REBIND PACKAGE subcommand 127

F

FLAG option
 BIND PACKAGE subcommand 128
 BIND PLAN subcommand 128
 DSNH command 300, 309
 FREE PACKAGE subcommand 323
 FREE PLAN subcommand 326
 REBIND PACKAGE subcommand 128

FLAG option (*continued*)
 REBIND PLAN subcommand 128
 REBIND TRIGGER PACKAGE subcommand 128
 FORCE option
 RESET INDOUBT command 392
 START DATABASE command 429
 STOP DB2 command 491
 STOP DDF command 493
 FORTLIB option of DSNH command 300
 FORTLOAD option of DSNH command 300
 FREE PACKAGE subcommand of DSN
 description 321
 option descriptions 322
 FREE PLAN subcommand of DSN
 description 325
 Example 326
 Option descriptions 325
 FREE QUERY subcommand of DSN
 description 327
 option descriptions 327
 functions, displaying information about 225

G

GBPCACHE option of ALTER GROUPBUFFERPOOL
 command 68
 GBPCHKPT option of ALTER GROUPBUFFERPOOL
 command 69
 GBPOOLT option of ALTER GROUPBUFFERPOOL
 command 69
 GDETAIL option of DISPLAY GROUPBUFFERPOOL
 command 235
 general-use programming information, described 539
 GENERIC option
 BIND PACKAGE subcommand 129
 REBIND PACKAGE subcommand 129
 GETACCELARCHIVE option
 BIND PACKAGE subcommand 130
 REBIND PACKAGE subcommand 130
 GLOBAL option of START irlmproc command 448
 group buffer pool RECOVER-pending (GRECP)
 status,removing using START DATABASE command 430
 GROUP option
 DSN command 286
 group, scope of command 17
 GTF option
 DISPLAY TRACE command 274
 START TRACE command 463
 STOP TRACE command 515

H

HOST option of DSNH command 300

I

I/O processing, parallel, DEGREE option of bind
 subcommands 114
 IBM Data Server clients and drivers 157
 IBMCOB option of DCLGEN subcommand 170
 ID option
 RECOVER INDOUBT command 380
 START RLIMIT command 457
 IFCID (instrumentation facility component identifier),
 identifiers by trace class 467

- IFCID option
 - MODIFY TRACE command 358
 - START TRACE command 470
- IMMEDWRITE option
 - BIND PACKAGE subcommand 131
 - BIND PLAN subcommand 131
 - REBIND PACKAGE subcommand 131
 - REBIND PLAN subcommand 131
- IMS
 - commands
 - CHANGE 165
 - DISPLAY 177
 - SSR 415
 - START 423
 - STOP 481
 - TRACE 527
 - facilities, events tracing 527
- IMS commands 8
- IMSBMP option
 - BIND and REBIND subcommands 123
 - DSNH command 310
- IMSMPP option
 - BIND and REBIND subcommands 123
 - DSNH command 310
- IMSPRE option of DSNH command 301
- INACTIVE option
 - DISPLAY THREAD command 258
- INCLUDE statement of DCLGEN subcommand output 172
- indicator variable, array declaration in DCLGEN 172
- INDOUBT option
 - DISPLAY THREAD command 258
- indoubt thread, recovering 379
- INDVAR option of DCLGEN subcommand 172
- information about 335
- INPUT option of DSNH command 301
- INTERVAL option of DISPLAY BUFFERPOOL command 189
- invalidated packages 127
- IPADDR option
 - DISPLAY LOCATION command 242
 - RESET INDOUBT command 393
- IRLM (internal resource lock manager)
 - commands
 - MODIFY irlmproc,ABEND 341
 - MODIFY irlmproc,DIAG 343
 - MODIFY irlmproc,PURGE option 345
 - MODIFY irlmproc,SET option 347
 - MODIFY irlmproc,STATUS option 351
 - START irlmproc 445
 - STOP irlmproc 501
 - TRACE CT 529
 - CSA
 - setting maximum amount of 347
 - delays, diagnosing 343
 - deregistering 341
 - diagnostic dumps 343
 - locks, releasing 345
 - modifying, diagnostic trace 529
 - restarting
 - effect on CSA value 349
 - starting
 - an IRLM component 445
 - diagnostic trace 529
 - status
 - checking 347
 - status, checking 351
 - stopping
 - diagnostic trace 529

- IRLM (internal resource lock manager) (*continued*)
 - stopping (*continued*)
 - normal 501
 - terminating
 - abnormal 341
 - normal 501
 - trace buffers, setting number of 347
- IRLM commands 9
- IRLMGRP option of START irlmproc command 446
- IRLMID option of START irlmproc command 446
- IRLMNM option of START irlmproc command 446
- ISOLATION
 - option of BIND PLAN subcommand
 - description 133
- ISOLATION option
 - BIND PACKAGE subcommand 133
 - DSNH command 310
 - REBIND PACKAGE subcommand 133
 - REBIND PLAN subcommand 133
 - REBIND TRIGGER PACKAGE subcommand 133

K

- KEEPDYNAMIC option
 - BIND PACKAGE subcommand 134
 - BIND PLAN subcommand 134
 - DSNH command 310
 - REBIND PACKAGE subcommand 134
 - REBIND PLAN subcommand 134

L

- LABEL option of DCLGEN subcommand 171
- LANGUAGE option of DCLGEN subcommand 170
- LEAVE option of DSNH command 307
- LIBRARY option
 - BIND PACKAGE subcommand 135
 - BIND PLAN subcommand 135
 - DCLGEN subcommand 169
 - RUN subcommand 396
- LIGHT option, START DB2 command 436
- light restart, with ARM 437
- LIMIT option
 - DISPLAY THREAD command 260
- LIMIT option of DISPLAY DATABASE command 207
- LINECOUNT option of DSNH command 301
- LINK option of DSNH command 301
- link-editing, processing 289
- links
 - non-IBM Web sites 539
- LIST option of DISPLAY BUFFERPOOL command 189
- LLIB option of DSNH command 301
- LnLIB option of DSNH command 301
- LOAD option
 - DSNH command 301
 - SET SYSPARM command 410
- LOAD utility, effects of TERM command 525
- LOCAL option
 - CANCEL THREAD command 159
- LOCAL option of START irlmproc command 448
- location name
 - BIND PACKAGE subcommand 123, 139
 - DISPLAY LOCATION command 242
 - REBIND PACKAGE subcommand 123, 139
 - REBIND TRIGGER PACKAGE subcommand 375

LOCATION option
 DISPLAY THREAD command 259
 DISPLAY TRACE command 273
 RESET INDOUBT command 392
 START TRACE command 465

location statistics 459

LOCKS option of DISPLAY DATABASE command 206

LOCKTABL option of START irlmproc command 446

LOG option of TRACE command 527

logical page list (LPL) 430

logical partitions, displaying 210

LOGLOAD option
 SET LOG command 404

LONGLOG option of ALTER UTILITY option 73

LOPTION option of DSNH command 301

LPL (logical page list)
 option of DISPLAY DATABASE command 206
 recovering pages
 using START DATABASE command 430

LSTATS option of DISPLAY BUFFERPOOL command 189

LTE option of MODIFY irlmproc,SET command of z/OS 348

LTE option of START irlmproc command 447

LUNAME option
 DISPLAY LOCATION command 242
 RESET INDOUBT command 392

LUWID option
 DISPLAY THREAD command 259
 RECOVER INDOUBT command 381
 RESET INDOUBT command 393

M

MACRO option of DSNH command 301

MAINT option of MODIFY irlmproc,STATUS command of z/OS 352

MAXCSA option of START irlmproc command 447

MAXRO option of ALTER UTILITY command 73

MAXUSRS option of START irlmproc command 447

MCS consoles, scope of commands 17

MDETAIL option of DISPLAY GROUPBUFFERPOOL command 235

MEMBER option
 BIND PACKAGE subcommand 136
 BIND PLAN subcommand 136
 DISPLAY UTILITY command 282

member, scope of command 17

MERGECOPY utility, effects of TERM command 525

message
 DB2 commands 7
 DCLGEN subcommand 170
 DISPLAY UTILITY command 282
 DSN command of TSO 287
 DSNH command 300
 FLAG option of bind subcommands 128
 FREE PACKAGE subcommand 323
 FREE PLAN subcommand 326
 MODIFY irlmproc,STATUS command 353
 RUN subcommand 397

message by identifier
 DSN7100I 230
 DSN9022I 7
 DSN9023I 7
 DSNJ315I 78
 DSNJ316I 78
 DSNJ317I 78
 DSNJ318I 78
 DSNL440I to DSNL449I 393

message by identifier (*continued*)
 DSNL448I 392
 DSNL450I 160
 DSN736I 487
 DSNU100I 282
 DSNU105I 282
 DSNU106I 282
 DSNW133I 462

MLT option of MODIFY irlmproc,SET command of z/OS 348

MODE option
 ARCHIVE LOG command 76
 STOP DB2 command 491
 STOP DDF command 493

MODIFY admtproc
 shutdown 331
 Trace 333

MODIFY admtproc command of z/OS
 Examples 333

MODIFY admtproc,,APPL=SHUTDOWN command of z/OS
 description 331

MODIFY admtproc,,APPL=TRACE command of z/OS
 description 333

MODIFY admtproc,APPL=SHUTDOWN command of z/OS
 Examples 331

MODIFY DDF 335

MODIFY DDF command 335

MODIFY irlmproc,ABEND command of z/OS
 description 341
 Example 342

MODIFY irlmproc,DIAG command of z/OS
 description 343
 Example 344

MODIFY irlmproc,PURGE command of z/OS
 description 345
 Example 345

MODIFY irlmproc,SET command of z/OS
 description 347
 example 349
 option descriptions 345, 347

MODIFY irlmproc,STATUS command of z/OS
 description 351
 example 353
 Option descriptions 351

MODIFY RECOVERY utility, effects of TERM command 525

MODIFY STATISTICS utility, effects of TERM command 525

MODIFY TRACE command
 description 357
 Example 359
 modifying 335

MONITOR option
 DISPLAY TRACE command 274
 MODIFY TRACE command 358
 STOP TRACE command 515

monitor trace
 displaying 271
 starting 459
 stopping 511

MSTR option of START DB2 command 436

N

NAMES option of DCLGEN subcommand 170

NEWFUN option of DSNH command 301

NEWLOG option
 SET LOG command 405

NID (network ID) option of RECOVER INDOUBT command 381

- NO LIMIT option of SET ARCHIVE command 400
- NO option
 - START irlmproc command 448
- NOBACKOUT option of CANCEL THREAD command 160
- NODEFER option
 - BIND PACKAGE subcommand 113
 - BIND PLAN subcommand 113
 - DSNH command 310
 - REBIND PACKAGE subcommand 113
 - REBIND PLAN subcommand 113
- NODISCON option of START irlmproc command 448
- NODUMP option of MODIFY irlmproc, ABEND command 341
- NOFOR option of DSNH command 302
- NOINDOUBTS option
 - START DB2 command 435
- NONE keyword of MODIFY irlmproc, DIAG command 344
- NOWRAP option of TRACE CT command 530

O

- OASN option
 - CHANGE command 165
 - DISPLAY command 177
- ONLY option of DISPLAY DATABASE command 206
- OP option
 - START TRACE command 463
 - STOP TRACE command 515
- operands, DB2 commands 11
- OPTHINT option
 - BIND PACKAGE subcommand 137
 - BIND PLAN subcommand 137
 - DSNH command 310
 - REBIND PACKAGE subcommand 137
 - REBIND PLAN subcommand 137
- options for starting 46
- OPTIONS option
 - BIND PACKAGE subcommand 138
 - DSNH command 302
- OUTNAME option of DSNH command 302
- output size 19
- OVERVIEW option of DISPLAY DATABASE command 207
- OWNER option
 - BIND PACKAGE subcommand 138
 - BIND PLAN subcommand 138
 - DCLGEN subcommand 169
 - DSNH command 310
 - REBIND PACKAGE subcommand 138
 - REBIND PLAN subcommand 138

P

- P irlmproc command. 501
- package
 - binding, initiating 81
 - identifier
 - BIND PACKAGE subcommand 139
 - REBIND TRIGGER PACKAGE subcommand 375
 - rebinding 361
 - rebinding trigger 373
 - replacing version of 103
- PACKAGE option of DSNH command 312
- PACTION option of DSNH command 311
- parallel processing
 - DEGREE option of bind subcommands 114
 - VPPSEQT option of ALTER BUFFERPOOL command 62

- parameter, passing to application program 396
- PARAM option of START DB2 command 436
- PARMS option
 - DSNH command 302
 - RUN subcommand 396
- parsing rules, DB2 commands 11
- PART option
 - DISPLAY DATABASE command 207
 - START DATABASE command 428
 - STOP DATABASE command 486
- partial-location name, DISPLAY LOCATION command 242
- PASS option of DSNH command 302
- PATH option
 - BIND PACKAGE subcommand 141
 - BIND PLAN subcommand 141
 - DSNH command 310
 - REBIND PACKAGE subcommand 141
 - REBIND PLAN subcommand 141
- PATHDEFAULT option
 - REBIND PACKAGE subcommand 142
 - REBIND PLAN subcommand 142
- PBIND option of DSNH command 311
- PC option of START irlmproc command 448
- PCICS option of DSNH command 311
- PCLOAD option of DSNH command 302
- PDBPROTOCOL option of DSNH command 311
- PDBRMLIB option of DSNH command 311
- PDEFER option of DSNH command 311
- PDEGREE option of DSNH command 311
- PDISABLE option of DSNH command 311
- PDLIBATCH option of DSNH command 311
- PDMEM option of DSNH command 312
- PDYNAMICRULES option of DSNH command 312
- PENABLE option of DSNH command 312
- PERFM option
 - DISPLAY TRACE command 274
 - MODIFY TRACE command 358
 - STOP TRACE command 514
- performance trace
 - displaying 271
 - stopping 511
- performance, trace
 - starting 459
- PFLAG option of DSNH command 312
- PGPROT option of START irlmproc command 448
- PGSTEAL option of ALTER BUFFERPOOL command 63
- phases of execution for DSNH processing 289
- PIMSBMP option of DSNH command 312
- PIMSMPP option of DSNH command 312
- PISOLATION option of DSNH command 312
- PKEEPDYNAMIC option of DSNH command 312
- PKGCOL option
 - DISPLAY TRACE command 273
 - START TRACE command 464
 - STOP TRACE command 514
- PKGLOC option
 - DISPLAY TRACE command 273
 - START TRACE command 464
 - STOP TRACE command 514
- PKGPROG option
 - DISPLAY TRACE command 273
 - START TRACE command 464
 - STOP TRACE command 514
- PKLIST option
 - BIND PLAN subcommand 143
 - DSNH command 310
 - REBIND PLAN subcommand 143

- PL/I application program, macro processing step for DSNH 289
- PLAN
 - option of DSNH command 302
- PLAN option
 - BIND PLAN subcommand 144
 - DISPLAY TRACE command 273
 - DSNH command 310
 - REBIND PLAN subcommand 144
 - RUN subcommand 396
 - START TRACE command 464
 - STOP TRACE command 514
- PLANMGMT option
 - BIND PACKAGE subcommand 144
 - REBIND PACKAGE subcommand 144
 - REBIND TRIGGER PACKAGE subcommand 144
- PLANMGMTSCOPE option
 - FREE PACKAGE subcommand 323
- PLI option of DCLGEN subcommand 170
- PLI2LIB option of DSNH command 303
- PLIB option of DSNH command 302
- PLILIB option of DSNH command 303
- PLILOAD option of DSNH command 303
- PLIPLNK option of DSNH command 303
- PLIPMSG option of DSNH command 303
- PLOCK keyword of MODIFY irlmproc,DIAG command 343
- PnLIB option of DSNH command 302
- PNODEFER option of DSNH command 312
- POPTHINT option of DSNH command 312
- POPTION option of DSNH command 303
- POSTPONED option
 - DISPLAY THREAD command 258
- postponed units of recovery, recovering 383
- POWNER option of DSNH command 312
- PPATH option of DSNH command 312
- PQUALIFIER option of DSNH command 312
- PRECOMP option of DSNH command 303
- precompiler
 - DSNH command options 302
 - invoking DSNH 289
 - producing members for 136
- PRELEASE option of DSNH command 313
- PRELINK option of DSNH command 303
- PREOPT option of DSNH command 313
- PRINT option of DSNH command 304
- privileges
 - commands 3
- PROC option
 - DISPLAY THREAD command 258
- processing, parallel
 - VPPSEQT option of ALTER BUFFERPOOL command 62
- product-sensitive programming information, described 539
- PROGRAM option of RUN subcommand 395
- programming interface information, described 539
- PSECSPAC option of DSNH command 304
- PSPACE option of DSNH command 304
- PSPI symbols 539
- PVALIDATE option of DSNH command 313
- PVT option of MODIFY irlmproc,SET command of z/OS 348

Q

- QUALIFIER option
 - BIND PACKAGE subcommand 146
 - BIND PLAN subcommand 146
 - DSNH command 310
 - REBIND PACKAGE subcommand 146

- QUALIFIER option (*continued*)
 - REBIND PLAN subcommand 146
- query
 - binding 97
 - removal 327
- QUERYACCELERATION option
 - BIND PACKAGE subcommand 146
 - REBIND PACKAGE subcommand 146
- QUERYID option
 - FREE QUERY subcommand 148
- QUEUE option of STOP FUNCTION SPECIFIC 498, 499
- QUIESCE option
 - STOP DB2 command 491
 - STOP DDF command 493
- QUIESCE utility, effects of TERM command 525
- QUOTE option
 - DCLGEN subcommand 171
 - DSNH command 300

R

- RATIO option of ALTER GROUPBUFFERPOOL command 68
- RCTERM option of DSNH command 304
- REBIND PACKAGE subcommand of DSN
 - description 361
 - example 364
 - option descriptions 101, 154
- REBIND PLAN subcommand of DSN
 - description 367
 - option descriptions 101, 154
- REBIND TRIGGER PACKAGE subcommand of DSN
 - description 373
 - example 375
 - option descriptions 101, 154
- rebinding
 - initiating 361, 367
 - options for 101, 154
- REBUILD INDEX utility, TERM command effects 525
- recognition character 11
- recorded in DB2 recovery log 51
- RECOVER BSDS command
 - description 377
 - Example 377
- RECOVER INDOUBT command
 - description 379
 - example 382
 - Option descriptions 380
- RECOVER POSTPONED command
 - description 383
 - Example 384
- RECOVER utility, TERM command effects 525
- recovery
 - BSDS 377
 - indoubt threads 379
 - postponed units of recovery 383
- REFRESH DB2 command
 - example 388
- REFRESH DB2, EARLY command
 - description 387
- REFRESH DB2,EARLY command
 - Option descriptions 388
- REGISTER XMLSCHEMA command 42
- REJECT option of STOP FUNCTION SPECIFIC 498, 499
- RELEASE option
 - BIND PACKAGE subcommand 149
 - BIND PLAN subcommand
 - description 149

RELEASE option (*continued*)
 DSNH command 310
 REBIND PACKAGE subcommand 149
 REBIND PLAN subcommand 149
 REBIND TRIGGER PACKAGE subcommand 149
 RELOAD option of SET SYSPARM command 410
 REMOTE option of DSNH command 313
 REMOVE XMLSCHEMA command 44
 REOPT option
 BIND PACKAGE subcommand 151, 152
 BIND PLAN subcommand 151, 152
 DSNH command 310
 REBIND PACKAGE subcommand 151, 152
 REBIND PLAN subcommand 151, 152
 REORG INDEX utility, effects of TERM command 525
 REORG TABLESPACE utility, effects of TERM command 525
 REPAIR utility, effects of TERM command 525
 REPLACE option
 DCLGEN subcommand 170
 DSNH command 308, 311
 replacing, version of a package 103
 REPLVER option
 BIND PACKAGE subcommand 102
 DSNH command 313
 effect of 103
 REPORT utility, effects of TERM command 525
 reports
 summary report
 DISPLAY GROUPBUFFERPOOL command 235
 RES option of STOP TRACE command 515
 RESET GENERICLU command
 description 389
 example 390
 Option descriptions 389
 RESET INDOUBT command
 description 391
 Option descriptions 392
 RESET option of CHANGE command 166
 resource limit facility
 displaying 210
 restarting
 connections between IMS and a subsystem 423
 status of DB2 resources 387, 435
 terminated utility job steps 524
 RESTRICT option of DISPLAY DATABASE command 207
 RESUME option
 SET LOG command 405
 RETAIN option
 BIND PLAN subcommand 102
 DSNH command 310
 retained locks 206
 RETRY option
 DSN command 286
 return code
 DSN command 287
 RUN subcommand of DSN 287
 RMARGIN option of DCLGEN subcommand 172
 RMID option
 DISPLAY TRACE command 275
 START TRACE command 462
 STOP TRACE command 515
 RO option of START DATABASE command 429
 ROLE option
 DISPLAY TRACE command 273
 START TRACE command 467
 STOP TRACE command 514
 RREPL option of START DATABASE command 429
 RRSURID option of DISPLAY THREAD command 260
 RUN
 subcommand of DSN
 description 395
 example 397
 Option descriptions 395
 return codes 287
 RUN option
 DSNH command parameters 305
 RUNIN option of DSNH command 305
 running DSNH processing 289
 RUNOUT option of DSNH command 305
 RUNSTATS utility, effects of TERM command 525
 RW option of START DATABASE command 429

S

scanning rules, DB2 commands 11
 schema.partial-name option
 DISPLAY FUNCTION SPECIFIC command 226
 START FUNCTION SPECIFIC command 442
 STOP FUNCTION SPECIFIC command 498
 schema.specific-function-name option
 DISPLAY FUNCTION SPECIFIC command 226
 START FUNCTION SPECIFIC command 442
 STOP FUNCTION SPECIFIC command 498
 schema.specific-function-name option of DISPLAY FUNCTION SPECIFIC command 226
 scope of commands 17
 SCOPE option
 ARCHIVE LOG command 77
 DISPLAY FUNCTION SPECIFIC command 226
 DISPLAY PROCEDURE command 248
 DISPLAY THREAD command 257
 DISPLAY TRACE command 274
 START FUNCTION SPECIFIC command 442
 START irlmproc command 448
 START PROCEDURE command 452
 STOP FUNCTION SPECIFIC command 498
 STOP PROCEDURE command 505
 STOP TRACE command 515
 SET ARCHIVE command
 description 399
 example 401
 Option descriptions 400
 SET LOG command
 description 403
 example 407
 Option descriptions 404
 SET SYSPARM command
 description 409
 example 410
 Option descriptions 410
 shortcut keys
 keyboard x
 SINGLE option
 SET LOG command 404
 SMF option
 DISPLAY TRACE command 274
 START TRACE command 463
 STOP TRACE command 515
 SOMDLLI option of DSNH command 305
 SOURCE option of DSNH command 305
 SPACENAM option
 DISPLAY DATABASE command 204
 START DATABASE command 427
 STOP DATABASE command 485

- SPACEUN option of DSNH command 306
- SPUFI, description 413
- SQL option of DSNH command 306
- SQLDELIM option of DSNH command 306
- SQLERROR option
 - BIND PACKAGE subcommand 153
 - DSNH command 313
- SQLRULES option
 - BIND PLAN subcommand 154
 - REBIND PLAN subcommand 154
- SQLRULES option of DSNH command 310
- SRC (subsystem recognition character) 11
- SRV option
 - DISPLAY TRACE command 274
 - START TRACE command 463
 - STOP TRACE command 515
- SSR command of IMS, description 415
- START ACCEL command 417
 - Option descriptions 418
- START admtproc command of z/OS
 - description 421
 - Examples 422
- START command of IMS 423
- START DATABASE command
 - description 425
 - example 432
 - Option descriptions 427
 - recovering object in group buffer pool 430
 - recovering pages on logical page list 430
- START DB2 command
 - description 435
 - example 438
 - Option descriptions 436
- START DDF command 439
- START FUNCTION SPECIFIC command
 - description 441
 - example 443
 - Option descriptions 442
- START irlmproc command of z/OS
 - description 445
 - Examples 448
 - Option descriptions 445
- START PROCEDURE command
 - description 451
 - example 453
 - Option descriptions 452
- START PROFILE command
 - example 455
- START RLIMIT command
 - description 457
 - Example 458
- START TRACE command
 - description 459
 - example 475
 - Option descriptions 461
- STARTUP option of SET SYSPARM command 410
- STAT option
 - DISPLAY TRACE command 274
 - MODIFY TRACE command 358
 - STOP TRACE command 514
- statistics trace
 - stopping 511
- statistics, trace
 - displaying 271
 - starting 459
- status
 - checking, IRLM 351
 - status (*continued*)
 - cross-system coupling facility (XCF), status of members 230
- STDSQL option of DSNH command 306
- STOP ACCEL command
 - description 477
 - example 478
- STOP admtproc command of z/OS
 - description 479
 - Examples 479
- STOP command of IMS 481
- STOP DATABASE command
 - description 483
 - example 488
 - Option descriptions 484
- STOP DB2 command
 - description 491
 - Example 492
- STOP DDF command
 - description 493
 - example 495
- STOP FUNCTION SPECIFIC command
 - description 497
 - Examples 499
 - limitations of 499
 - Option descriptions 498
- STOP irlmproc command of z/OS 501
- STOP PROCEDURE command
 - description 503
 - example 505
 - Option descriptions 504
- STOP PROFILE command
 - example 507
- STOP RLIMIT command 509
- STOP TRACE command
 - description 511
 - example 520
 - Option descriptions 514
- STOR option of MODIFY irlmproc,STATUS command of z/OS 352
- stored procedure
 - displaying status 247
 - starting 451
 - stopping 503
- STOSPACE utility, effects of TERM command 525
- string, delimiter
 - COBOL 171
 - SQL 171
- STRUCTURE option of DCLGEN subcommand 170
- SUB option of TRACE CT command 530
- SUBS option of TRACE command 527
- SUBSYS option
 - CHANGE command 165
 - DISPLAY command 177
 - START command 423
 - STOP command 481
- SUFFIX option of DSNH command 306
- summary report
 - DISPLAY GROUPBUFFERPOOL command 235
- SUSPEND option
 - SET LOG command 405
 - STOP DDF command 494
- SWITCH option
 - REBIND PACKAGE subcommand 155
 - REBIND TRIGGER PACKAGE subcommand 155
- syntax diagram
 - how to read xi

SYSTEM option
 DISPLAY THREAD command 258
 DSN command 286
 DSNH command 306

T

TABLE option of DCLGEN subcommand 168
 TDATA option of START TRACE command 471
 TERM option of DSNH command 307
 TERM UTILITY command
 description 523
 example 525

terminating
 connections between IMS and a subsystem 481
 databases 483
 DB2, description 491
 IRLM
 abnormal 341
 normal 501
 stored procedures 503
 trace activity 511
 utilities, description 523

TEST option
 DSN command 286

thread
 ACTIVE option
 DISPLAY THREAD command 255
 canceling 159
 displaying 255
 message
 DISPLAY THREAD with ACTIVE 255

TIME option
 ARCHIVE LOG command 76
 DSNH command 307
 SET ARCHIVE command 400

TIMEOUT option of MODIFY irlmproc,SET command of z/OS 348

TNO option
 DISPLAY TRACE command 275
 MODIFY TRACE command 358

trace
 changing active traces 357
 displaying 271
 events 527
 IFCIDs activated by trace class 467
 starting 459
 stopping 511

TRACE command of IMS
 description 527
 example 527

TRACE CT command of z/OS
 description 529
 example 531
 Option descriptions 529

TRACE option
 MODIFY irlmproc,SET command of z/OS 348
 MODIFY irlmproc,STATUS command of z/OS 352
 START irlmproc command 448
 START TRACE command 472

TRACK option of DSNH 306

TSO CLIST commands 10

TSO CLISTs of DSNH 289

TSO option of DSNH command 305

TYPE option
 DISPLAY GROUPBUFFERPOOL command 234
 DISPLAY THREAD command 257

U

unit of recovery, in IMS 178

unit of work
 displaying an outstanding 177
 resetting
 IMS 165
 indoubt 165

UNLOAD utility, effects of TERM command 525

USE option of DISPLAY DATABASE command 205

USERID option
 DISPLAY TRACE command 273
 START TRACE command 466
 STOP TRACE command 514

UT option of START DATABASE command 429

utilities
 displaying status 281
 identifier 524
 terminating 523

V

VALIDATE option
 BIND PACKAGE subcommand 156
 BIND PLAN subcommand 156
 DSNH command 310
 REBIND PACKAGE subcommand 156
 REBIND PLAN subcommand 156

VDWQT option of ALTER BUFFERPOOL command 63

version of a package
 BIND PACKAGE subcommand 139
 REBIND PACKAGE subcommand 139

VERSION option of DSNH command 307

VPSEQT option of ALTER BUFFERPOOL command 62

VPSEQT option of ALTER BUFFERPOOL command 62

VPSIZE option of ALTER BUFFERPOOL command 62

VPXPSEQT option of ALTER BUFFERPOOL command 62

VSAM (virtual storage access method) password, DCLGEN subcommand 169

VTAM (Virtual Telecommunications Access Method), DISPLAY NET command 161

W

WAIT option of ARCHIVE LOG command 77

WEPR option of DISPLAY DATABASE command 206

WORKUNIT option of DSNH command 307

WRAP option of TRACE CT command 529

WRKSTN option
 DISPLAY TRACE command 273
 START TRACE command 466
 STOP TRACE command 514

WSECPAC option of DSNH command 307

WSPACE option of DSNH command 307

WTRSTART option of TRACE CT command 529

WTRSTOP option of TRACE CT command 530

X

XAPPNAME option
 DISPLAY TRACE command 273
 START TRACE command 466
 STOP TRACE command 514

XAUTHID option
 DISPLAY TRACE command 273
 START TRACE command 464

XAUTHID option (*continued*)
 STOP TRACE command 514
 XCF (cross-system coupling facility), status of members 230
 XCONNID option
 DISPLAY TRACE command 273
 START TRACE command 466
 STOP TRACE command 514
 XCORRID option
 DISPLAY TRACE command 273
 START TRACE command 467
 STOP TRACE command 514
 XLIB option of DSNH command 307
 XLOC option
 START TRACE command 273, 465
 XPKGCOL option
 DISPLAY TRACE command 273
 START TRACE command 464
 STOP TRACE command 514
 XPKGLOC option
 DISPLAY TRACE command 273
 START TRACE command 464
 STOP TRACE command 514
 XPKGPROG option
 DISPLAY TRACE command 273
 START TRACE command 464
 STOP TRACE command 514
 XPLAN option
 DISPLAY TRACE command 273
 START TRACE command 464
 STOP TRACE command 514
 XREF option of DSNH command 307
 XROLE option
 DISPLAY TRACE command 273
 START TRACE command 467
 STOP TRACE command 514
 XUSERID option
 DISPLAY TRACE command 273
 START TRACE command 466
 STOP TRACE command 514
 XWRKSTN option
 DISPLAY TRACE command 273
 START TRACE command 466
 STOP TRACE command 514

Y

YES option
 START DB2 command 436
 START irlmproc command 448

Z

z/OS commands 331, 333
 MODIFY irlmproc,ABEND 341
 MODIFY irlmproc,DIAG 343
 MODIFY irlmproc,PURGE 345
 MODIFY irlmproc,SET 347
 MODIFY irlmproc,STATUS 351
 START admtproc 421
 START irlmproc 445
 STOP admtproc 479
 STOP irlmproc 501
 TRACE CT 529



Product Number: 5605-DB2
5697-P31

Printed in USA

SC19-2972-13



Spine information:

DB2 10 for z/OS

Command Reference

