

DB2 10 for z/OS

*Command Reference*





DB2 10 for z/OS

*Command Reference*



**Note**

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

**Sixth edition (October 2012)**

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, 2012.

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

---

# Contents

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

---

## Part 1. Privileges, authorization IDs, and the bind process . . . . . 1

### Chapter 1. Privileges and authorization IDs . . . . . 3

### Chapter 2. The bind process . . . . . 5

---

## Part 2. Description of DB2 and related commands . . . . . 7

### Chapter 3. Types of commands. . . . . 9

The DSN command and its subcommands	9
DB2 commands	10
IMS commands	11
CICS attachment facility commands	11
Administrative task scheduler commands	12
z/OS IRLM commands	12
TSO CLISTs	13

### Chapter 4. DB2 command parsing . . . . . 15

Characters with special meanings	16
Examples of keyword entry	18

### Chapter 5. DSN subcommand parsing . . . . . 19

### Chapter 6. Scope of DB2 and related commands . . . . . 21

### Chapter 7. Output from DB2 commands . . . . . 23

### Chapter 8. Issuing commands to DB2 from IFI. . . . . 25

### Chapter 9. Command line processor . . . . . 27

### | Chapter 10. DB2 command text is recorded in the DB2 recovery log . . . . . 45

---

## Part 3. DB2 and related commands . . . . . 47

### Chapter 11. -ACCESS DATABASE (DB2) . . . . . 49

### Chapter 12. -ALTER BUFFERPOOL (DB2) . . . . . 53

### Chapter 13. -ALTER GROUPBUFFERPOOL (DB2) . . . . . 59

### Chapter 14. -ALTER UTILITY (DB2). . . . . 63

Chapter 15. -ARCHIVE LOG (DB2) . . . . .	67
Chapter 16. BIND PACKAGE (DSN). . . . .	73
Chapter 17. BIND PLAN (DSN). . . . .	81
Chapter 18. BIND QUERY (DSN) . . . . .	87
Chapter 19. BIND and REBIND options . . . . .	91
Chapter 20. -CANCEL THREAD (DB2) . . . . .	143
Chapter 21. /CHANGE (IMS) . . . . .	149
Chapter 22. DCLGEN (DECLARATIONS GENERATOR) (DSN) . . . . .	151
Chapter 23. /DISPLAY (IMS) . . . . .	161
Chapter 24. -DISPLAY ACCEL (DB2). . . . .	165
Chapter 25. -DISPLAY ARCHIVE (DB2). . . . .	171
Chapter 26. -DISPLAY BUFFERPOOL (DB2) . . . . .	173
Chapter 27. -DISPLAY DATABASE (DB2) . . . . .	189
Chapter 28. -DISPLAY DDF (DB2) . . . . .	209
Chapter 29. -DISPLAY FUNCTION SPECIFIC (DB2) . . . . .	213
Chapter 30. -DISPLAY GROUP (DB2) . . . . .	219
Chapter 31. -DISPLAY GROUPBUFFERPOOL (DB2) . . . . .	227
Chapter 32. -DISPLAY LOCATION (DB2) . . . . .	241
Chapter 33. -DISPLAY LOG (DB2). . . . .	247
Chapter 34. -DISPLAY PROCEDURE (DB2) . . . . .	249
Chapter 35. -DISPLAY PROFILE (DB2). . . . .	255
Chapter 36. -DISPLAY RLIMIT (DB2). . . . .	257
Chapter 37. -DISPLAY THREAD (DB2) . . . . .	259
Chapter 38. -DISPLAY TRACE (DB2). . . . .	293
Chapter 39. -DISPLAY UTILITY (DB2) . . . . .	305
Chapter 40. DSN (TSO) . . . . .	311
Chapter 41. DSNC (CICS attachment facility) . . . . .	315

Chapter 42. DSNB DISCONNECT (CICS attachment facility)	317
Chapter 43. DSNB DISPLAY (CICS attachment facility)	319
Chapter 44. DSNB MODIFY (CICS attachment facility)	323
Chapter 45. DSNB STOP (CICS attachment facility)	325
Chapter 46. DSNB STRT (CICS attachment facility)	327
Chapter 47. DSNH (TSO CLIST)	329
Chapter 48. END (DSN)	359
Chapter 49. FREE PACKAGE (DSN)	361
Chapter 50. FREE PLAN (DSN)	365
Chapter 51. FREE QUERY (DSN)	367
Chapter 52. MODIFY admtproc,APPL=SHUTDOWN	371
Chapter 53. MODIFY admtproc,APPL=TRACE	373
Chapter 54. -MODIFY DDF (DB2)	375
Chapter 55. MODIFY irlmproc,ABEND (z/OS IRLM)	379
Chapter 56. MODIFY irlmproc,DIAG (z/OS IRLM)	381
Chapter 57. MODIFY irlmproc,PURGE (z/OS IRLM)	383
Chapter 58. MODIFY irlmproc,SET (z/OS IRLM)	385
Chapter 59. MODIFY irlmproc,STATUS (z/OS IRLM)	389
Chapter 60. -MODIFY TRACE (DB2)	401
Chapter 61. REBIND PACKAGE (DSN)	405
Chapter 62. REBIND PLAN (DSN)	411
Chapter 63. REBIND TRIGGER PACKAGE (DSN)	417
Chapter 64. -RECOVER BSDS (DB2)	421
Chapter 65. -RECOVER INDOUBT (DB2)	423
Chapter 66. -RECOVER POSTPONED (DB2)	427
Chapter 67. -REFRESH DB2,EARLY (DB2)	429
Chapter 68. -RESET GENERICLU (DB2)	431

Chapter 69. -RESET INDOUBT (DB2) . . . . .	433
Chapter 70. RUN (DSN) . . . . .	437
Chapter 71. -SET ARCHIVE (DB2). . . . .	441
Chapter 72. -SET LOG (DB2) . . . . .	445
Chapter 73. -SET SYSPARM (DB2) . . . . .	451
Chapter 74. SPUFI (DSN) . . . . .	453
Chapter 75. /SSR (IMS) . . . . .	455
Chapter 76. -START ACCEL (DB2) . . . . .	457
Chapter 77. START admtproc . . . . .	461
Chapter 78. /START (IMS) . . . . .	463
Chapter 79. -START DATABASE (DB2). . . . .	465
Chapter 80. -START DB2 (DB2) . . . . .	473
Chapter 81. -START DDF (DB2). . . . .	477
Chapter 82. -START FUNCTION SPECIFIC (DB2) . . . . .	479
Chapter 83. START irlmproc (z/OS IRLM) . . . . .	483
Chapter 84. -START PROCEDURE (DB2). . . . .	489
Chapter 85. -START PROFILE (DB2). . . . .	493
Chapter 86. -START RLIMIT (DB2). . . . .	495
Chapter 87. -START TRACE (DB2) . . . . .	497
Chapter 88. -STOP ACCEL (DB2) . . . . .	515
Chapter 89. STOP admtproc (z/OS ) . . . . .	517
Chapter 90. /STOP (IMS). . . . .	519
Chapter 91. -STOP DATABASE (DB2) . . . . .	521
Chapter 92. -STOP DB2 (DB2) . . . . .	529
Chapter 93. -STOP DDF (DB2) . . . . .	531
Chapter 94. -STOP FUNCTION SPECIFIC (DB2) . . . . .	535
Chapter 95. STOP irlmproc (z/OS IRLM) . . . . .	539



<b>Chapter 96. -STOP PROCEDURE (DB2)</b>	<b>541</b>
<b>Chapter 97. STOP PROFILE (DB2)</b>	<b>545</b>
<b>Chapter 98. -STOP RLIMIT (DB2)</b>	<b>547</b>
<b>Chapter 99. -STOP TRACE (DB2)</b>	<b>549</b>
<b>Chapter 100. -TERM UTILITY (DB2)</b>	<b>559</b>
<b>Chapter 101. /TRACE (IMS)</b>	<b>563</b>
<b>Chapter 102. TRACE CT (z/OS IRLM)</b>	<b>565</b>
<hr/> <b>Part 4. Appendixes</b>	<hr/> <b>569</b>
<b>Information resources for DB2 for z/OS and related products</b>	<b>571</b>
<b>How to obtain DB2 information.</b>	<b>577</b>
<b>How to use the DB2 library</b>	<b>579</b>
<b>Notices</b>	<b>583</b>
Programming Interface Information	584
Trademarks	585
<b>Glossary</b>	<b>587</b>
<b>Index</b>	<b>633</b>



---

## 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.

This information assumes that your DB2® subsystem is running in Version 10 new-function mode. Generally, new functions that are described, including changes to existing functions, statements, and limits, are 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 Versions 8 and 9, most utility functions were available in conversion mode. However, for Version 10, most utility functions work only 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

**Important:** In this version of DB2 for z/OS®, the DB2 Utilities Suite 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.

The DB2 Utilities Suite can work with DB2 Sort and the DFSORT program, which you are licensed to use 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 information**

DB2 utilities packaging (Utility Guide)

---

## 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.

**OMEGAMON®**

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 on 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 Information Management Software for z/OS Solutions Information Center (which includes information for DB2 10 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**

You can access DB2 10 for z/OS ISPF panel functions by using a keyboard or keyboard shortcut keys.

For information about navigating the DB2 10 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**

Online documentation for DB2 10 for z/OS is available in the Information Management Software for z/OS Solutions Information Center, which is available at the following website: <http://publib.boulder.ibm.com/infocenter/imzic>

## 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. You can use the following methods to provide comments:

- Send your comments by email to [db2zinfo@us.ibm.com](mailto: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).
- You can also send comments by using the **Feedback** link at the footer of each page in the Information Management Software for z/OS Solutions Information Center at <http://publib.boulder.ibm.com/infocenter/imzic>.

---

## 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 ►— symbol indicates the beginning of a statement.

The —► symbol indicates that the statement syntax is continued on the next line.

The ►— symbol indicates that a statement is continued from the previous line.

The —► symbol indicates the end of a statement.

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

►—required\_item—►

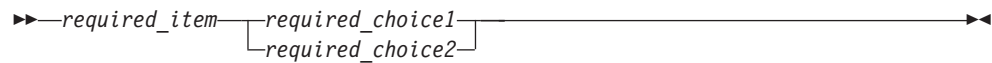
- Optional items appear below the main path.

►—required\_item—  
                    └optional\_item┘—►

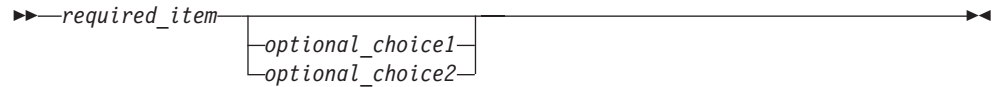
If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

►—required\_item—  
                    └optional\_item┘—►

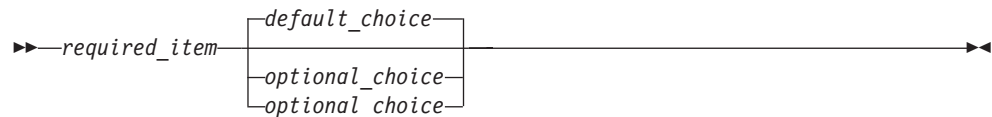
- If you can choose from two or more items, they appear vertically, in a stack. If you *must* choose one of the items, one item of the stack appears on the main path.



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



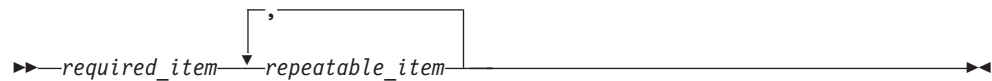
If one of the items is the default, it appears above the main path and the remaining choices are shown below.



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

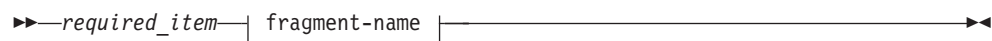


If the repeat arrow contains a comma, you must separate repeated items with a comma.

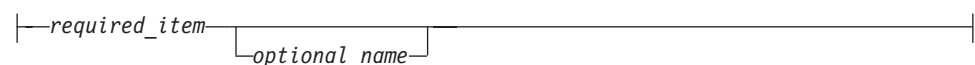


A repeat arrow above a stack indicates that you can repeat the items in the stack.

- 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.

---

## **Part 1. Privileges, authorization IDs, and the bind process**

These topics contain information about the privileges and authorization IDs that are required to issue the various commands used for DB2 and a summary of the bind process.





---

## Chapter 1. Privileges and authorization IDs

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

A privilege is the capability of performing a specific function, sometimes on a specific object. The types of privileges are:

**explicit privileges**

which have names and are held as the result of SQL GRANT and REVOKE statements. For example, the SELECT privilege.

**implicit privileges**

which accompany the ownership of an object, such as the privilege to drop a synonym that one owns, or the holding of an authority, such as the privilege of SYSADM authority to terminate any utility job.

An authorization ID is a string that can be verified for connection to DB2 and to which a set of privileges is allowed. It can represent an individual, an organizational group, or a function, but DB2 does not determine this representation.

The issuer of a command can be an individual user. It can also be a program running in batch mode or an IMS or CICS transaction. The term *process* is used to represent any or all of those. *The privilege set of a process* means the entire set of privileges and authorities that can be used by the process in a specific situation.

There are three types of identifiers: primary authorization IDs, secondary authorization IDs, and SQL IDs.

- Generally the primary authorization ID identifies a specific process. For example, in the process initiated through the TSO attachment facility, the primary authorization ID is identical to the TSO logon ID. A trace record identifies the process by the primary authorization ID.

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.

- Secondary authorization IDs, which are optional, can hold additional privileges that are available to the process. A secondary authorization ID is often a Resource Access Control Facility (RACF®) group ID. For example, a process can belong to a RACF group that holds the LOAD privilege on a particular database. Any member of the group can run the LOAD utility to load table spaces into the database.

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.

- An SQL authorization ID (SQL ID) holds the privileges that are exercised when a process issues certain dynamic SQL statements. This ID does not effect most DB2 and related commands.

Within DB2, a process can be represented by a primary authorization ID and possibly by one or more secondary IDs.

A privilege or authority is granted to, or revoked from, an identifier by executing an SQL GRANT or REVOKE statement.

*Authorization for trusted context:* You can use DB2 authorization to check DB2 commands 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.

---

## Chapter 2. The bind process

The bind process establishes a relationship between an application program and its relational data. This process is necessary before you can execute your program.

During the precompilation process, the DB2 precompiler produces both modified source code and a database request module (DBRM) for each application program. The modified source code must be compiled and link-edited before the program can be run. DBRMs must be bound to a package. You can then associate that package with a particular application plan.

When determining the maximum size of a plan, you must consider several physical limitations, including the time required to bind the plan, the size of the EDM pool, and fragmentation. As a general rule, the EDM pool should be at least 10 times the size of the largest DBD or plan, whichever is greater.

To bind individual DBRMs into packages, use the BIND PACKAGE subcommand. Packages provide the flexibility for you to test different versions of a program without having to rebind everything in the application plan.

All packages must be designated in an application plan. Use the BIND PLAN command to build such an application plan, allocate resources for the plan, and specify which packages are associated with that plan. Plans can specify packages, collections of packages, or a combination of these elements. If you specify one or more DBRMs to include in the plan (by using the MEMBER option of BIND PLAN), DB2 automatically binds those DBRMs into packages and then binds those packages into the plan. The plan contains information about the designated packages and about the data that the application programs intend to use. The plan is stored in the DB2 catalog.

In addition to building packages and plans, the bind process does the following tasks:

- **Validates the SQL statements using the DB2 catalog.** During the bind process, DB2 checks your SQL statements for valid table, view, and column names. Because the bind process occurs as a separate step before program execution, errors are detected and can be corrected before the program is executed.
- **Verifies that the process binding the program is authorized to perform the data accessing operations requested by your program's SQL statements.** When you issue BIND, you can specify an authorization ID as the owner of the plan or package. The owner can be any one of the authorization IDs of the process that is performing the bind. The bind process determines whether the owner of the plan or package is authorized to access the data the program requests.
- **Selects the access paths that are needed to access the DB2 data your program needs to process.** In selecting an access path, DB2 considers indexes, table sizes, and other factors. DB2 considers all indexes that are available to access the data and decides which ones (if any) to use when selecting a path to the data.

BIND PLAN and BIND PACKAGE can be accomplished using DB2I panels, the DSNH CLIST, or the DSN subcommands BIND PLAN and BIND PACKAGE.

**Related reference:**

Chapter 16, “BIND PACKAGE (DSN),” on page 73

Chapter 17, “BIND PLAN (DSN),” on page 81

---

## Part 2. Description of 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.

**Related information:**

Part 3, “DB2 and related commands,” on page 47



---

## Chapter 3. Types of commands

DB2 supports several different types of commands.

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 12	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.

---

### The DSN command and its subcommands

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

**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.

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 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 QUERY (DSN)	The DSN subcommand FREE QUERY removes one or more queries from the access path repository. If any of the specified queries are in the dynamic statement cache, FREE QUERY purges them from the dynamic statement cache.
FREE PLAN (DSN)	The DSN subcommand FREE PLAN deletes application plans from DB2.
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 that affect the plan, but do not change the SQL statements in the programs.
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.

---

## DB2 commands

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

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

- A z/OS console or through an application program
- A DSN session
- A DB2I panel
- An IMS terminal
- A CICS terminal
- An application program, using the DB2 instrumentation facility interface (IFI)

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.

**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 information:**

Part 3, “DB2 and related commands,” on page 47

---

## 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.

/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.
/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.
/SSR (IMS)	The IMS /SSR command allows the IMS operator to enter an external subsystem command.
/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.
/STOP (IMS)	The IMS /STOP command (with the SUBSYS parameter) prevents application programs from accessing external subsystem resources.
/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.

**Related reference:**

 DSNAIMS stored procedure (DB2 Administration Guide)

 DSNAIMS2 stored procedure (DB2 Administration Guide)

---

## 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.

DSNC (CICS attachment facility)	The CICS attachment facility DSNC command allows you to enter DB2 commands from CICS.
DSNC DISCONNECT (CICS attachment facility)	The CICS attachment facility command DSNC DISCONNECT disconnects threads.

DSNC DISPLAY (CICS attachment facility)	The CICS attachment facility command DSNC DISPLAY displays information on CICS transactions accessing DB2 data, or statistical information associated with DB2ENTRYs and the DB2CONN.
DSNC MODIFY (CICS attachment facility)	The CICS attachment facility command DSNC MODIFY modifies the message queue destination of the DB2CONN, or modifies the maximum active thread value for the pool, for DSNC commands, or for DB2ENTRY.
DSNC STOP (CICS attachment facility)	The CICS attachment facility command DSNC STOP stops the attachment facility.
DSNC STRT (CICS attachment facility)	The DSNC STRT command starts the CICS attachment facility, which allows CICS application programs to access DB2 databases.

---

## 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.

Chapter 52, “MODIFY admtproc,APPL=SHUTDOWN,” on page 371	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.
Chapter 53, “MODIFY admtproc,APPL=TRACE,” on page 373	The MODIFY <i>admtproc</i> , APPL=TRACE command starts or stops traces in the administrative task scheduler.
Chapter 77, “START admtproc,” on page 461	The START <i>admtproc</i> command starts the scheduler that is specified in the <i>admtproc</i> parameter
Chapter 89, “STOP admtproc (z/OS ),” on page 517	The STOP <i>admtproc</i> command stops the administrative task scheduler that is specified in the <i>admtproc</i> parameter.

---

## 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 *irlmproc* .

---

## TSO CLISTS

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

Chapter 47,  
“DSNH (TSO  
CLIST),” on  
page 329

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 DB2I.

### Related tasks:

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



---

## Chapter 4. DB2 command parsing

All of the DB2 and related commands have a similar structure that is necessary to execute valid commands.

The following figure illustrates the pattern of DB2 commands.

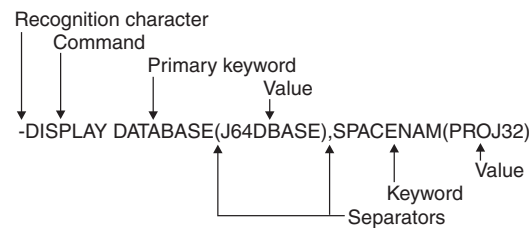


Figure 1. The format and parts of a DB2 command

The parts of a command are:

### 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

Several characters have special meaning for the syntax of DB2 commands.

The following characters have special meaning:

A blank is a separator.

Multiple blanks are equivalent to a single blank, except in strings enclosed between apostrophes.

, A comma is a separator.

' An apostrophe is 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, please refer to book "z/OS: MVS System Commands", 4.48.2 Starting a System Task from a Console.

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.

" A quotation mark is 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.

- = An 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.
- ( An open parenthesis marks the beginning of a parameter list.
- ) A close parenthesis marks the end of a parameter list.
- : A colon means 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.
- \* An asterisk has the following meanings:
  - \* 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 an asterisk. The asterisk can be used this way only in commands in which the pattern-matching operation is specifically permitted.

- 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.
- NO (two-character string) 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.

---

## Examples of keyword entry

The following examples illustrate valid combinations of keywords and parameters:

- MODE (FORCE)
- MODE=FORCE
- MODE (NOFORCE) (keyword negation)
- MODE=NOFORCE (keyword negation)
- DATABASE(name1 name2 . . . name*n*) ACCESS(RO)
- SPACENAM (name1,name2) ACCESS(RO)
- ACCESS (RO),SPACENAM=name
- Combinations of the preceding

Do not use more than one parameter after an equal sign or an error condition occurs.



---

## Chapter 5. DSN subcommand parsing

The parsing of DSN subcommands conforms to standard TSO command parsing conventions.

To continue a subcommand on the next line while using 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 will reduce the overall size of the command.

**Important:** The names of the DSN command and its subcommands cannot be abbreviated. For compatibility with prior releases of DB2, abbreviations for some keywords are allowed. To avoid potential problems, avoid abbreviating keywords.



---

## Chapter 6. Scope of DB2 and related commands

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 which options you specify with them:

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. The description of each command includes its scope.



---

## Chapter 7. 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 you can receive:

- 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 parameter.
- For DISPLAY THREAD, the number of lines of output.  
DISPLAY THREAD does not display more than 254 lines of output.



---

## Chapter 8. Issuing commands to DB2 from IFI

Consider using IFI to let your programs issue commands to DB2. This method returns information about the success or failure of the command to your program.

If the command issues a non-zero return code, the information returned to your program includes diagnostic information about the command processed. For commands that are executed asynchronously, the return code indicates whether the command started successfully.





---

## Chapter 9. Command line processor

You can use the command line processor to issue SQL statements, bind DBRMs that are stored in HFS files, and call stored procedures.

The command line processor operates as follows:

- The user types a command at the UNIX System Services system prompt to start the command line processor.

The command line processor can be started in either batch mode or interactive input mode.

- The command line processor automatically redirects output to the standard output device.

Piping and redirection are supported.

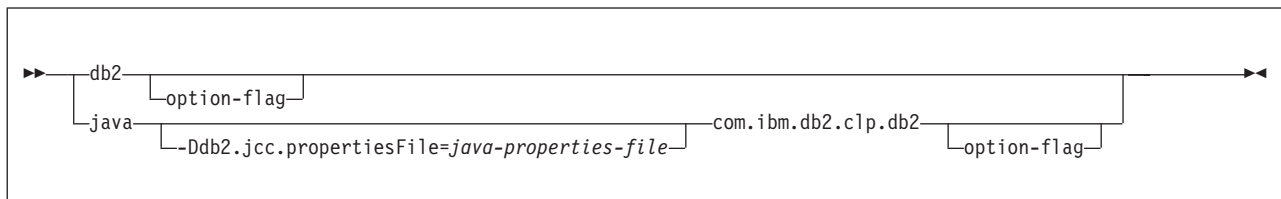
- The command line processor notifies the user of successful or unsuccessful completion.
- Following execution of a command line processor command, control returns to the operating system command prompt, and the user can enter more commands.

### How to start and set up the command line processor

The command line processor can be started in the following two modes:

- Batch mode
- Interactive input mode

The syntax shows how to start the command:



If you start the command line processor with the db2 command, db2 must have been previously defined as an alias for the command that invokes the command line processor:

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

Specify -Ddb2.jcc.propertiesFile if you want to run the command line processor with JDBC properties other than the default properties.

For example, if you want XML data that the command line processor processes to be in textual format, rather than binary XML format, specify a Java properties file that contains this line:

```
xmlFormat=0
```

### Information about batch and interactive modes

#### Batch mode

To run the processor in batch mode issue the word `db2`, followed by the option `-f filename`. In this mode, the command line processor takes all of the data from the named *file*, and processes all of the commands in the file consecutively. For example, the following command will direct the command line processor to process all the statements in the file named `demo.clp`:

```
db2 -f demo.clp
```

### Interactive Mode

Issue the word `db2`, and the command line processor automatically goes into interactive mode. The command line processor prompts you to enter SQL statements or commands, one at a time. The simplest form of the DB2 command is as follows:

```
db2
```

While the command line processor runs in interactive mode, it prompts you for input with the following output:

```
db2 =>
```

### Command line processor options and settings

You can specify command options for the command line processor for either the entire command line processor session or with option flags. You can view the options settings by issuing the command line processor to issue the `LIST COMMAND OPTIONS` command. You can change option settings from the interactive input mode or a command file by using command line processor to issue the `UPDATE COMMAND OPTIONS` command. The command line processor sets options in the following order:

1. Sets up default options.
2. Reads option settings from the properties file to override the defaults.
3. Reads the command line options to override option settings from properties file.
4. Accepts input from the `UPDATE COMMAND OPTIONS` command as a final override.

The following table outlines the default options for the command line processor.

*Table 1. Default options for the command line processor*

Option description	Default setting	Keyword in properties file	Options flag
Display SQLCA data.	OFF	DisplaySQLCA	a
Automatically commit SQL statements.	ON	AutoCommit	c
Read command input from the named file.	OFF	InputFilename	f<filename>
Display data and messages in standard output.	ON	DisplayOutput	o

Table 1. Default options for the command line processor (continued)

Option description	Default setting	Keyword in properties file	Options flag
Stop the command line processor and exit to the operating system if error is encountered during execution of a statement.	OFF	StopOnError	s
Use semicolon( ;) as the statement termination character. If this option is not set, each line is considered a statement unless it ends with a space followed by a backslash( \)..	OFF	TerminationChar	t
Use x as the statement termination character.	OFF	TerminationChar	tdx
Echo command text to standard output.	OFF	Echo	v
Return data without any headers, including column name.	OFF	StripHeaders	x
Output data and messages are stored in the given file.	OFF	OutputFilename	z<filename>
Transaction isolation level.	CS	IsolationLevel	
Maximum number of lines to return from a SELECT query.	ALL	MaxLinesFromSelect	
Controls the maximum number of characters of a column that is to be displayed on either <ul style="list-style-type: none"> <li>• Execution of SQL SELECT statements</li> <li>• Result set that is returned from CALL stored procedures</li> <li>• Command line processor catalog commands.</li> </ul>	32	MaxColumnWidth	Width=<value> pair

## Setting up the command line properties file

You can create this optional file for the command line processor in order to customize your environment. When the processor is invoked from the UNIX system services prompt, in either interactive or batch mode, the processor checks for the definition of CLPPROPERTIESFILE environment variable to locate the properties file. You can define this environment variable if you intend to use the command line processor properties file to change the command line processor default values.

The command line properties file is a simple text file that contains attributes of the form key=value pair. The properties file is initially loaded when the command line processor is invoked. Any changes to the properties file while the command line processor session is active do not take effect. In order for the changes to take effect, you need to terminate the current session and start a new command line processor session. You can always change command line processor default settings using either option flags or the UPDATE COMMAND OPTIONS command, regardless of whether you use the command line processor properties file. The following keywords can be used to set the command line processor command options in the properties file.

*Table 2. properties file : table for command options*

Keyword name	Acceptable keyword value
DisplaySQLCA	ON   OFF   blank
AutoCommit	ON   OFF   blank
InputFilename	<Filename>   blank
OutputFilename	<Filename>   blank
DisplayOutput	ON   OFF   blank
StopOnError	ON   OFF   blank
TerminationChar	ON   OFF   any single character or blank
Echo	ON   OFF   blank
StripHeaders	ON   OFF   blank

**Note:**

If blanks are specified for the keyword in the properties file, the default value is used.

If InputFilename is specified in the properties file, the command line processor executes in the batch mode.

To enter into interactive mode, you can do one of the following actions:

- Remove InputFilename keyword or blank out the value
- Invoke command line processor with +f command line options flag. For example:  
db2 +f

You can use the keywords in the following table to set the command line processor non-command options in the properties file:

*Table 3. properties file : table for non-command options*

Keyword name	Acceptable keyword value
IsolationLevel	RR   RS   CS   UR   blank

Table 3. properties file : table for non-command options (continued)

Keyword name	Acceptable keyword value
MaxLinesFromSelect	ALL   Positive numeric value   blank

### UPDATE COMMAND OPTIONS command

You can use the UPDATE COMMAND OPTIONS command by using the following format:

```
UPDATE COMMAND OPTIONS USING [OPTIONS FLAG|KEYWORD] [ON|OFF]
```

Table 1 on page 28 shows the defaults for option flags and keywords. You can change these default settings by using the UPDATE COMMAND OPTIONS command.

If you want to turn off AutoCommit, specify one of the following commands:

- UPDATE COMMAND OPTIONS USING c OFF
- UPDATE COMMAND OPTIONS USING AutoCommit OFF

### LIST COMMANDS OPTIONS command

You can tell which options are currently in effect by issuing the LIST COMMAND OPTIONS command, which is as follows:

```
LIST COMMAND OPTIONS
```

### Examples of command line processor invocations

Assume that the clp.properties file contains the following attributes:

```
AutoCommit=OFF  
Echo=ON
```

Assume also that the command line processor is invoked from Unix System Services prompt as:

```
db2 -t
```

The options are processed as follows:

- The command line processor starts with default options.
- The command line processor reads the properties file, and turns off Autocommit and turn on Echo options.
- The command line processor reads the command line options flags (-t) and sets the termination character to semicolon (;).

However, the command line processors enter into interactive mode if both of the conditions following are true:

- No InputFilename is specified in the properties file.
- No '-f' options flag is specified during the command line processor command invocation.

Assume that the clp.properties file contains the following attributes:

```
AutoCommit=ON
Echo=
InputFilename=/tmp/test.clp
TerminationChar=%
OutputFilename=
```

Assume also that the command line processor is invoked from the UNIX System Services prompt as:

```
db2 +f
```

The options are processed by the command line processor, which takes the following actions:

- Starts with default options.
- Reads the properties file and will turn on Autocommit, sets termination character to (%) and reads input from the file named '/tmp/test.clp'. Because Echo and OutputFilename keywords in the properties file are blank, the default value is assigned.
- Reads the command line options flags (+f) that turns off the file mode, and enters into interactive mode.

## Connecting to a DB2 server

Use the CONNECT command to connect to a DB2 server. Before you can access any data you must first connect to a server. Issue the following command:

```
CONNECT TO <URL> USER <username> USING <password>
```

where URL is in the form :

**server:port database property = value;**

The URL specifies a data source name that consists of server, port, database and property=value fields.

The part of the URL has the following meanings:

### **server**

The domain name or IP address of the database server.

### **port**

The TCP/IP server port number that is assigned to the database server. Inclusion is optional.

### **database**

The DB2 location name that is defined during installation.

### **property=value**

Specifies the property for the JDBC connection. Inclusion is optional

The command line processor uses IBM Data Server Driver for JDBC and SQLJ type 4 connectivity to connect to data sources. Ensure that the driver packages are bound at the target data sources and that you have execute permissions on those packages. All characters in the DB2 location name must be uppercase characters. The IBM Data Server Driver for JDBC and SQLJ does not convert lowercase characters in the database value to uppercase for IBM Data Server Driver for JDBC and SQLJ type 4 connectivity.

You can determine the location name by executing the following SQL statement on the server:

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

You can also define your own alias names in the properties file and use the alias names to connect to specified DB2 servers. The alias names that are specified in the properties file should not be confused with alias that you can create for a DB2 table or view through CREATE ALIAS SQL statement. The alias name specified in the properties file is defined as follows :

```
DB2ALIASNAM=URL,username,password
```

You can then use the following command to connect to the DB2 server.

```
CONNECT TO DB2ALIASNAME
```

For example, in the properties file, DSN1 is the alias name, which is defined as follows:

```
DSN1=v01ec107.svl.ibm:8000/STLEC1,JOHND0E,PASSW0RD
```

You can issue the following command using the command line processor to connect to DSN1 alias.

```
CONNECT TO DSN1
```

## Command line processor basic commands

### **BIND** *filename* [-collection collection] [options]

This command binds the HFS DBRMs into packages or collections at the target DB2 server.

### **COMMIT**

This command makes all changes that have been requested since the previous commit or rollback and releases any database locks that are currently held by the active connection. If the auto-commit is on, the command line processor automatically commits at the end of each SQL statement at the current server. 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. If auto-commit is off and a connection is made to a new server without issuing COMMIT or ROLLBACK at the current server where a transaction was in progress, the command line processor issues error message DSN109I.

### **DISCONNECT**

This command closes and releases the connection from the currently connected database. Use this command should only be used when there are not any transactions in progress or the command line processor will produce error message DSN113I.

If this command is issued when there are no active database connections, the command line processor issues error message DSN108I.

### **CHANGE ISOLATION TO** *RR* | *RS* | *CS* | *UR*

This command changes the level of isolation between the user transactions and the transactions of other concurrent users. Choosing a high isolation level protects against interference from other users, and also protects

system performance and limits concurrent access to data. The supported isolation levels, from highest to lowest, are as follows:

**Repeatable read (RR)**

Repeated reads of the same data are guaranteed consistent within a transaction.

**Read stability (RS)**

When performing a repeated read of a table, old rows remain unchanged but new rows might appear.

**Cursor stability (CS)**

Guarantees that a row of data will not change while your cursor is positioned on it. This is the default isolation level.

**Uncommitted read (UR)**

Allows you to read updates that have not been committed by other users and that are therefore subject to being rolled back.

By default, the command line processor uses the isolation level of cursor stability for executing transactions. To override this default, you must define `IsolationLevel` keyword in the properties file. To change the isolation level later during a command line processor session, you can use the `CHANGE ISOLATION LEVEL` command, which you can execute after making a connection to a DB2 server. If connection is active, the command line processor issues message `DSNC108I`.

**ECHO** *string*

This command repeats, or echoes its content to the output stream. This can be useful for generating labels in an output file. For example, if the command line processor is running test cases from an input file and saving the results in an output file, you can use the `ECHO` command to label the test cases. For example, you can issue the following command:

```
ECHO Test Case 10
```

**ROLLBACK**

This command undoes all the changes that were made in the current transaction and releases any database locks that are currently held by the active connection. If auto-commit is on, this command has no effect.

**TERMINATE**

This command terminates the database connection, if any, and ends your command line processor session. If a transaction is in progress, it will be committed.

**!** *UNIX System Services command*

Within the command line processor, you can issue UNIX System Services command by prefixing the command with the `!` character. The following command lists all the files and directories in the current directory.

```
! ls
```

## Command line processor information commands

**CHANGE MAXCOLUMNWIDTH TO** *number*

This command controls the maximum number of characters of a column that would be displayed on execution of SQL `SELECT` statements.

**CHANGE MAXLINESFROMSELECT TO** *ALL|number*

This command controls the number of rows that are to be displayed from



the result set that is returned by SELECT SQL statements, CALL stored procedure statements, or command line processor catalog commands. The DSN1111I error message is returned if you don't specify ALL or a positive number.

## DESCRIBE CALL

This command specifies the name of the stored procedure that you want to be described. If the stored procedures are not fully qualified, the schema that is used is that of the current user. Fully qualified stored procedures must be in the form *schema.stored procedure*.

The DESCRIBE CALL command lists the following information about each parameter of the stored procedure and returns a table with the following columns:

- COLUMN\_NAME: The column or parameter name.
- COLUMN\_TYPE: A type of column or parameter. The following are possible return values are possible:
  - 0: unknown
  - 1: IN parameter
  - 2: INOUT parameter
  - 3: result column in ResultSet
  - 4: OUT parameter
  - 5: procedure return value
- TYPE\_NAME: The name of the data source dependent type.
- LENGTH: The length in bytes of data.
- PRECISION: The precision.
- SCALE: The scale of decimal data or the number of fractional second digits of timestamp data.
- NULLABLE: Has the following possible returned values:
  - 0: Does not allow NULL values.
  - 1: Allows NULL values.
  - 2: Unknown if value is nullable.

**Example:** Assuming that TEST.EMPLOYEE\_INFO is a stored procedure that is already defined, the user can issue the following command to get information about this stored procedure:

```
DESCRIBE CALL TEST.EMPLOYEE_INFO
```

## DESCRIBE TABLE [schema] tablename | view

This command specifies the table or view that you want described. An alias for the table cannot be used in place of the actual table. If the table name or view is not fully qualified, the schema that is used is that of the current user. Fully qualified table names must be in the form *schema.table*.

The DESCRIBE TABLE command lists the following information about each column:

- COLUMN\_NAME: The column of the table.
- TABLE\_SCHEM: A table schema.
- TYPE\_NAME: The name of the data source dependent type.
- COLUMN\_SIZE: The column size. For character or date types, this is the maximum number of characters. For number or decimal type, this is precision.

- DECIMAL\_DIGITS: The number of fractional digits or number of fractional-second digits for the timestamp.
- IS\_NULLABLE: Has the following possible returned values:
  - No: Column does not allow NULL values.
  - Yes: Column might allow NULL values.
  - Empty string: Unknown if column allows NULL values.

#### DISPLAY RESULT SETTINGS

This command displays the current settings for MAXCOLUMNWIDTH and MAXLINESFROMSELECT.

#### LIST TABLES FOR *USER* | *ALL* | *SYSTEM* | **SCHEMA***schema-name* |

This command lists tables, views, or aliases that are associated with the connected database. If the FOR clause is not specified, then tables for USER are listed.

**Example 1:** To list all tables or views in the connected database, use the following command.

```
FOR ALL
```

**Example 2:** To list all catalog tables in the connected database, use the following command.

```
FOR SYSTEM
```

**Example 3:** To list all tables or views in the database for the specified *<schema-name>* only, use the following command.

```
FOR SCHEMA <schema-name>
```

**Example 4:** To list all tables or views in the connected database for the current user, use the following command.

```
FOR USER
```

The LIST TABLE command lists the following information about each table or view:

- The table, view, or alias.
- schema.
- Type- typical types include:
  - TABLE
  - VIEW
  - SYSTEM TABLE
  - GLOBAL TEMPORARY
  - LOCAL TEMPORARY
  - ALIAS
  - SYNONYM
  - AUXILIARY
- Creation time

### Command line processor package-related commands

#### **BIND** *filename* [-collection *collection*] [options]

This command binds the HFS DBRMs into packages or collections at the target DB2 server.

## 11

## 11

11



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 URI's that use the format: 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.

**Example 1:** 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 the DBA is certain that no other documents are referenced by the schema. To complete it, the DBA can issue the following command:

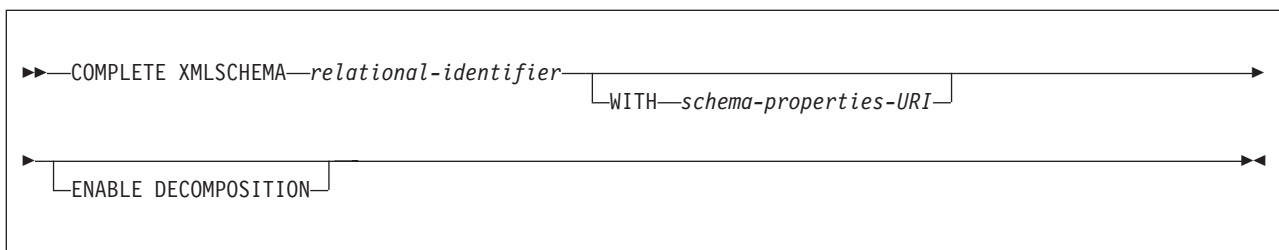
```
-ADD XMLSCHEMA DOCUMENT TO JOHNDOE.PRODSHEMA1
ADD http://mycompany.com/xml/schema/order.xsd
FROM file:///u/temp/order.xsd
ADD http://othercompany.com/external/po.xsd
FROM file:///u/temp/po.xsd
COMPLETE
```

**Example 2:** An XML schema that is referred to by the relational name JOHNDOE.TEST is registered, and the DBA needs 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. The command to issue is:

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

**-COMPLETE XMLSCHEMA**

Completes the registration of an XML schema.



**Option descriptions:**

*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 URI's that use the format: file://.

**ENABLE DECOMPOSITION**

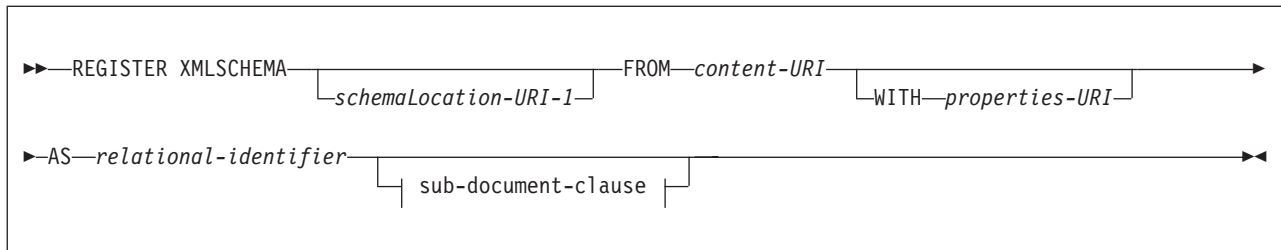
Indicates that this schema is to be used for the purpose of decomposing XML documents.

**Example:** Assumes that a complicated schema, which consists of multiple documents is already registered. Assume also that it has the relational name PRODSHEMA. The schema has additional properties that you want to store in the database catalogs, which is stored locally in /u/temp/CUSTPROP.XML. The command is as follows:

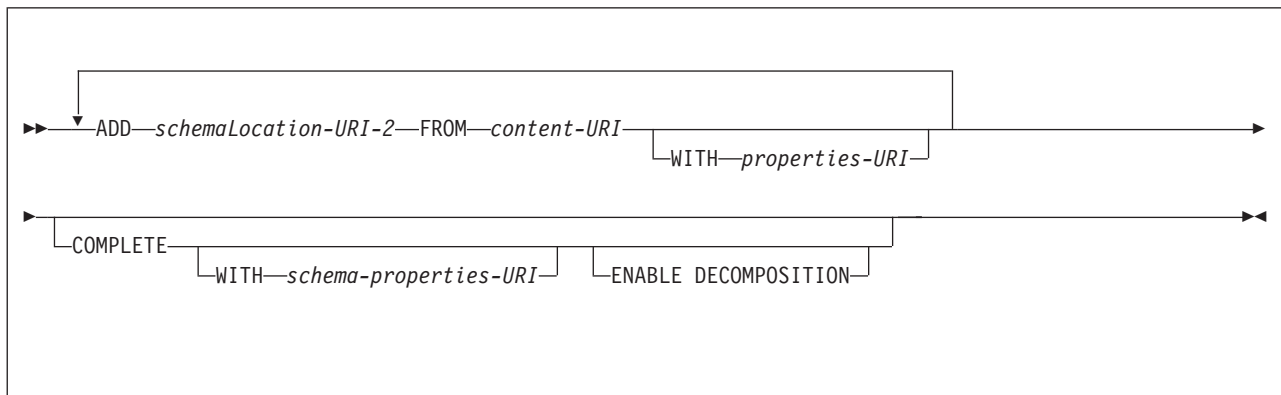
-COMPLETE XMLSCHEMA PRODSHEMA WITH 'file:///u/temp/CUSTPROP.xml'

**-REGISTER XMLSCHEMA**

Registers an XML schema with the database manager.



**sub-document-clause:**



**Option descriptions**

*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 URIs that use the format: 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 URIs that use the format: 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:// based URI. 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. It can only be specified when the XML schema is also being completed.

**Example 1:** Assume that you want to register a simple schema named EXAMPLE.TEST1 and to make it available immediately. The schema is http://mycompany.com/prod/p1\_1.xsd, and is stored locally in /u/temp. The command is as follows:

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

**Example 2:** Assume that you want to register a more complicated schema named PRODSHEMA of multiple documents SQL or XML commands. The schema is `http:// mycompany.com/prod/cust.xsd`, and is stored locally in `/u/temp`. Three of the subdocuments, stored locally in the same location and named `cust1.xsd` through `cust3.xsd`, registered along with any additional properties that you want to store. The schema properties are stored locally in `/u/temp/CUSTPROP.XML`. Do not finalize the schema at this time, because you need to add more related schema documents that you will need to add later on. Because the XML schema is not yet completed, you cannot store the schema properties document in the catalogs yet. When the schema is completed by using the COMPLETE keyword in either the COMPLETE XMLSCHEMA or ADD XMLSCHEMA DOCUMENT commands, you can then store the schema properties document. The command is as follows:

```
-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
```

**Example 3:** This example specifies that the XML document in `/u/gb/doc1.xml` is to be decomposed according to the XSR-registered schema `genbankschema`.

```
-DECOMPOSE XML DOCUMENT /u/gb/doc1.xml
XMLSCHEMA genbankschema
```

#### **-REMOVE XMLSCHEMA**

Removes the previously registered XML schema.

►►—REMOVE XMLSCHEMA—*relational-identifier*—◄◄

#### **Option descriptions:**

##### *relational-identifier*

Specifies the relational name given to an XML schema in a previous REGISTER XMLSCHEMA command.

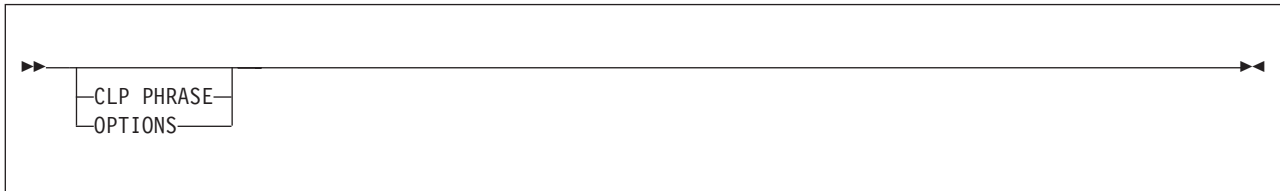
The schema of the relational identifier must be SYSXSR. If you specify a qualified name, the qualifier must be SYSXSR; for example, SYSXSR.MYSCHEMA. If you specify an unqualified name, you need to execute the SQL statement SET SCHEMA="SYSXSR" before you execute the command. A name that is enclosed in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

**Example 1:** This example specifies the removal of the previously registered schema PRODSHEMA.

```
-REMOVE XMLSCHEMA PRODSHEMA
```

## Command line processor help commands

You can use the command line processor help commands in order to invoke information or options from the command line processor. The following syntax diagram shows these commands:



**Example 1:** Displays all of the available command line processor commands.

```
?
```

**Example 2:** Displays information about the command line processor command that starts with the keyword LIST.

```
? LIST
```

**Example 3:** Displays information about the command line processor command that starts with the keywords, "Update Command."

```
? Update Command
```

**Example 4:** Displays information about all of the command line processor option flags.

```
? OPTIONS
```

## SQL statements for the command line processor

SQL statements allow you to work with stored data in your database. The statements are applied against the database you are connected to. The command line processor supports the following SQL statements:


- ALTER
- CALL
- CREATE
- DELETE
- DROP
- GRANT
- INSERT
- REVOKE
- SELECT
- UPDATE




**Related concepts:**

 Customization of IBM Data Server Driver for JDBC and SQLJ configuration properties (Application Programming for Java)

**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)



---

## **Chapter 10. 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 3. DB2 and related commands

The DB2 command topics contain syntax diagrams, semantic descriptions, rules, and usage examples of commands, organized alphabetically by command name.

**Related information:**

Part 2, “Description of DB2 and related commands,” on page 7



---

## Chapter 11. -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.

This command is also available in conversion mode.

**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

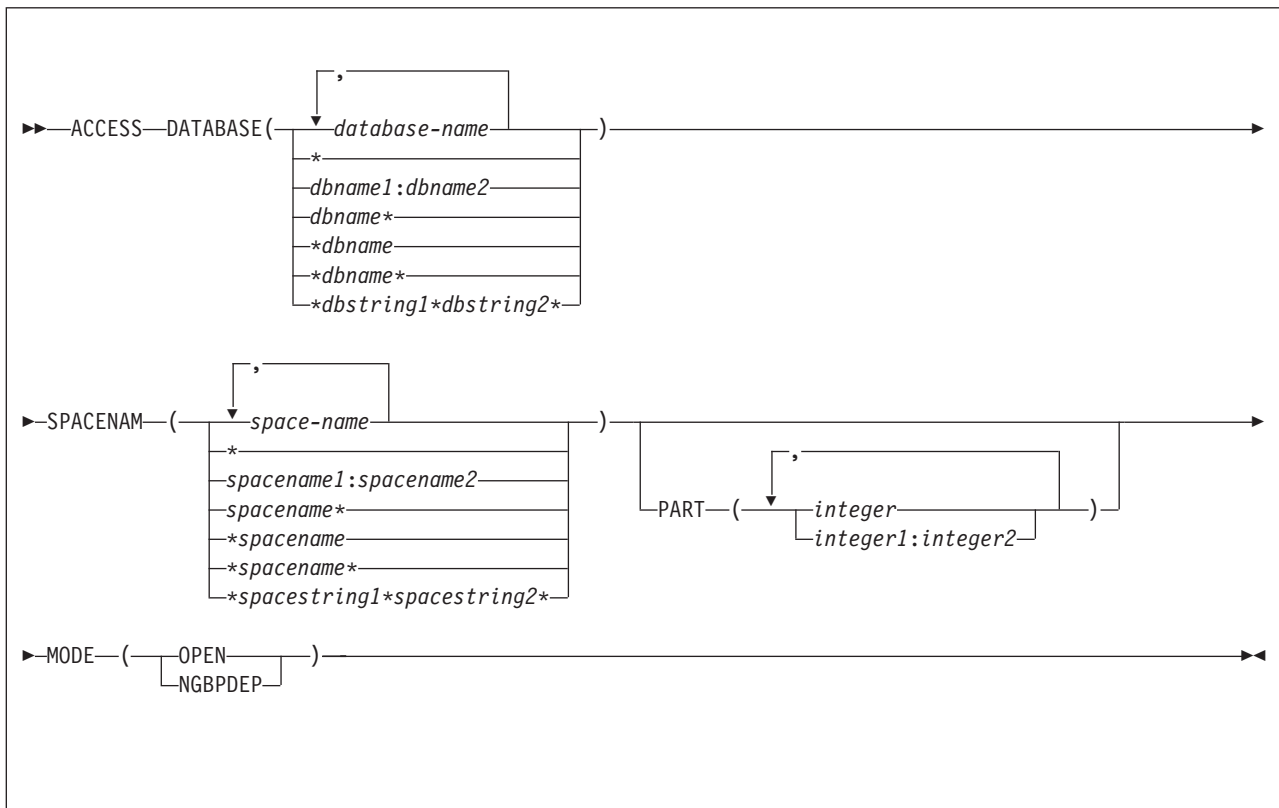
When you are using a privilege set that does not contain the STARTDB privilege for a specified database, DB2 issues an error message and the ACCESS command fails.

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 START or STOP state of other partitions does not change.

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.

##### **NGBPDEP**

Converts the specified page set or partition, non-disruptively to a non-group buffer pool dependent. You should use this before running large batch processes against a particular page set or partition to improve performance in a data sharing environment. Only issue this command to the member on which you plan to run the batch programs.

## **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
```

**Related concepts:**

-  Physical open of a page set of partition (DB2 Data Sharing Planning and Administration)
-  Removal of group buffer pool dependency (DB2 Data Sharing Planning and Administration)
-  Inter-DB2 interest and GBP-dependency (DB2 Data Sharing Planning and Administration)

---

## Chapter 12. -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

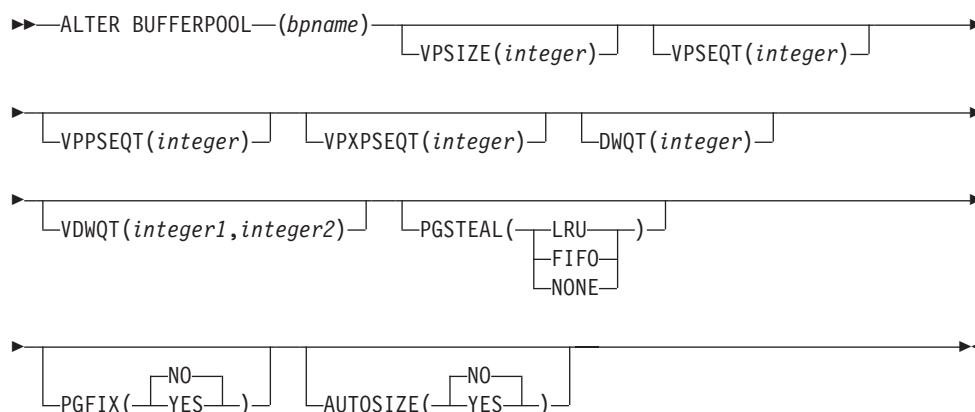
### Authorization

To 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

( *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.

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

*integer* can range from 0 to 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 approximately 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.

*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 from 0 to 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 being 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 being released by the accessing agent. 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.

*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 from 0 to 100. The initial default value is 50.

When VPPSEQT=0, parallel processing operations 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.

*integer* specifies the assisting parallel sequential threshold for the buffer pool. *integer* is expressed as a percentage of the parallel sequential threshold (VPPSEQT). Whenever the sequential steal threshold or the parallel sequential

threshold is altered, it directly affects the portion of buffer resources dedicated to assistant parallel operations. The valid values range from 0 to 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.

*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 from 0 to 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 ( integer1 , integer2 )**

Changes the buffer pool's vertical deferred write threshold.

*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 from 0 to 90.

This threshold determines when deferred writes begin, based on the number of updated pages for a given 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 and 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.

*integer2* specifies the vertical deferred write threshold for the buffer pool. *integer2* 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 *integer1* cannot provide a fine enough granularity between *integer1* values of 0 and 1. *integer2* only applies when *integer1* is 0; DB2 ignores a value specified for *integer2* if the value specified for *integer1* is non-zero. *integer2* can range from 0 to 9999. The default value is 0.

If *integer1* is 0 and *integer2* is a non-zero value, DB2 uses the value specified for *integer2* to determine the threshold. If both values are 0, the *integer1* 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 should be managed by using the least recently used (LRU) algorithm.

##### **(FIFO)**

Specifies that the buffer pool buffers should be managed by using the first-in-first-out (FIFO) algorithm.

##### **(NONE)**

Specifies that no page stealing is to take place. Data that is brought in stays resident in the buffer pool.

|  
|  
|

**PGFIX**

Specifies whether the buffer pool should be fixed in real storage when it is used.

**(NO)**

Specifies that the buffer pool is not fixed in real storage. Page buffers are fixed and unfixed in real storage across each I/O and group buffer pool operation.

This is the default.

**(YES)**

Specifies that the buffer pool is fixed in real storage. Page buffers are fixed when they are first used after the buffer pool is allocated or expanded.

**AUTOSIZE**

Specifies whether the buffer pool adjustment is turned on or off.

**(NO)**

Specifies that the buffer pool does not use Workload Manager (WLM) services for automatic buffer pool sizing adjustment.

This is the default.

**(YES)**

Specifies that the buffer pool uses WLM services, if available, to automatically adjust the size of the buffer pool based on dynamic monitoring of the workload goals and the available storage on the system.

**Usage notes**

The following description contains additional information about how to use the ALTER BUFFERPOOL command.

***Changing several buffer pool attributes:*** A failure in modifying one buffer pool attribute has no effect on other modifications requested in the same command.

***Contracting an active buffer pool:*** If you use ALTER BUFFERPOOL to contract the size of an active buffer pool, DB2 contracts the pool by marking active buffers as "to be deleted," which means that they are not reusable to satisfy other page requests. However, the virtual storage might not be freed immediately. A system administrator can determine the status of the buffer pool by issuing the DISPLAY BUFFERPOOL command.

***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 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 will become 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 4. 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)
```

Be very careful when using this option 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)



---

## Chapter 13. -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

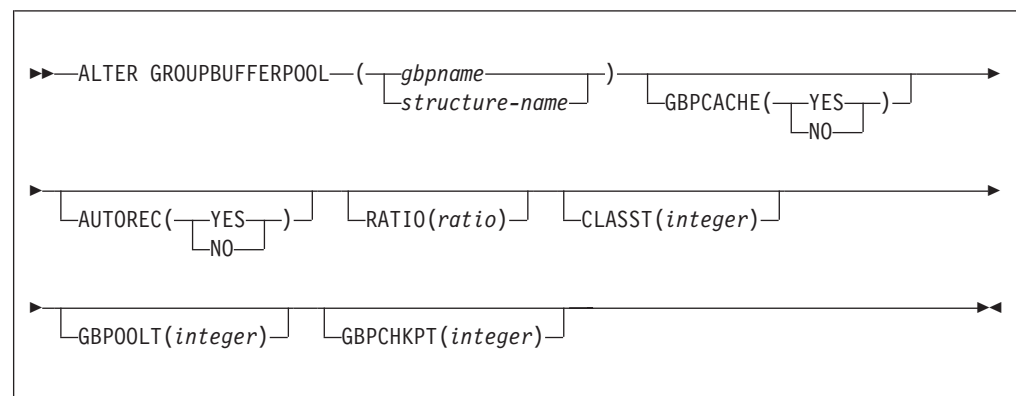
### Authorization

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

- SYSOPR authority
- SYSTRL 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 5. Precedence of a no-data-caching specification

Group buffer pool specification	Page set specification	Attribute that takes precedence
GBPCACHE(NO)	GBPCACHE CHANGED	GBPCACHE(NO)
	GBPCACHE ALL	
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 6. 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 7. 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 <sup>2</sup>	No <sup>3</sup>

Table 7. Changing group buffer pool attributes (continued)

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

**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 14. -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

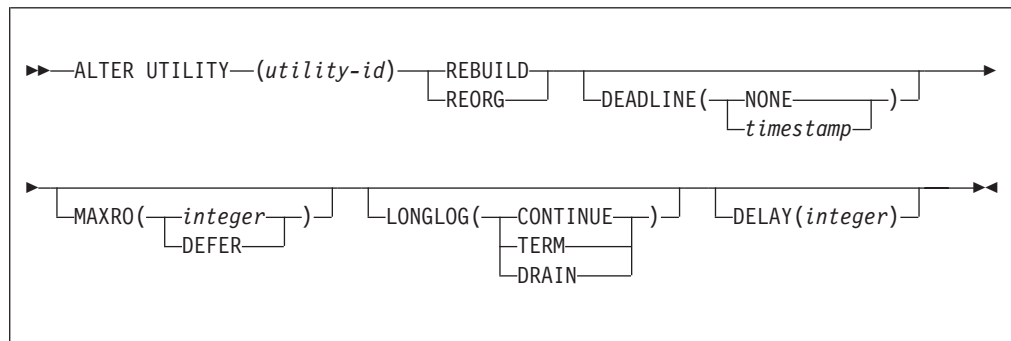
To execute this command, you must use the primary or some secondary authorization ID of the process that originally submitted the utility job. Alternatively, you must use a privilege set of the process that includes one of the following authorities:

- 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 15. -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.

Diagram illustrating the syntax for the ARCHIVE LOG command:

```

ARCHIVE LOG
  MODE (QUIESCE)
    TIME (nnn)
    WAIT (
      NO
      YES
    )
  SCOPE (
    MEMBER
    GROUP
  )
  CANCEL OFFLOAD
  
```

**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.

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.

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 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.

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 initiate 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 16. 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 DSNTPP. 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 field BIND NEW PACKAGE.

Table 8. Summary of privileges needed for BIND PACKAGE options

Bind option	Installation panel field BIND NEW PACKAGE	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"> <li>• The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections</li> <li>• SYSADM, SYSCTRL, or system DBADM authority</li> </ul>
	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"> <li>• The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections</li> <li>• SYSADM, SYSCTRL, or system DBADM authority</li> <li>• PACKADM authority on the collection or on all collections</li> <li>• The BIND package privilege (can only add a new version of an existing package)</li> </ul>



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

Bind option	Installation panel field BIND NEW PACKAGE	Authorization required to run BIND PACKAGE
ADD, specifying an OWNER other than the primary authorization ID <sup>(1, 2)</sup>	BINDADD	<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 to add a new package or new version of an existing package to a collection:</p> <ul style="list-style-type: none"> <li>• The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections</li> <li>• SYSADM, SYSCTRL, or system DBADM authority</li> </ul> <p>The binder can also be a role.</p>
	BIND	<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 to add a new package or new version of an existing package to a collection:</p> <ul style="list-style-type: none"> <li>• The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections</li> <li>• SYSADM, SYSCTRL, or system DBADM authority</li> <li>• PACKADM authority on the collection or on all collections</li> <li>• The BIND package privilege (can only add a new version of an existing package)</li> </ul> <p>The binder can also be a role.</p>
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"> <li>• Ownership of the package</li> <li>• 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."</li> <li>• PACKADM authority on the collection or on all collections</li> <li>• SYSADM, SYSCTRL, or system DBADM authority</li> </ul>

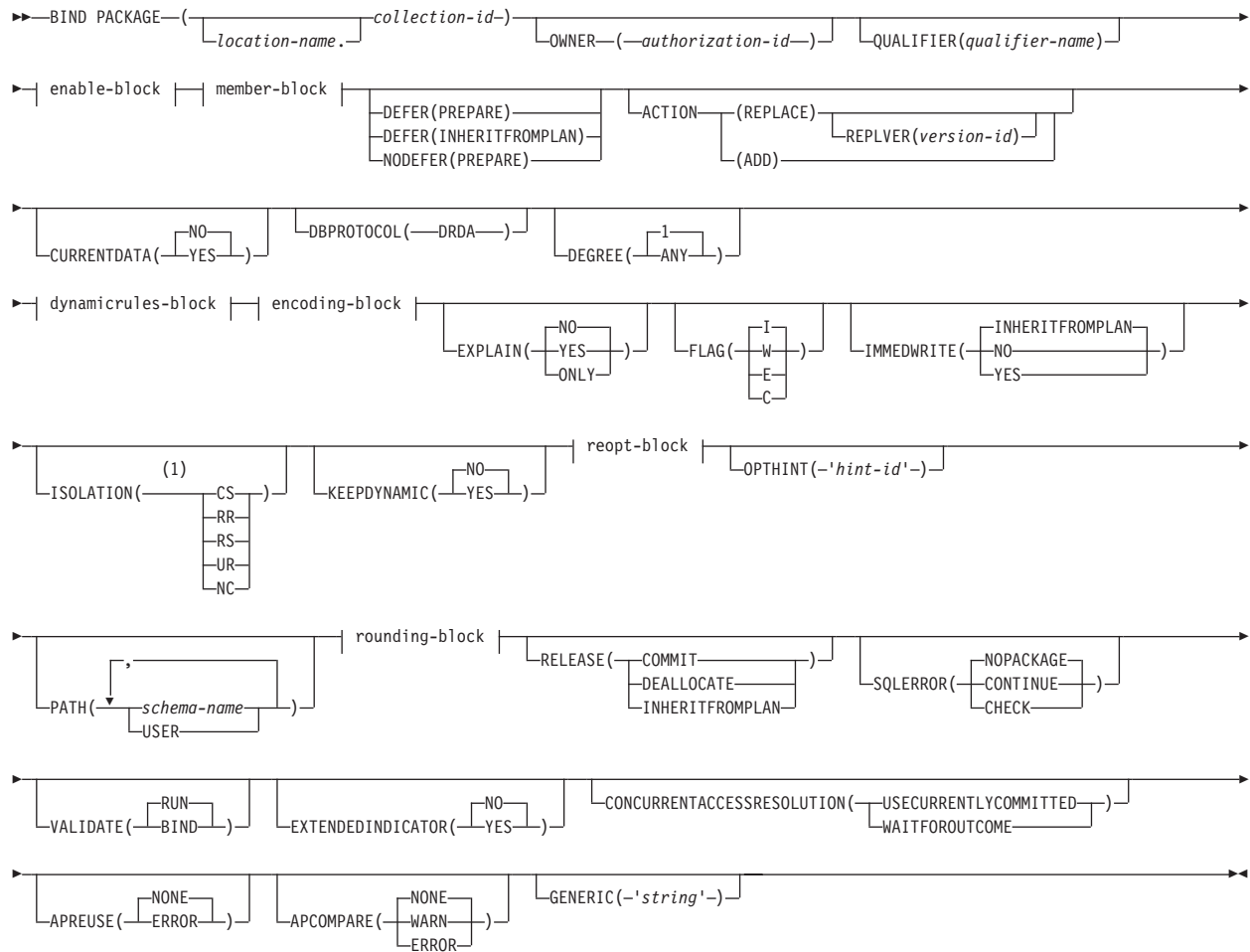
Table 8. Summary of privileges needed for BIND PACKAGE options (continued)

Bind option	Installation panel field BIND NEW PACKAGE	Authorization required to run BIND PACKAGE
REPLACE, specifying an OWNER other than the primary authorization ID <sup>(1, 2)</sup>	BINDADD or BIND	<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"> <li>• BIND privilege on the package. If the package does not exist, see the authorization that is required for "ADD, specifying an OWNER other than the primary authorization ID."</li> <li>• PACKADM authority on the collection or on all collections</li> <li>• SYSADM, SYSCTRL, or system DBADM authority</li> </ul> <p>The binder can also be a role.</p>
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"> <li>• Ownership of the package</li> <li>• COPY privilege on the package</li> <li>• BINDAGENT privilege from the owner of the package</li> <li>• PACKADM authority on the collection or on all collections</li> <li>• SYSADM, SYSCTRL, or system DBADM authority</li> </ul>

**Note:**

1. If any of the authorization IDs of the process has the SYSADM authority or SYSCTRL authority or system DBADM authority, OWNER *authorization-id* 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 *authorization-id* can specify the grantor as OWNER. Otherwise, the OWNER *authorization-id* must be one of the primary or secondary authorization IDs of the binder.
2. 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.

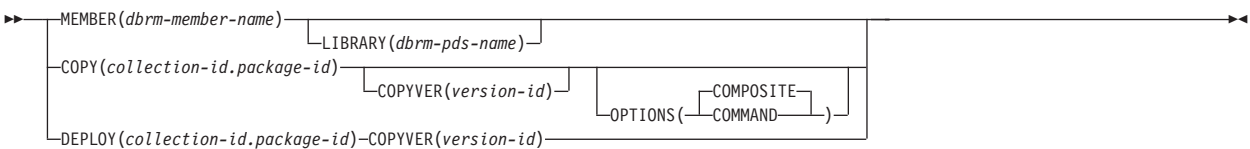
## Syntax



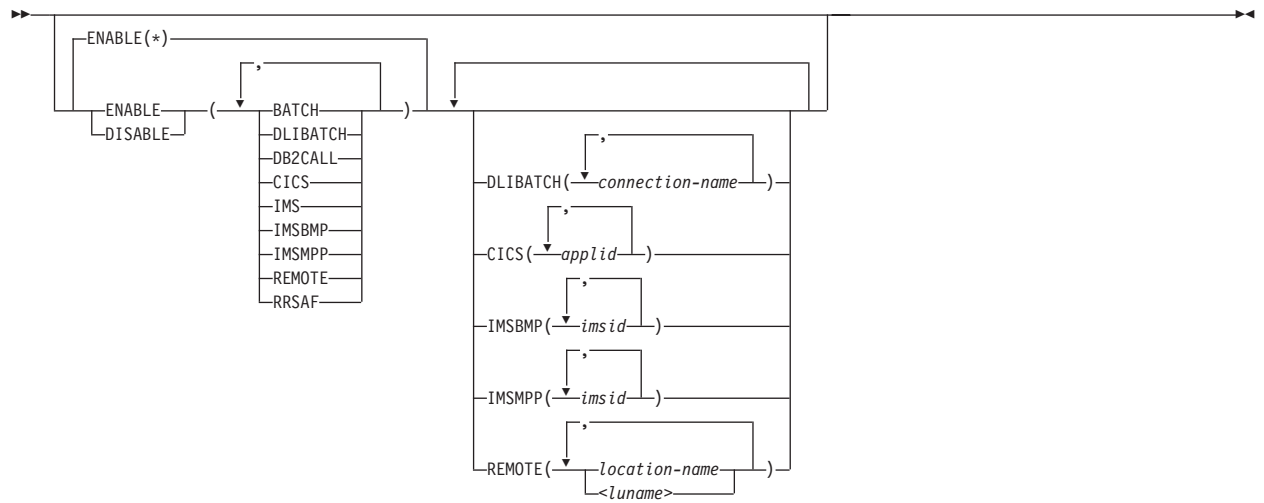
### Notes:

- 1 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 19, "BIND and REBIND options," on page 91.

## 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 SPUI 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 SPUI 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:**

Chapter 2, “The bind process,” on page 5

**Related reference:**

Chapter 19, “BIND and REBIND options,” on page 91

Chapter 61, “REBIND PACKAGE (DSN),” on page 405

Chapter 17, “BIND PLAN (DSN),” on page 81

---

## Chapter 17. 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 9. 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: <ul style="list-style-type: none"><li>• BINDADD privilege</li><li>• SYSADM, SYSCTRL, or system DBADM authority</li></ul>
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: <ul style="list-style-type: none"><li>• BINDADD privilege</li><li>• SYSADM, SYSCTRL, or system DBADM authority</li></ul>
REPLACE, using the default owner or primary authorization ID	Primary authorization ID of the process must have one of the following: <ul style="list-style-type: none"><li>• Ownership of the plan</li><li>• BIND privilege on the plan. If the plan does not exist, see the authorization that is required for "ADD, using the default owner or primary authorization ID."</li><li>• SYSADM, SYSCTRL, or system DBADM authority</li></ul>

Table 9. Summary of privileges needed for BIND PLAN options (continued)

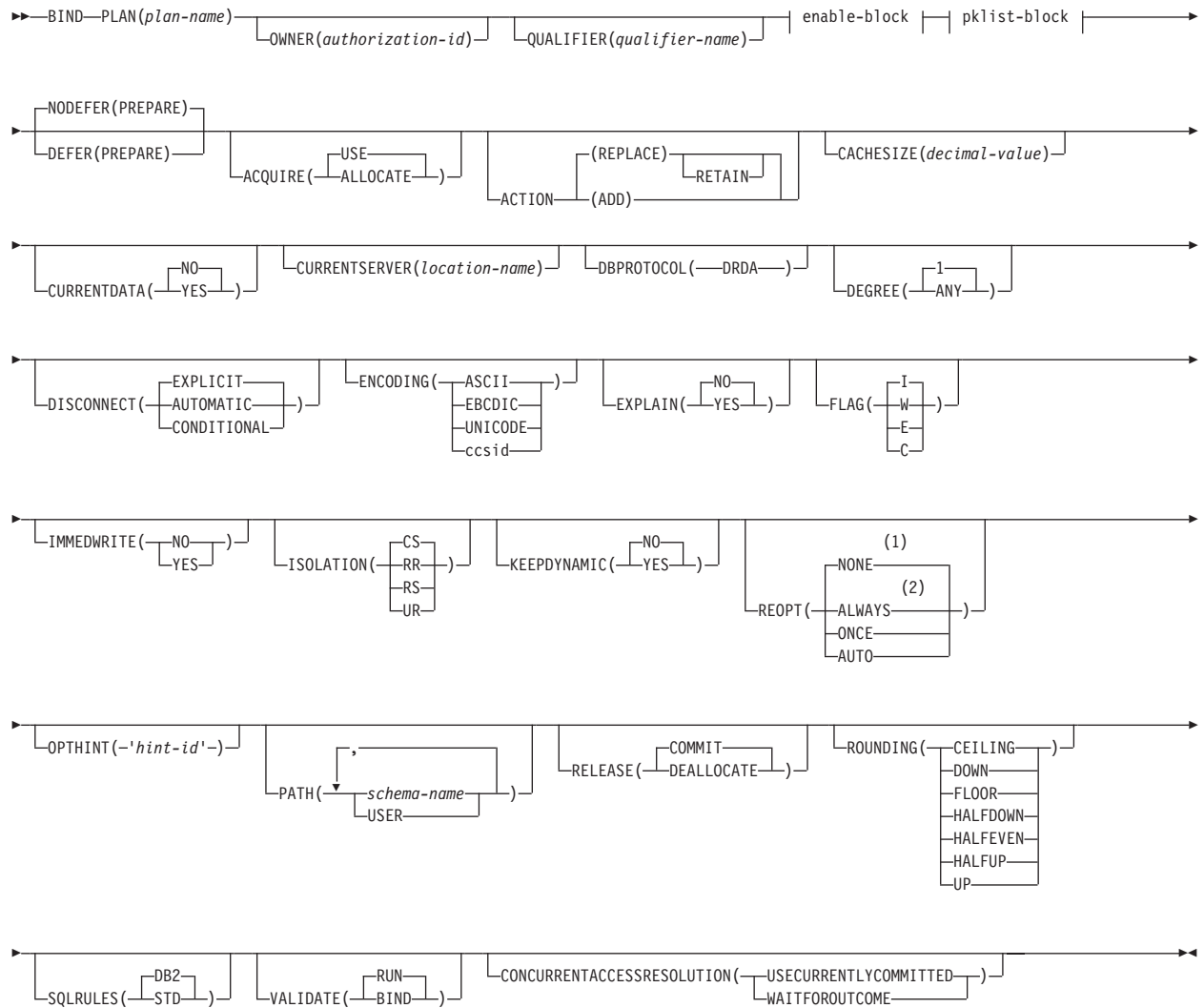
Option	Authorization required to run BIND PLAN
REPLACE, specifying an OWNER other than the primary authorization ID	<p>If the binder does not have SYSADM or SYSCTRL or system DBADM authority, the authorization ID of the OWNER must have one of the following:</p> <ul style="list-style-type: none"> <li>• Ownership of the plan</li> <li>• BIND privilege on the plan. If the plan does not exist, see the authorization that is required for "ADD, specifying an OWNER other than the primary authorization ID."</li> <li>• SYSADM, SYSCTRL, or system DBADM authority</li> </ul>
PKLIST, specifying individual packages	<p>Authorization ID of the process must include one of the following:</p> <ul style="list-style-type: none"> <li>• EXECUTE authority on each package specified in the PKLIST</li> <li>• PACKADM authority on specific collections that contain the packages or on all collections</li> <li>• SYSADM or DATAACCESS authority</li> </ul>
PKLIST, specifying (*), indicating all packages in the collection	<p>Authorization ID of the process must include one of the following:</p> <ul style="list-style-type: none"> <li>• EXECUTE authority on <i>collection-id</i> .*</li> <li>• PACKADM authority on specific collections that contain the packages or on all collections</li> <li>• SYSADM or DATAACCESS authority</li> </ul>
<p><b>Note:</b> 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.

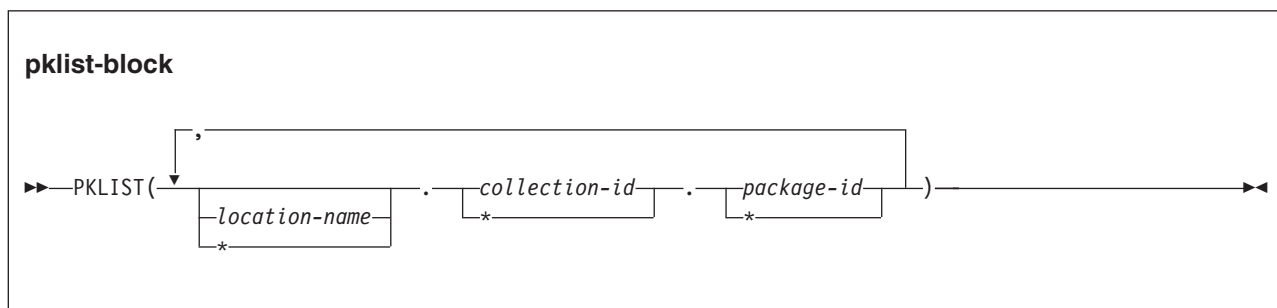
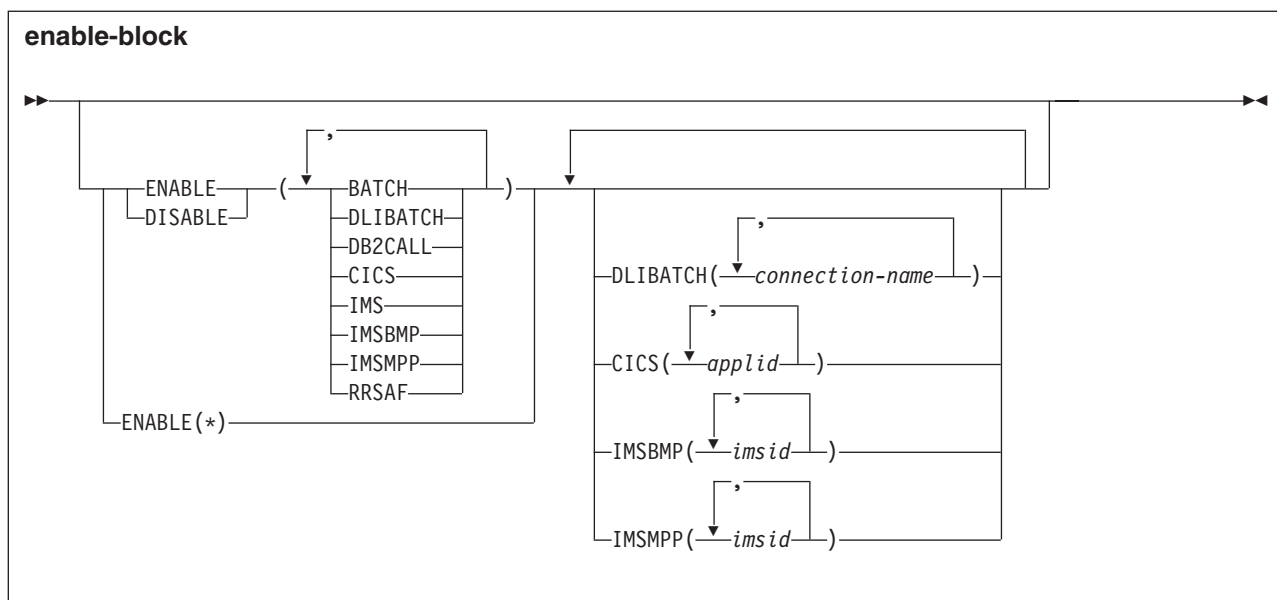


## Syntax



### Notes:

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



## Option descriptions

For descriptions of the options shown in the syntax diagram, see Chapter 19, “BIND and REBIND options,” on page 91.

## 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® node, 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:**

Chapter 2, “The bind process,” on page 5

**Related reference:**

Chapter 19, “BIND and REBIND options,” on page 91

Chapter 16, “BIND PACKAGE (DSN),” on page 73

Chapter 62, “REBIND PLAN (DSN),” on page 411



## Chapter 18. 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 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 types of statement-level optimization hints:

- Access path hints
- Optimization parameter overrides

### 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.

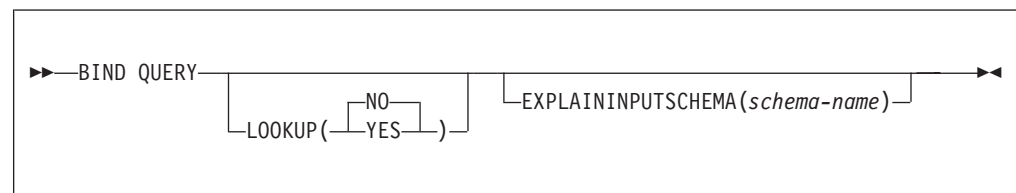
**Data sharing scope:** Group

### Authorization

To 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



### Option description

#### LOOKUP

Specifies whether to check catalog tables for existing statement-level access path or optimization parameter hints that match rows in DSN\_USERQUERY\_TABLE. LOOKUP(NO) is the default value. When LOOKUP(YES) is in effect, new rows are not inserted into the catalog tables and new hints are not created. When matching rows are found in the SYSIBM.SYQUERY and SYSIBM.SYSQUERYPLAN catalog tables, DB2 inserts the value of the SYQUERYPLAN.QUERYID column into the DSN\_USERQUERY\_TABLE.QUERYID column of the matching row.

**(NO)**

DB2 reads the information from the DSN\_USERQUERY\_TABLE and certain EXPLAIN tables and inserts the data into certain catalog tables as appropriate to create the specified statement-level optimization hints. NO is the default value.

Depending on the values that are specified in DSN\_USERQUERY\_TABLE, rows might be read from the following additional input tables:

- *schema*.PLAN\_TABLE

Where *schema* is the authorization ID of the issuer of the BIND QUERY command.

Depending on the values that are specified in the input tables, data might be inserted in the following catalog tables:

- SYSIBM.SYSQUERY
- SYSIBM.SYSQUERYPLAN
- SYSIBM.SYSQUERYOPTS

The catalog table rows create the following types of statement-level optimization hints:

- Access path hints
- Optimization parameter hints

DB2 issues the following messages to indicate the results of BIND QUERY operation:

- A DSNT280I message for each DSN\_USERQUERY\_TABLE row that is inserted successfully into the catalog tables.
- A DSNT281I message for each DSN\_USERQUERY\_TABLE row that is not successfully inserted into the catalog tables.
- A single DSNT290I message if some rows were inserted into catalog tables successfully or a DSNT291I message if no rows were inserted successfully.

#### **(YES)**

DB2 reads information from the DSN\_USERQUERY\_TABLE and looks for the matching rows in the following catalog tables:

- SYSIBM.SYSQUERY
- SYSIBM.SYSQUERYPLAN
- SYSIBM.SYSQUERYOPTS

The catalog tables are not populated with new values and no new hints are created. When matching rows exist in the catalog tables, DB2 inserts the value of the SYQUERYPLAN.QUERYID column into the DSN\_USERQUERY\_TABLE.QUERYID column of the matching row.

New rows are not inserted into the catalog tables when LOOKUP(YES) is specified. Instead, DB2 issues messages to indicate whether existing hints were identified:

- A DSNT280I message for each row in the DSN\_USERQUERY\_TABLE that has a valid matching row in the SYSIBM.SYSQUERY table and the SYSIBM.SYSQUERYPLAN table
- A DSNT281I message for each row in DSN\_USERQUERY\_TABLE that does not have valid matching rows in the SYSIBM.SYSQUERY table and the SYSIBM.SYSQUERYPLAN table.
- A single DSNT290I message if some matching rows were found or a DSNT291I message if no matches were found.

## EXPLAININPUTSCHEMA

Specifies the schema of the EXPLAIN tables to be used for input during BIND QUERY processing. This option 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 hint to be populated 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 hints in the SYSIBM.SYSQUERY and SYSIBM.SYSQUERYOPTS catalog tables.



BIND QUERY

**Related tasks:**





-  Creating statement-level optimization hints and parameters (DB2 Performance)
-  Specifying statement-level optimization parameters (DB2 Performance)
-  Looking up statement-level optimization hints (DB2 Performance)

**Related reference:**

Chapter 19, “BIND and REBIND options,” on page 91

-  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 19. BIND and REBIND options

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.

The following BIND and REBIND options are listed in alphabetical order.

## ACQUIRE

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

Specifies that resources for the packages in the plan are to be acquired when the application first accesses them. This behavior is the default, regardless of whether you specify ACQUIRE(USE), ACQUIRE(ALLOCATE), which has been deprecated, or neither option.

### (USE)

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

### (ALLOCATE)

This option has no affect. 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.

### ACTION

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

Determines whether the object (plan or package) replaces an existing object with the same name or is new.

#### ( 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 affects 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	N/A
REBIND PACKAGE	N/A

**Catalog record:** Tables SYSPLAN or SYSPACKAGE.

**APCOMPARE**

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

Determines whether the new access paths are different from the older access paths.

( NO )  
DB2 does not perform access path comparison.

( NONE )  
DB2 does not perform access path comparison.

( 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.

**Defaults:**

**Process**

**Default value**

**BIND PLAN**

N/A

**BIND PACKAGE**

NONE

**REBIND PLAN**

NONE

**REBIND PACKAGE**

NONE

**REBIND TRIGGER PACKAGE**

NONE

**Automatic rebind**

NONE

**APRETAINDUP**

**APRETAINDUP**

( **YES** )

( **NO** )

**On: REBIND PACKAGE, and  
REBIND TRIGGER PACKAGE**

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. 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

## APREUSE

APREUSE	( <u>NONE</u> ) ( <u>NO</u> ) ( ERROR )	On: BIND PACKAGE, REBIND PACKAGE, and REBIND TRIGGER PACKAGE
---------	---	--

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. 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 is ended, 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.

### Defaults:

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

**CACHESIZE**

<b>CACHESIZE</b>	( <i>value of field PLAN AUTH CACHE</i> ) ( <i>decimal-value</i> )	<b>On: BIND and REBIND PLAN</b>
------------------	---	---------------------------------

Determines the size (in bytes) of the authorization cache acquired in the EDM pool for the 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:**

<b>Process</b>	<b>Default value</b>
<b>BIND PLAN</b>	<i>value of field PLAN AUTH CACHE</i> on installation panel DSNTIPP, which has a default of 0
<b>BIND PACKAGE</b>	N/A
<b>REBIND PLAN</b>	Existing value
<b>REBIND PACKAGE</b>	N/A

**Catalog record:** Column CACHESIZE of table SYSPLAN.

**COLLID**

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

Specifies that any DBRMs in the plan are to be bound to packages. 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 Version 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 Version 10 and has DBRMs bound directly into it, COLLID(\*) is in effect.

**Catalog record:** Tables SYSPACKAGE, SYSPACKAUTH, SYSPACKDEP, SYSPACKSTMT, and SYSPACKSYSTEM.

**CONCURRENTACCESSRESOLUTION**

CONCURRENTACCESSRESOLUTION	( WAITFOROUTCOME ) ( USECURRENTLYCOMMITTED )	On: BIND and REBIND PLAN and PACKAGE, REBIND TRIGGER PACKAGE
----------------------------	---	---

Specifies which concurrent access resolution option to use for statements in a 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.

If USECURRENTLYCOMMITTED is specified for a package, XML data is being selected, and that data does not support multiple XML versions, the database manager cannot determine whether the data has been committed. Therefore, the database manager uses WAITFOROUTCOME behavior when it accesses the data.

**( 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 10. 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

<b>COPY</b>	( <i>collection-id.package-id</i> ) ( <i>collection-id.package-id</i> ) <b>COPYVER</b>	<b>On: BIND PACKAGE</b>
-------------	---	-------------------------

Determines that you are copying an existing package and names that 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 Version 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<sup>®</sup> 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

CURRENTDATA	( NO ) ( YES )	On: BIND and REBIND PLAN and PACKAGE, REBIND TRIGGER PACKAGE
-------------	-------------------	--

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.

( 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

CURRENTSERVER	( <i>location-name</i> )	On: BIND and REBIND PLAN
---------------	--------------------------	--------------------------

Determines the location to connect to before running the 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 DBMS (regardless of the name of the local location)

**REBIND PLAN**

Existing value

**Catalog record:** Column CURRENTSERVER of table SYSPLAN.

## DBPROTOCOL

DBPROTOCOL	( <u>DRDA</u> )	On: BIND and REBIND PLAN and PACKAGE
------------	-----------------	--------------------------------------

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.

A copy of the package must also be bound to each remote site that is referenced by a three-part name 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. This keyword is the default.

**Catalog record:** Column DBPROTOCOL of tables SYSPACKAGE and SYSPLAN.

## NODEFER and DEFER

NODEFER DEFER	NODEFER(PREPARE) DEFER(PREPARE)	On: BIND and REBIND PLAN and PACKAGE
DEFER(INHERITFROMPLAN)		

Determines whether to defer preparation for dynamic SQL statements that refer to remote objects, or to prepare them immediately. 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**

DEGREE	( 1 ) ( ANY )	On: BIND and REBIND PLAN and PACKAGE
--------	------------------	---

Determines whether to attempt to run a query using parallel processing to maximize performance.

For plans, DEGREE has no affect.

The value has no effect on dynamic SQL statements, which use the value of the special register CURRENT DEGREE. The value of the special register can be changed by executing the SET CURRENT DEGREE statement.

( 1 ) Prohibits parallel processing.

( ANY )

Allows parallel processing.

**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**

DEPLOY	(collection-id.package-id ) COPYVER(version-id)	On: BIND PACKAGE
--------	--	------------------

|

Deploys a non-inline SQL function or a native SQL procedure.

|

|

|

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.

You can specify the target collection-id, QUALIFIER, ACTION, TARGET-LOCATION, and OWNER options on the BIND PACKAGE DEPLOY command. If you specify any other bind option DB2 returns message DSNT215I. If

you do not specify QUALIFIER and OWNER their default values are the authorization ID that issues the BIND 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:

##### **BINDADD and CREATEIN**

Necessary authorization if a version of a native SQL procedure or a non-inline SQL function, and the associated package, exist at the target location.

##### **BINDADD and ALTERIN**

Necessary authorization if a version of a native SQL procedure or a non-inline SQL function, and the associated package that has a different version ID, exist at the target location.

To issue a BIND DEPLOY ACTION(REPLACE) command, the following authorization is necessary:

##### **BINDADD and CREATEIN**

Necessary authorization if the object and its package do not exist at the target location.

#### **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

DISCONNECT	( <b>EXPLICIT</b> ) ( <b>AUTOMATIC</b> ) ( <b>CONDITIONAL</b> )	On: BIND and REBIND PLAN
------------	---	--------------------------

Determines which remote connections to destroy during commit operations. 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

DYNAMICRULES	( <b>RUN</b> ) ( <b>BIND</b> ) ( <b>DEFINEBIND</b> ) ( <b>DEFINERUN</b> ) ( <b>INVOKEBIND</b> ) ( <b>INVOKERUN</b> )	On: BIND and REBIND PACKAGE
--------------	---	--------------------------------

Determines what values apply at run time for the following dynamic SQL attributes:

- 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 11 on page 110.

#### ( 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 11. 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

Table 11. Definitions of dynamic SQL statement behaviors (continued)

Dynamic SQL attribute	Value for bind behavior	Value for run behavior	Value for define behavior	Value for invoke behavior
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

## ENABLE and DISABLE

ENABLE	DISABLE	On: BIND and REBIND PLAN and PACKAGE
( * )		
( BATCH )		
( CICS )		
( CICS ) CICS ( applid , ...)		
( DB2CALL )		
( DLIBATCH )		
( DLIBATCH ) DLIBATCH ( connection-name , ...)		
( IMS )		
( IMSBMP )		
( IMSBMP ) IMSBMP ( imsid , ...)		
( IMSMPP )		
( IMSMPP ) IMSMPP ( imsid , ...)		
( REMOTE ) (BIND and REBIND PACKAGE only)		
( REMOTE ) REMOTE ( location-name ,..., < luname >,...)		
( RRSF )		

Determines which connections can use the plan or package. 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<sup>®</sup>. 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.

**( REMOTE ) REMOTE ( *location-name* ,...,< *luname* >,...)** **(PACKAGE only)**

Specifies that the remote connections identified by the following are either enabled or disabled for the package:

*location-name*

Specifies the location name of a requesting DBMS that **is** a DB2 for z/OS subsystem.

< *luname* >

Specifies the logical unit name, as defined to VTAM at the server location, of a requesting DBMS that **is not** a DB2 for z/OS subsystem.

You must bracket a logical unit name with the less than (<) and the greater than (>) characters to differentiate it from a location name.

**( RRSF )**

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.

**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**

ENCODING	( ASCII )( EBCDIC )( UNICODE )( <i>ccsid</i> )	On: BIND and REBIND PLAN and PACKAGE
----------	--	--------------------------------------

Specifies the application encoding for all host variables in static statements in the plan or package. EBCDIC is the only valid option for a plan or package that was precompiled prior to Version 7. If you specify *ccsid* on any plan or package precompiled prior to 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 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.

#### 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

The value that was specified the last time that the plan was bound.

#### REBIND PACKAGE

The value that was specified the last time the package was bound.

**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

PSPI

EXPLAIN ( NO ) ( YES ) ( ONLY )

On BIND and REBIND PLAN and  
PACKAGE, REBIND TRIGGER  
PACKAGE

Obtains information about how SQL statements in the package or packages are to execute. It 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

Existing value

##### REBIND PACKAGE

Existing value

**Catalog record:** Column EXPLAIN of table SYSPACKAGE and column EXPLAIN of SYSPLAN.

## EXTENDEDINDICATOR

EXTENDEDINDICATOR	( <u>NO</u> )	On: BIND and REBIND PACKAGE
	( YES )	

Determines whether DB2 recognizes extended indicator variables when the associated package is run.

( 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

## FILTER

FILTER	( ' <u>filter-name</u> ' )	On: FREE QUERY
--------	----------------------------	----------------

Lets you delete a set of queries in the SYSIBM.SYSQUERY table under a {tag} value specified by the SYSQUERY.USERFILTER column.

( 'filter-name' )

## FLAG

FLAG	( <u>I</u> )( W )( E )( C )	On: BIND and REBIND PLAN and PACKAGE, REBIND TRIGGER PACKAGE
------	-----------------------------	--

Determines what messages to display.

( 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

**GENERIC**

GENERIC	( 'string' )	On: BIND and REBIND PACKAGE
	<p>Specifies one or more bind options that are supported by the target server, but are not supported by DB2 for z/OS as options for BIND PACKAGE or REBIND PACKAGE.</p> <p>Do not use GENERIC to specify any of the following options:</p> <ul style="list-style-type: none"> <li>• bind options that are explicitly supported by DB2 for z/OS for BIND PACKAGE or REBIND PACKAGE</li> <li>• SQL processing options that are explicitly supported by DB2 for z/OS</li> </ul> <p>( 'string ' )</p> <p>One or more pairs of a bind option and its corresponding value. Separate each item by one or more blank spaces.</p> <p>For example, the following GENERIC option specifies several bind options and their corresponding values:</p> <pre>GENERIC( ' option-1 value-1 option-2 value-2..... option-n value-n ' )</pre> <p>In this example, <i>value-1</i> is specified for bind option <i>option-1</i>, <i>value-2</i> is specified for bind option <i>option-2</i>, and so on.</p> <p>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:</p> <pre>GENERIC( ' firstoption firstvalue secondoption secondvalue firstoption thirdvalue' )</pre> <p>In this example, <i>firstoption</i> and <i>secondoption</i> are bind options that are sent to the target server with their corresponding values. The third option</p>	

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.

## IMMEDWRITE

IMMEDWRITE	( NO ) ( YES ) ( <u>INHERITFROMPLAN</u> )	On: BIND and REBIND PLAN and PACKAGE
------------	---	--------------------------------------

Indicates whether immediate writes are to be done for updates that are made to group buffer pool dependent page sets or partitions. 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 12. The implied hierarchy of the IMMEDIATEWRITE option

IMMEDWRITE bind option	IMMEDWRITE subsystem parameter	Value at run time
NO	NO	NO
NO	PH1	PH1
NO	YES	YES
PH1	NO	PH1

Table 12. The implied hierarchy of the IMMEDIATE option (continued)

IMMEDIATE bind option	IMMEDIATE subsystem parameter	Value at run time
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.

**Performance hints:** 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

##### BIND PACKAGE

INHERITFROMPLAN

##### REBIND PLAN

Existing value

##### REBIND PACKAGE

Existing value

The default for a package on a remote DB2 server is IMMEDIATEWRITE(NO).

## ISOLATION

ISOLATION	( <u>CS</u> )( RS )( RR )( UR )( NC )	On: BIND and REBIND PLAN and PACKAGE, REBIND TRIGGER PACKAGE
-----------	---------------------------------------	--

Determines how far to isolate an application from the effects of other running applications.

( 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

Plan value for a local server; CS for a remote server.

## REBIND PLAN

Existing value

## REBIND PACKAGE

Existing value

For **REBIND PACKAGE**, 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.

## KEEPDYNAMIC

KEEPDYNAMIC	( <u>NO</u> )( YES )	On: BIND and REBIND PLAN and PACKAGE
-------------	----------------------	--------------------------------------

Determines whether DB2 keeps dynamic SQL statements after commit points.

( 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 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 KEEPDYNAMIC(YES), you must not specify REOPT(ALWAYS). KEEPDYNAMIC(YES) and REOPT(ALWAYS) are mutually exclusive. However, you can use KEEPDYNAMIC(YES) with REOPT(ONCE).

**Performance hint:** KEEPDYNAMIC(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

The default for a package on a remote DB2 server is KEEP\_DYNAMIC(NO).

**Catalog record:** Column KEEP\_DYNAMIC of table SYSPLAN and SYSPACKAGE.

## LIBRARY

<b>LIBRARY</b>	( <i>dbrm-pds-name</i> )( <i>dbrm-pds-name</i> , ... ) (BIND PLAN only)	<b>On: BIND PLAN, BIND PACKAGE</b>
----------------	---	------------------------------------

Determines what partitioned data sets (libraries) to search for the DBRMs listed in the MEMBER option. 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**  
None

**BIND PACKAGE**  
None

**REBIND PLAN**  
N/A

**REBIND PACKAGE**  
N/A

The default is to search only the libraries described by the DD statement for DBRMLIB.

## LOOKUP

<b>LOOKUP</b>	(NO (YES	<b>On: BIND QUERY</b>
---------------	-------------	-----------------------

Specifies whether to check catalog tables for existing statement-level access path hints that match rows in DSN\_USERQUERY\_TABLE. When LOOKUP(YES) is specified, new rows are not inserted into the catalog tables and new hints not are created. When matching rows are found in the SYSIBM.SYQUERY and SYSIBM.SYSQUERYPLAN catalog tables, DB2 inserts the value of the SYQUERYPLAN.QUERYID column into the DSN\_USERQUERY\_TABLE.QUERYID column of the matching row.

**( NO )**

Reads the information from the DSN\_USERQUERY\_TABLE and inserts the data into the SYSIBM.SYSQUERY and SYSIBM.SYSQUERYPLAN or SYSIBM.SYSQUERYOPTS tables as appropriate to create the specified statement level optimization hints.

DB2 issues the following messages to indicate the results of BIND QUERY operation:

- A DSNT280I message for each DSN\_USERQUERY\_TABLE row that is inserted successfully into the catalog tables.
- A DSNT281I message for each DSN\_USERQUERY\_TABLE row that is not successfully inserted into the catalog tables.
- A single DSNT290I message if some rows were inserted into catalog tables successfully or a DSNT291I message if no rows were inserted successfully.

**( YES )**

Reads information from the DSN\_USERQUERY\_TABLE and looks for the matching rows in the SYSIBM.SYSQUERY and SYSIBM.SYSQUERYPLAN catalog tables. The catalog table are not populated with new values and no new hints are created. When matching rows exist in the catalog tables, DB2 inserts the value of the SYQUERYPLAN.QUERYID column into the DSN\_USERQUERY\_TABLE.QUERYID column of the matching row.

New rows are not inserted into the catalog tables when LOOKUP(YES) is specified. Instead, DB2 issues messages to indicate whether existing hints where identified:

- A DSNT280I message for each row in the DSN\_USERQUERY\_TABLE that has a valid matching row in the SYSIBM.SYSQUERY table and the SYSIBM.SYSQUERYPLAN table
- A DSNT281I message for each row in DSN\_USERQUERY\_TABLE that has valid matching rows are identified in both the SYSIBM.SYSQUERY table and the SYSIBM.SYSQUERYPLAN table.
- A single DSNT290I message if some matching rows were found or a DSNT291I message if no matches were found.

When you specify LOOKUP(YES) only rows for statement-level access path hints are identified. Rows in the SYSIBM.SYSQUERY which correspond to rows in the SYSIBM.SYSQUERYOPTS catalog table for statement-level optimization parameters are not detected.

**Defaults:**

**Process**

**Default value**

**BIND PLAN**

N/A

**BIND PACKAGE**

N/A

NO

N/A

N/A

<b>MEMBER</b>	( <i>dbrm-member-name</i> )( <i>dbrm-member-name</i> , ... ) (BIND PLAN only)	<b>On: BIND PLAN, BIND PACKAGE</b>
---------------	---	------------------------------------

N/A

## OPTHINT

OPTHINT	(' hint-id ')	On: BIND and REBIND PLAN and PACKAGE
---------	---------------	--------------------------------------

Controls whether query optimization hints are used for static SQL. 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, which is used by the optimizer when searching the PLAN\_TABLE for rows to use as input to the optimizer. 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 in the OPTIMIZATION HINTS field of installation panel DSNTIP8.

**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

The default for a package on a remote server is all blanks.

**Catalog record:** Column OPTHINT of tables SYSPLAN and SYSPACKAGE.

## OPTIONS

OPTIONS	( <u>COMPOSITE</u> ) (COMMAND)	On: BIND PACKAGE COPY
---------	--------------------------------	-----------------------

Specifies which bind options to use for the new package.

### COMPOSITE

The options for the new package are what you specify on the BIND PACKAGE COPY subcommand. Options that you do not specify are the option values taken from 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 Version 8 or higher.

## COMMAND

The options for the new package are what you specify on the BIND PACKAGE COPY subcommand. Options that you do not specify 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 PACKAGE COPY**  
COMPOSITE

## OWNER

OWNER	( <i>authorization-id</i> )	On: <b>BIND and REBIND PLAN and PACKAGE</b>
-------	-----------------------------	---

Determines the authorization ID of the owner of the object (plan or 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.

**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 ID

**BIND PACKAGE**

Primary ID

**REBIND PLAN**

Existing value

**REBIND PACKAGE**

Existing value

The default owner is the primary authorization ID of the agent that runs the bind process.

**Catalog record:** Column OWNER of table SYSPACKAGE, column GRANTOR of table SYSPACKAUTH, and column CREATOR of table SYSPLAN.

**PACKAGE**

<b>PACKAGE</b>	( <i>location-name.collection-id</i> ) (BIND PACKAGE), ( <i>location-name.collection-id.package-id.(version-id)</i> ) (REBIND PACKAGE)	<b>On: BIND and REBIND PACKAGE</b>
----------------	--	------------------------------------

Determines what package or packages to bind or rebind.

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

PATH	( <i>schema-name</i> ) ( USER ) ( <i>schema-name</i> , USER , ...)	On: BIND and REBIND PLAN and PACKAGE
------	--	---

Determines the SQL path that DB2 uses to resolve unqualified user-defined distinct types, functions, and stored procedure names (in CALL statements).

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.

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.

( *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 8-byte *schema-name* . At bind time, DB2 includes this 8-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 8-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.

## Defaults:

### Process

Default value

### BIND PLAN

"SYSIBM," "SYSFUN," "SYSPROC," *plan qualifier*

### BIND PACKAGE

"SYSIBM," "SYSFUN," "SYSPROC," *package qualifier*

### REBIND PLAN

Existing value

### REBIND PACKAGE

Existing value

Although *plan qualifier* is the default value for BIND PLAN, it is not stored in the catalog. Instead, the catalog value is blank. The catalog value is also blank for *package qualifier*.



| 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

PATHDEFAULT	On: REBIND PLAN and PACKAGE										
<p>Resets the <b>PATH</b> for a package or plan to “SYSIBM,” “SYSFUN,” “SYSPROC,” or <i>plan qualifier</i> or <i>package qualifier</i> .</p> <p>The <b>PATHDEFAULT</b> keyword is mutually exclusive with the <b>PATH</b> keyword. Do not specify both keywords in the same REBIND command.</p> <p><b>Defaults:</b></p> <p><b>Process</b></p> <table> <tr> <th></th><th>Default value</th></tr> <tr> <td><b>BIND PLAN</b></td><td>N/A</td></tr> <tr> <td><b>BIND PACKAGE</b></td><td>N/A</td></tr> <tr> <td><b>REBIND PLAN</b></td><td>None</td></tr> <tr> <td><b>REBIND PACKAGE</b></td><td>None</td></tr> </table>			Default value	<b>BIND PLAN</b>	N/A	<b>BIND PACKAGE</b>	N/A	<b>REBIND PLAN</b>	None	<b>REBIND PACKAGE</b>	None
	Default value										
<b>BIND PLAN</b>	N/A										
<b>BIND PACKAGE</b>	N/A										
<b>REBIND PLAN</b>	None										
<b>REBIND PACKAGE</b>	None										

## PKLIST and NOPKLIST

PKLIST NOPKLIST	( <i>location-name.collection-id.package-id</i> , ...) PKLIST only	On: BIND and REBIND PLAN
<p>PKLIST determines what packages to include in the package list for the plan. The order in which you list packages with partial identifiers determines the search order at run time and can affect performance.</p> <p>NOPKLIST is used with REBIND PLAN only. NOPKLIST determines that the plan rebinds without a package list. If a package list already exists, NOPKLIST deletes it.</p> <p>If you specify REBIND PLAN for a plan that was bound prior to Version 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 affects:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> </ul>		

*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

**BIND PACKAGE**

N/A

**REBIND PLAN**

Existing value

**REBIND PACKAGE**

N/A

PKLIST has no default. If you do not use PKLIST, you must use MEMBER.

The default for NOPKLIST is to use the package list specified in the PKLIST option, if any, during the current or previous bind or rebind.

**Catalog record:** Table SYSPACKLIST.

**PLAN**

<b>PLAN</b>	( <i>plan-name</i> ) * (*) (REBIND PLAN only)	<b>On: BIND and REBIND PLAN</b>
-------------	---	---------------------------------

Determines what plan or plans to bind or rebind.

( *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.

The default is to perform all bind functions, including error diagnostics, without producing an application plan and without inserting rows into PLAN\_TABLE for the option EXPLAIN.

( \* ) (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

PLANMGMT 	( OFF )	On: REBIND PACKAGE and REBIND TRIGGER PACKAGE
	( BASIC )	
	( EXTENDED )	

Retains, during a rebind operation, all relevant package information (metadata, query text, dependencies, authorizations, access paths, and so on) in catalog tables and in the directory. In the event of a performance regression, you can direct DB2 to fall back to the older copy of a package. Over time, a package can have multiple copies that exist disk storage, but only one of those copies is the active or current copy. All other copies are inactive.

( OFF )

No access paths are retained. Any access paths that existed before the BIND or REBIND command is issued with this option are purged. Previous or original copies are also 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 already exists, it remains available.

(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.

**Behavior of a rebind operation with PLANMGMT (OFF):** With

PLANMGMT(OFF), DB2 normally only affects the current package copy. If any previous or original copies exist, they remain untouched. However, if you change any of the following REBIND options, all package copies (current, previous, and original) are purged:

- OWNER

- QUALIFIER
- ENABLE
- DISABLE
- PATH
- PATHDEFAULT
- IMMEDIATEWRITE

**Behavior of a rebind operation with PLANMGMT (BASIC) or PLANMGMT (EXTENDED):** If you change any of the following options with PLANMGMT (BASIC) or PLANMGMT (EXTENDED), DB2 issues an error message:

- OWNER
- QUALIFIER
- ENABLE
- DISABLE
- PATH
- PATHDEFAULT
- IMMEDIATEWRITE

**Behavior of a rebind operation in which PLANMGMT is not specified:** If the PLANMGMT option is not specified, the rebind operation takes the value from the subsystem parameter PLANMGMT.

**Wildcards:**PLANMGMT settings remain valid when wildcards (\*) 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**

Existing value

**REBIND TRIGGER PACKAGE**

Existing value

**Catalog record:** Column PLANMGMT of table SYSPACKAGE.

**QUALIFIER**

QUALIFIER	( <i>qualifier-name</i> )	On: BIND and REBIND PLAN and PACKAGE
-----------	---------------------------	--------------------------------------

Determines the implicit qualifier for unqualified names of tables, views, indexes, and aliases contained in the plan or 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.

**Defaults:**

**Process**

**Default value**

**BIND PLAN**

Owner ID

**BIND PACKAGE**

Owner ID

**REBIND PLAN**

Existing value

**REBIND PACKAGE**

Existing value

The default is the owner's authorization ID, whether you use the OWNER option or its default.

**Catalog record:** Column QUALIFIER of tables SYSPACKAGE and SYSPLAN.

**QUERYID**

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**

RELEASE	( COMMIT ) ( DEALLOCATE ) ( INHERITFROMPLAN )	On: BIND and REBIND PLAN and PACKAGE, REBIND TRIGGER PACKAGE
---------	---	--

Determines when to release resources that a program uses, either at each commit point or when the program terminates.

( **COMMIT** )

Releases resources at each commit point.

**DEALLOCATE**

Releases resources only when the program terminates.

RELEASE(DEALLOCATE) has no effect on dynamic SQL statements, which always use RELEASE(COMMIT), with one exception: When you use RELEASE(DEALLOCATE) and KEEPDMYMIC(YES), and your subsystem is installed with YES for field CACHE DYNAMIC SQL on installation panel

DSNTIP8, 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 RRSAP.
- 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**

Plan value

##### **REBIND PLAN**

Existing value

##### **REBIND PACKAGE**

Existing value

COMMIT is the default for a package that is bound at a remote server.

**Catalog record:** Column RELEASE of tables SYSPACKAGE and SYSPLAN.

## REOPT

REOPT	( <u>NONE</u> ) ( ALWAYS ) ( ONCE ) ( AUTO )	On: BIND and REBIND PLAN and PACKAGE
-------	---	--------------------------------------

Specifies whether to have DB2 determine an access path at run time by using the values of host variables, parameter markers, and special registers. 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 KEEP DYNAMIC(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 KEEP\_DYNAMIC(YES).

**Defaults:**

Process	Default value
<b>BIND PLAN</b>	NONE
<b>BIND PACKAGE</b>	NONE
<b>REBIND PLAN</b>	Existing value
<b>REBIND PACKAGE</b>	Existing value

REOPT(NONE) is the default for a package that is bound on a remote DB2 server.

**Catalog record:** Column REOPTVAR of table SYSPLAN and SYSPACKAGE.

**ROUNDING**

ROUNDING		On: BIND and REBIND PLAN and PACKAGE
	( CEILING )	
	( DOWN )	
	( FLOOR )	
	( HALFDOWN )	
	( <u>HALFEVEN</u> )	
	( HALFUP )	
	( UP )	

Specifies the rounding mode at bind time. Use rounding mode to manipulate DECFLOAT data.

**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**

SQLERROR	(NOPACKAGE) (CONTINUE) (CHECK)	On: BIND PACKAGE
----------	--------------------------------------	------------------

Determines whether to create a package if SQL errors occur.

**(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)**, those SQL statements found in error at bind time do not rebind.

**Catalog record:** Column **SQLERROR** of table **SYSPACKAGE**.

**SQLRULES**

<b>SQLRULES</b>	<b>( DB2 )</b> <b>( STD )</b>	<b>On: BIND and REBIND PLAN</b>
-----------------	----------------------------------	---------------------------------

Determines whether you can execute a type 2 **CONNECT** statement to an existing SQL connection, according to DB2 rules. 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**

SWITCH	( PREVIOUS )( ORIGINAL )	On: REBIND PACKAGE and REBIND TRIGGER PACKAGE
--------	--------------------------	--

Restores all previous or original package information in the catalog tables and directory to that of the specified package copy. This option allows you to fallback 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 you specified does not have the previous or original copy to switch to, DB2 will issue an error message, and will then continue processing the rest of the list. Use this option with wildcards (\*) 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	NO
BIND PACKAGE	NO
REBIND PLAN	Existing value
REBIND PACKAGE	Existing value

## VALIDATE

VALIDATE	( RUN ) ( BIND )	On: BIND and REBIND PLAN and PACKAGE
----------	---------------------	---

Determines whether to recheck, at run time, errors of the type "OBJECT NOT FOUND" and "NOT AUTHORIZED" found during bind or rebind. 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.

### Related concepts:

➞ CURRENT OPTIMIZATION HINT (DB2 SQL)

➞ Bind options for locks (DB2 Performance)

### Related tasks:

➞ BIND options for distributed applications (DB2 Performance)

### Related reference:

➞ Bind options for remote access (DB2 Application programming and SQL)

---

## 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

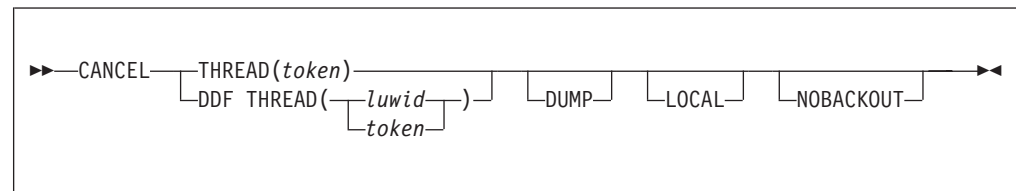
### Authorization

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

- SYSOPR authority
- 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

#### 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, then 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.

You might have two or more distributed threads with 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**

Provides a dump 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
<b>IST635I</b>	<b>LUND1</b>	<b>ACTIV-R</b>	<b>D2D3590EA1E89701</b>	<b>0022</b>	<b>0031</b>	<b>0</b>	<b>0</b>	<b>NET2</b>
IST635I	LUND1	ACTIV-R	D2D3590EA1E89802	0022	0031	0	0	NET2
<b>IST635I</b>	<b>LUND1</b>	<b>ACTIV-R</b>	<b>D2D3590EA1E89822</b>	<b>0022</b>	<b>0031</b>	<b>0</b>	<b>0</b>	<b>NET2</b>

```
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 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 TR      4 CTHDCORID001 SYSADM   DONSQ11   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 TCP/IP
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, STMT=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 (VTAM Operation)

 VTAM VARY NET,TERM command (VTAM 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 150
- “Examples” on page 150

### Environment

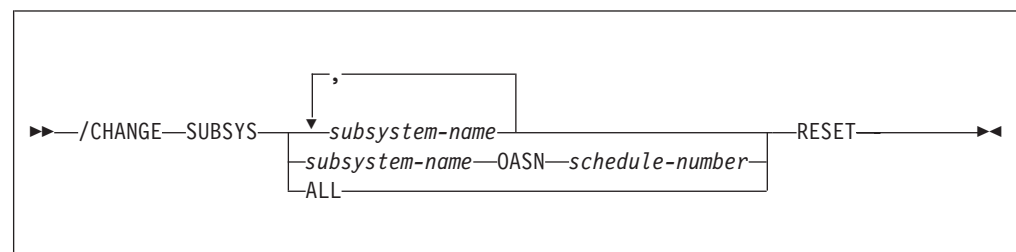
This command can be issued only from an IMS terminal.

**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.

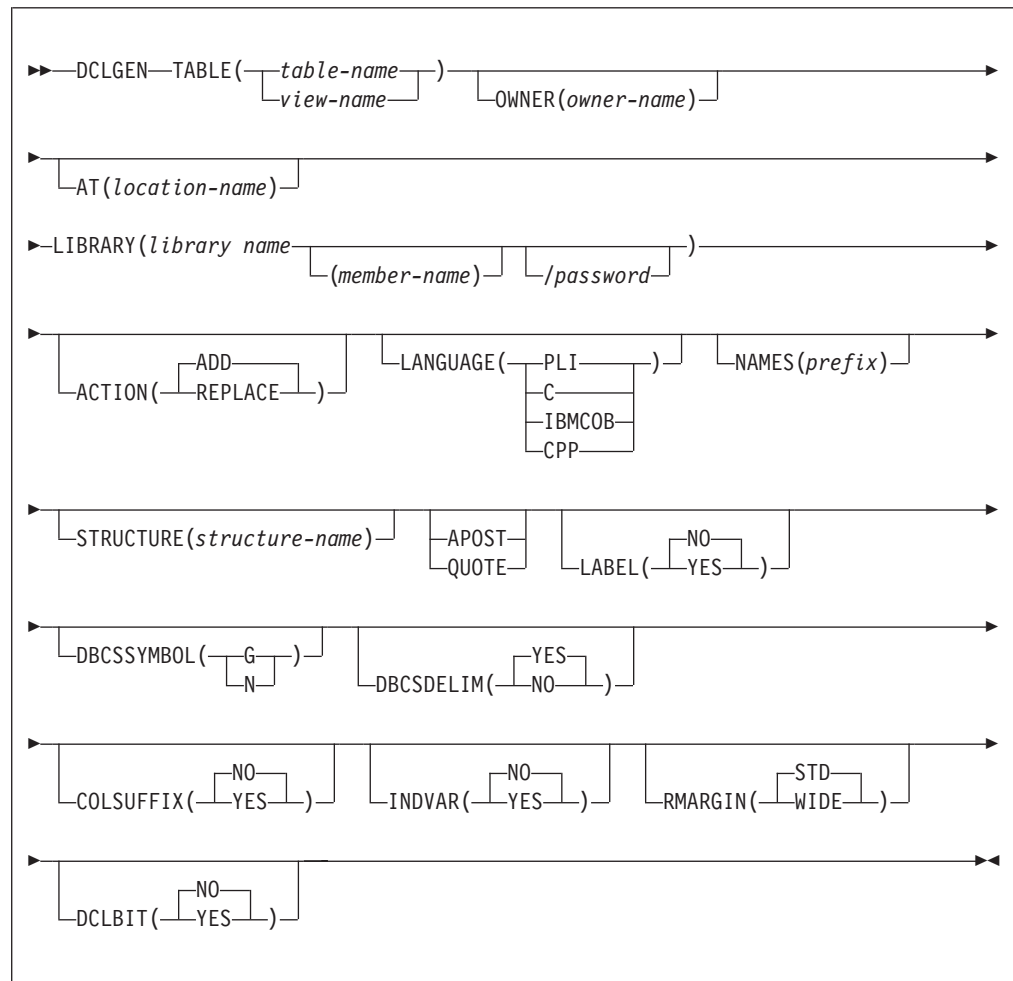
**Data sharing scope:** Group

### 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 13. 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, or C;

( *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 the COBOL declaration, which can result in truncated values.

For small integers that can exceed 9999, use S9(5). For large integers that can exceed 999999999, use S9(10) COMP-3 to obtain the decimal data type. If COBOL is used for integers that exceed the COBOL PICTURE, specify the column as decimal to ensure that the data types match and perform well.

**COBOL and the underscore character:** Because COBOL does not allow the use of 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:** OS/VS COBOL does not support DBCS, but later versions of COBOL (VS COBOL II and COBOL/370) do. 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 *table-name* [ *n* ];

**Cobol** 01 I *table-name* PIC S9(4) USAGE COMP OCCURS *n* TIMES

**PL/I** DCL I *table-name* ( *n* ) BIN FIXED (15);

Where *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(VEEMPL) -
/*     LIBRARY('prefix.SRCLIB.DATA(DSN8MPPEM)') -
/*     LANGUAGE(PLI) -
/*     NAMES(FIELD) -
/*     STRUCTURE(EMPREGORD) -
/*     APOST
/* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS
*****/
EXEC SQL DECLARE VEEMPL 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 VEEMPL
*****/
DCL 1 EMPREGORD,
      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)

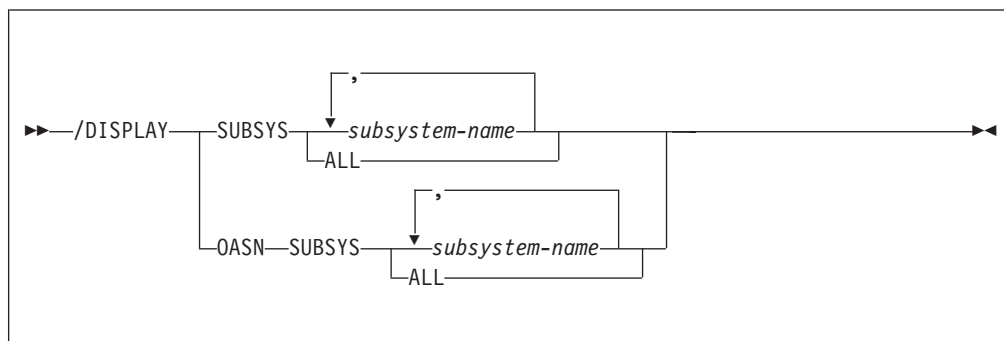
The IMS command /DISPLAY displays the status of the connection between IMS and an external subsystem (as well as all application programs communicating with the external subsystem), or the outstanding recovery units that are associated with the subsystem.

**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:

### 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:



**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	DDLTL17	PTERM01	CONN,ACTIVE
		2	DDLTL06	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 using the instrumentation facility interface (IFI).

**Data sharing scope:** Group or local, depending on the SCOPE option.

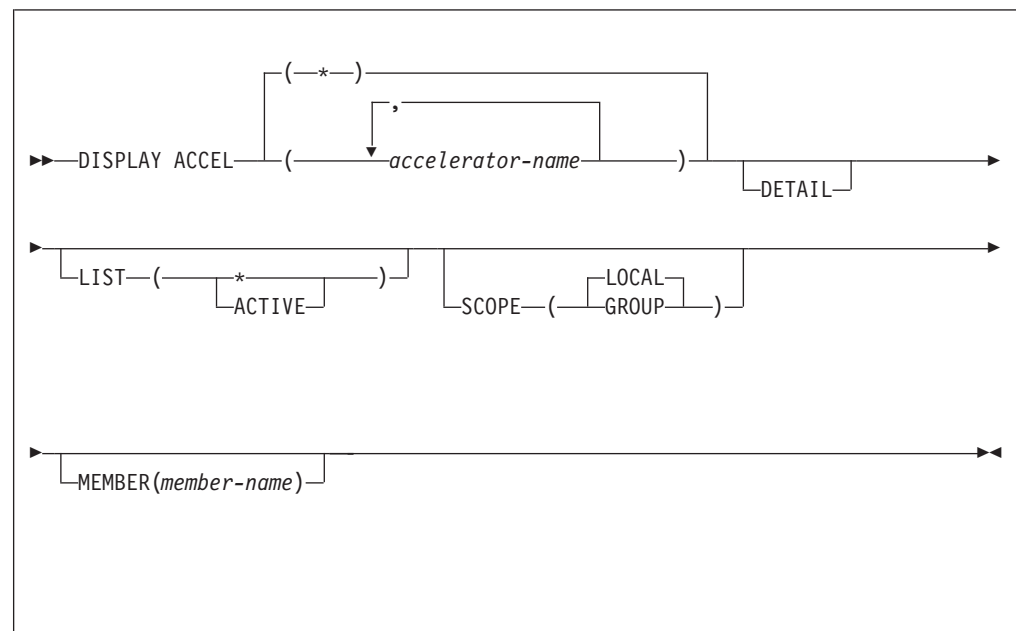
### Authorization

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

- DISPLAY 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

**(*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. Supplying an \* 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 which have been started. There are two accepted values.

**(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. Remember that an accelerator that has just been created appears in the list only if it has been started using the START ACCEL command.

### SCOPE

Specifies the scope of the command. In a non-data-sharing environment, the option is ignored. There are two accepted values.

**(LOCAL)**

Displays accelerator servers that are on the current member.

**(GROUP)**

Displays accelerator servers that are on the data sharing group.

### MEMBER

Restricts the display for the identified accelerator server to specific members of the data sharing group. The default is to display accelerator servers on the local member. In a non-data-sharing environment, the option is ignored.

## Output

Message DSNX830I indicates the beginning of the output of the command.

## Examples

**Example: Display 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 ACC(*) SCOPE(GROUP)`

The output is similar to the following output:

```
DSNX830I - DSNX8CMD DISPLAY ACCELERATOR FOLLOWS -
ACCELERATOR          MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
ACCEL1                DB1A  STARTED    32769     2     5    23
ACCEL1                DB1B  STARTED    23456     1     0     2
ACCEL1                DB1C  STARTED     734     0     0     4
ACCEL1                DB1D  STARTED    9210     0     0     1
ACCEL2                DB1A  STOPPED    37235     1     7    17
```

```

ACCEL2          DB1B STOPPED      47    0    0    0
ACCEL2          DB1C STARTED       2    0    0    0
ACCEL2          DB1D STOPPED       0    0    0    0
ACCEL3          DB1A STARTED     3256    5   23   41
ACCEL3          DB1B STOPPED      92    0    0    2
ACCEL3          DB1C STARTED      87    0    0    7
ACCEL3          DB1D STOPPED      21    0   11   11
DISPLAY ACCELERATOR REPORT COMPLETE
DSN9022I ) DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION

```

**Example: 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 ACC(ACCEL1,ACCEL2)
  DETAIL
  LIST(ACTIVE)
  SCOPE(LOCAL)
  MEMBER(DB1D)

```


The output is similar to the following output:

```

ACCELERATOR          MEMB  STATUS  REQUESTS  ACTV  QUED  MAXQ
-----
ACCEL1               DB1D  STARTED    9210     7     5     9
LOCATION=ACCELERATOR1 HEALTHY
DETAIL STATISTICS
  LEVEL = AQT02010
  STATUS = ONLINE
  FAILED QUERY REQUESTS              =      3
  AVERAGE QUEUE WAIT                 =     99
  MAXIMUM QUEUE WAIT                  =    400
  TOTAL NUMBER OF PROCESSORS          =      4
  AVERAGE CPU UTILIZATION ON COORDINATOR NODES = 45.00%
  AVERAGE CPU UTILIZATION ON WORKER NODES   = 40.00%
  NUMBER OF ACTIVE WORKER NODES        =      2
  TOTAL DISK STORAGE AVAILABLE         = 93000 MB
  TOTAL DISK STORAGE IN USE            = 56100 MB
  DISK STORAGE IN USE FOR DATABASE      = 36100 MB
DISPLAY ACCEL REPORT COMPLETE
DSN9022I ) DSNX8CMD '-DISPLAY ACCEL' NORMAL COMPLETION

```

**Related reference:**

 Information about one example of an IBM version 2 accelerator product

| **DSNX830I** *csect-name*

| **Explanation:** This multiline message is the response to the DISPLAY ACCEL command.

| *csect-name*

| The name of the control section that issued the message.

| The report provides the following information about the accelerator server or servers:

| **ACCELERATOR**

| The name of the accelerator server.

| **MEMB** The name of the DB2 data sharing member.

| **STATUS**

| The status of the accelerator server. The status can be any of the following values:

| **STARTED**

| The accelerator server is able to accept requests.

| **STARTEXP**

| The accelerator server was started with the EXPLAINONLY option and is available only for EXPLAIN requests.

**STOPPEND**

The accelerator server is no longer accepting new requests. Active accelerator threads are allowed to complete normally and queued accelerator threads are terminated. The accelerator server was placed in this status by the STOP ACCEL MODE(QUIESCE) command.

**STOPPED**

The accelerator server is not active. New requests for the accelerator are rejected. The accelerator server was placed in this status by the STOP ACCEL command.

**STOPERR**

The accelerator server is not active. The accelerator server was placed in this status by an error condition. New requests for the accelerator are rejected.

**REQUESTS**

The number of query requests that have been processed.

**ACTV** The current number of active, accelerated queries.

**QUED** The current number of queued requests.

**MAXQ** The highest number of queued requests reached since the accelerator was started.

**Detailed report:** If the DISPLAY ACCEL command is issued with the DETAIL option, the following additional information is provided about the accelerator server.

**LOCATION**

The location name of the accelerator server, as specified in table SYSACCEL.SYSACCELERATORS.

*heartbeat-status*

The health status of the accelerator server. The status can be any of the following values:

**HEALTHY**

The accelerator server is accepting and processing requests.

**BUSY** The accelerator server is processing requests, but it is not accepting new requests.

**FLATLINE**

The accelerator server is not responding.

**AUTHFAIL**

DB2 and the accelerator server could not establish the proper credentials in order to communicate with each other. The accelerator server did not accept the value of the ACCELERATORAUTHTOKEN column in the SYSACCELERATORS table for this accelerator.

**DDFFAIL**

A DDF problem occurred during connection to the accelerator server. Messages on the z/OS console that begin with DSNL provide information on how to resolve the problem.

**LEVEL** The product level of the accelerator.

**STATUS**

The status of the accelerator server. The status can be any of the following values:

**CLUSTER IS INITIALIZING**

Data is still being loaded. The accelerator server is not ready for use.

**FULLY OPERATIONAL**

All nodes are operational.

**RECOVERY MODE**

A coordinator node has converted to a worker node as part of a failover. The accelerator server is operational.

**RECOVERING**

The accelerator server is currently recovering from a failing worker node.

**ERROR**

The accelerator server is not operational because of an unrecoverable problem.

**MAINTENANCE**

The accelerator server is in maintenance mode and is not operational.

**MAINTENANCE PENDING**

The accelerator server is preparing to enter maintenance mode. It will complete currently active and queued work but rejects new requests.

**BUSY** The accelerator server is operational, but queues are exceeding the configured limit. The accelerator is rejects new requests until the number of queues is again below the limit.

**FAILED QUERY REQUESTS**

The number of failed requests for the accelerator.

**AVERAGE QUEUE WAIT**

The length of the average queue wait time within the accelerator, measured in milliseconds.

**MAXIMUM QUEUE WAIT**

The length of the longest queue wait time within the accelerator, measured in milliseconds.

**TOTAL NUMBER OF PROCESSORS**

The number of processors within the accelerator. In the case of a multi-core processor, each core is counted as a separate processor.

**AVERAGE CPU UTILIZATION ON COORDINATOR NODES**

Average processor utilization for coordinator node in this accelerator server. This value is a number in the range from 0.00 to 100.00, where 0.00 indicates that all processors are idle and 100.00 means that all processors are fully utilized.

This value does not represent the current state of processor utilization, but rather the average utilization within the last 60 seconds. The returned value is the average, calculated over all coordinator nodes within the accelerator since startup.

**AVERAGE CPU UTILIZATION ON WORKER NODES**

The average processor utilization for worker nodes in this accelerator server. This value is a number in the range from 0.00 to 100.00, where 0.00 indicates that all processors are idle and 100.00 means that all processors are fully utilized.

This value does not represent the current state of processor utilization, but rather the average utilization within the last 60 seconds. The returned value is the average, calculated over all worker nodes within the accelerator since startup.

**NUMBER OF ACTIVE WORKER NODES**

The number of active worker nodes in the accelerator.

**TOTAL DISK STORAGE AVAILABLE**

The total amount of disk storage that is available in the accelerator, measured in megabytes (MB).

**TOTAL DISK STORAGE IN USE**

The percentage of the total disk storage within the accelerator that is being used by the accelerator. This value is a number in the range from 0.00 to 100.00, where 0.00 indicates that none of the disk storage is in use, and 100.00 means that all of the disk storage is in use.

**DISK STORAGE IN USE FOR DATABASE**

The amount of disk storage that is being used by the database within the accelerator, measured in megabytes (MB).

The report terminates with message DSNX859I.

**System action:** Processing continues.

**User response:** No action is required.

**Related reference:**

Chapter 24, “-DISPLAY ACCEL (DB2),” on page 165

 SYSACCEL.SYSACCELERATORS table (DB2 SQL)





---

## 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—————««
```

### 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)
DSNZPARM          2          0,00
CURRENT           2          0,00
=====
ADDR STATUS CORR-ID VOLSER DATASET_NAME
03B0  BUSY  SHEDDEN  A00001 DSNT2AR1.DT25.D04169.T1328583.A0012701
      RECALL 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

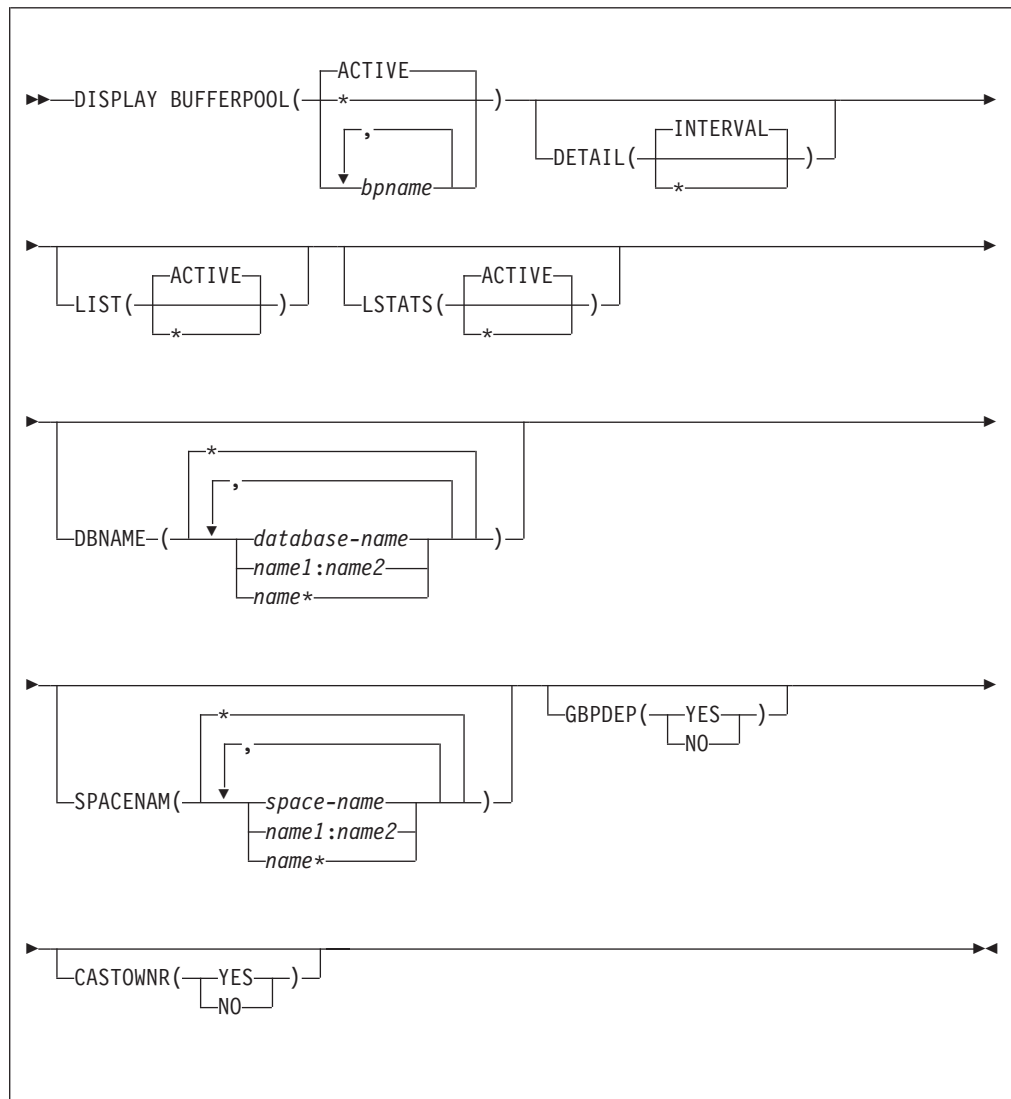
### Authorization

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 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
        PARALLEL SEQUENTIAL  = 50
        ASSISTING PARALLEL SEQT = 0
DSN9022I - 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
        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

```

I

```

DSNB440I - PARALLEL ACTIVITY -
          PARALL REQUEST =      0    DEGRADED PARALL =      0
DSN9022I - 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
          PARALLEL SEQUENTIAL = 50
          ASSISTING PARALLEL SEQT = 0
DSNB455I - SYNCHRONOUS I/O DELAYS -
          AVERAGE DELAY    = 22
          MAXIMUM DELAY     = 35
          TOTAL PAGES       = 23
DSN9022I - DSNB1CMD '-DISPLAY BUFFERPOOL' NORMAL COMPLETION

```

---

**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 (but not necessarily all) 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 report 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 available storage on the system.

|       **NO**      The buffer pool does not use WLM services for automatic buffer pool sizing adjustment.

| **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	= <i>vpseq</i>
DEFERRED WRITE	= <i>dwt</i>
VERTICAL DEFERRED WRT	= <i>dwv1,dwv2</i>
PARALLEL SEQUENTIAL	= <i>vppseqt</i>
ASSISTING PARALLEL SEQT	= <i>vpxpseqt</i>

## DSNB401I

| *vpseq*     The sequential steal threshold for the virtual pool, expressed as a percentage of the total buffer pool size.

| *dwt*       The free buffer deferred write threshold for the buffer pool, expressed as a percentage of the total buffer pool size.

| *dww1*      The vertical deferred write threshold for the buffer pool, expressed as a percentage of the total buffer pool size.

| *dww2*      The vertical deferred write threshold for the buffer pool, expressed as an absolute number of buffers. *dww2* is used to determine the threshold only if *dww1* is 0 and *dww2* is non-zero. Otherwise *dww1* 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. *vpxpseqt* allows you 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:

| 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 for only the duration of an I/O operation.

#### | **PENDING** = *pending-pgfix*

|         The value of the PGFIX attribute that will be applied for the next allocation of the virtual buffer pool.

|         **YES**     The buffer pool will be fixed in real storage for the long term.

|         **NO**      The buffer pool will be fixed in real storage for only the duration of 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 the time the buffer pool  
 |       was first activated, if no previous incremental display exists.  
 |       The format is *hh:mm:ss month dd, yyyy*  
 |       *hh:mm:ss*  
 |           hour:minutes:seconds  
 |       *month*   An alphanumeric abbreviation for the month. For example, a value of OCT means October.  
 |       *dd*       Day of the month  
 |       *yyyy*     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*  
 |           hour:minutes:seconds  
 |       *month*   An alphanumeric abbreviation for the month. For example, a value of OCT means October.  
 |       *dd*       Day of the month  
 |       *yyyy*     Year

| **DSNB411I: Page read statistics:**  
 | When you specify the DETAIL option, the output includes the page read statistics for the buffer pool.

| **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 12, “-ALTER BUFFERPOOL (DB2),” on page 53

| **DSNB412I: Sequential prefetch statistics:**

## DSNB401I

| 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 = *pfp***  
|       The number of pages read because of sequential prefetch.

| **Related information:**  
|       Sequential prefetch (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 = *pfp***  
|       The number of pages read because of list prefetch.

| **Related information:**  
|       List prefetch (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 = *pfp***  
|       The number of pages read because of dynamic prefetch.

| **Related information:**  
|       Dynamic prefetch (DB2 Performance)

### | **DSNB415I: Prefetch statistics:**

| When you specify the DETAIL option, the output includes the prefetch statistics for the buffer pool.

| **NO BUFFER = *pfd***  
|       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:**  
|       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.

| **SYS PAGES WRITTEN** = *pages-written*  
 |       The number of pages that are written to disk.

| **ASYNC 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 the number of 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.

## DSNB401I

| **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 incremental 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 (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 incremental 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.  
| *tio*     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 can still 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 report 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** Two-character indicator of object type: table space (TS) or index space (IX).

| **GBP DEP**

| An indicator of group buffer pool (GBP) dependency. Expected values:

| **Y** The page set or partition is GBP-dependent.

| **N** The page set or partition is not GBP-dependent.

| **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. Expected 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. Expected 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.

## DSNB401I

	<b>S</b>	R/O interest. Other members might be reading the page set or partition. The page set or partition is not GBP-dependent.
	<b>SIX</b>	R/W interest. Other members might be reading the page set or partition. The page set or partition is GBP-dependent.
	<b>NSU</b>	R/W interest. The page set or partition is GBP-dependent.
	<b>X</b>	R/W interest. No other members are accessing the page set or partition. The page set or partition is not GBP-dependent.
	<b>US</b>	A temporary state that can be held by a restarting DB2 when “waiting for retained locks” is enabled.
	<i>number</i>	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 report 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** Two-character indicator of object type: table space (TS) or index space (IX).



| **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 report 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: table space (TABLE SPACE) or index space (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.

| **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 for concurrency control (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 report is too long, the report ends with message DSNB499I. This message indicates that the command was unable to obtain storage for additional messages. This situation occurs for only a long display request, such as a detail display for many buffer pools.

| The report is truncated.

| Reissue the DISPLAY BUFFERPOOL command, and specify a smaller number of buffer pools.

| **System action:** Processing continues.

| **Related tasks:**

|  Monitoring and tuning buffer pools using online commands (DB2 Performance)

| **Related reference:**

| Chapter 26, “-DISPLAY BUFFERPOOL (DB2),” on page 173

| Chapter 12, “-ALTER BUFFERPOOL (DB2),” on page 53

---

## 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:

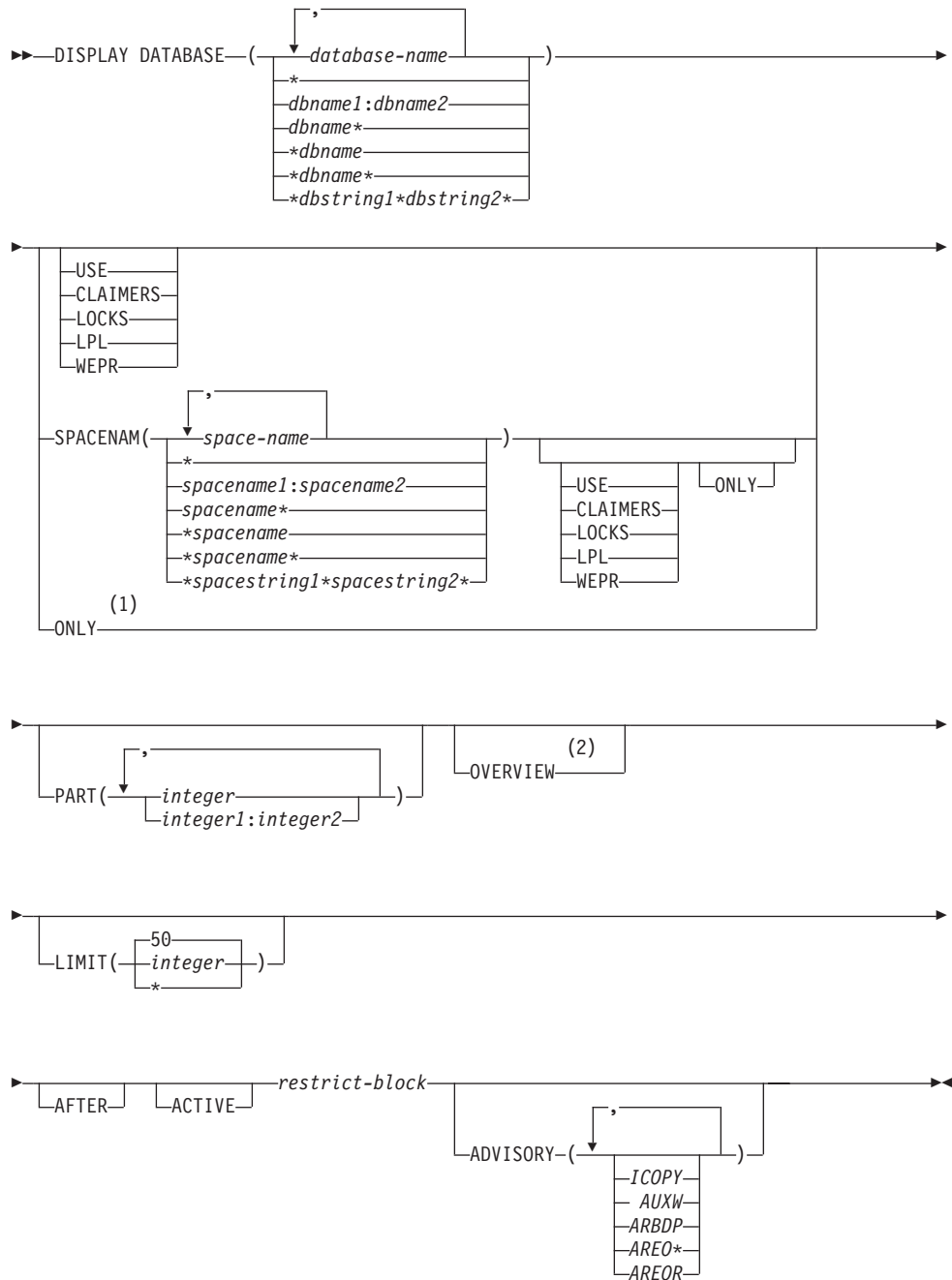
- DISPLAYDB privilege
- DISPLAY privilege
- DBMAINT authority
- DBCTRL authority
- DBADM authority
- System DBADM authority
- SYSOPR authority

- 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.

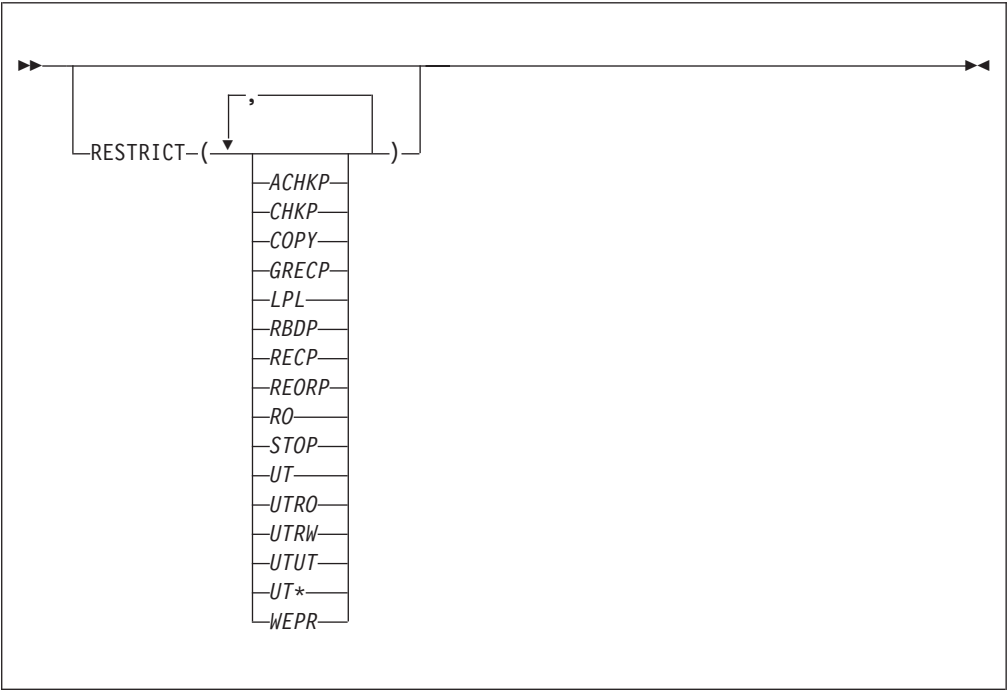
## Syntax



### Notes:

- 1 If you specify the `ONLY` option without the `SPACENAM()` keyword, only the `LIMIT`, `AFTER`, and `RESTRICT` keywords apply.
- 2 The `OVERVIEW` keyword cannot be specified with any other keywords except `SPACENAM`, `LIMIT`, and `AFTER`.

**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 14. 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 15. 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 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.

**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.

**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` will indicate what the current base and clone data set instance numbers are.

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.

See the following Examples section for examples of how to display cloned table information.

## Output

**Message DSNT392I status information:** The status codes that are displayed by the DISPLAY DATABASE command and their respective descriptions are as follows:

### ARBDP

Indicates that the index should be rebuilt to improve performance and allows the DB2 subsystem to pick this index for index-only access.

### AREO\*

Indicates that the table space, index, or partition identified should be reorganized for optimal performance.

### AREOR

Indicates that the table space or index identified should be reorganized to materialize pending definition changes.

### ACHKP

Indicates an error in the LOB column of the base table space. The base table space has the auxiliary CHECK-pending restrictive status.

### AREST

Indicates that an object (a table space, index space, or a physical partition of a table space or index space) is in an advisory RESTART-pending status. If backout activity against the object is not already underway, initiate it either by issuing the RECOVER POSTPONED command, or by recycling the system with the system parameter LBACKOUT=AUTO.

### AUXW

Either the base table space or XML table space is in the auxiliary warning advisory status, indicating an error in the LOB column, or the LOB table space is in the auxiliary warning advisory status, indicating an invalid LOB. This message can also indicate an error in a base table space with an XML column or in an XML table space.

**CHKP** The object (a table space, a partition within a table space, or an index) is in the CHECK-pending status.

**COPY** The object (a table space or a partition within a table space) is in the COPY-pending status. An image copy is required for this object.

### DBETE

DBETE status indicates one of the following object errors:

- The DBET status for the object (a table space, a partition within a table space, an index space, an index partition, or a logical index partition) is in error. That error occurs when the DBET states are modified during log apply or must-complete processing, which prohibits DB2 from successfully updating the DBET states.

- DB2 encountered an unexpected page set access error during restart. This error is not a DBET exception-state entry error.

When an object is in DBETE, RECP, RBDP, and PSRBD status, the page set type is unknown. In that case, TYPE UN (unknown TYPE) is issued because the object might be in a table space or an index space.

When an entire space has type UN and DBETE and RECP status, the entire space needs to be recovered using RECOVER, LOAD REPLACE or REBUILD. Depending on the DISPLAY command syntax, objects type UN and DBETE and RECP status might be displayed in various formats.

<b>GRECP</b>	The object is GBP-dependent and a group buffer pool RECOVER is pending.
<b>ICOPY</b>	The object is in the informational COPY-pending advisory status.
<b>LPL</b>	The object has entries in the logical page list.
<b>LSTOP</b>	The logical partition of a nonpartitioning index is stopped.
<b>PSRBD</b>	The entire nonpartitioning index space is in a page set REBUILD-pending status.
<b>RBDP</b>	The physical or logical index partition is in the REBUILD-pending status.
<b>RBDP*</b>	The logical partition of a nonpartitioning index is in the REBUILD-pending status, and the entire index is inaccessible to SQL applications. However, only the logical partition needs to be rebuilt.
<b>RECP</b>	The object (a table space, table space partition, index space, index partition, or logical index partition) is in the RECOVER-pending status.
<b>REFP</b>	The object (a table space, index space, or an index) is in the REFRESH-pending status.
<b>RELDP</b>	The object has a release dependency.
<b>REORP</b>	The data partition is in the REORG-pending status.
<b>RESTP</b>	The table space or index space is in the restart-pending status.
<b>RO</b>	The database, table space, table space partition, index space, or index space partition is started for read-only activity.
<b>RW</b>	The database, table space, table space partition, index space, or index space partition is started for read and write activity.
<b>STOP</b>	The database, table space, table space partition, index space, or index space partition is stopped.
<b>STOPE</b>	The table space or index space was implicitly stopped because there is a problem with the log RBA in a page. Message DSNT500I or DSNT501I is issued when the error is detected, indicating the inconsistency.
<b>STOPP</b>	A stop is pending for the database, table space, table space partition, index space, or index space partition.
<b>UT</b>	The database, table space, table space partition, index space, or index space partition is started for utility processing only.
<b>UTRO</b>	A utility is in process, on the table space, table space partition, index space, or index space partition, that allows only RO access. If the utility was canceled before the object was drained, the object can allow SQL access because the object was not altered by the utility.

## UTRW

A utility is in process, on the table space, table space partition, index space, or index space partition, that allows RW access.

## UTUT

A utility is in process, on the table space, table space partition, index space, or index space partition, that allows only UT access. If the utility was canceled before the object was drained, the object can allow SQL access because the object was not altered by the utility.

**WEPR** Displays write error page range information.

## 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
TBS33     TS    0002 RW          LSS001   DSN2SQL   SYSADM
TBS33     TS    0003 RW          LSS001   DSN2SQL   SYSADM
TBS33     TS    0004 RW          LSS001   DSN2SQL   SYSADM

***** 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
TBS33     TS    0002 RW
TBS33     TS    0003 RW
TBS33     TS    0004 RW          LSS004   DSN2SQL   H(IS,S,C)
TBS33     TS    0004 RW          LSS005   DSN2SQL   H(IS,S,C)

***** 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
TBS33     TS    0002 RW
TBS33     TS    0003 RW
TBS33     TS    0004 RW              LSS001  DSN2SQL  (RR,C)
TBS33     TS    0004 RW              LSS001  DSN2SQL  (WR,C)

***** 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			H-S,PP,I
-			MEMBER NAME DB1G			
TSPART	TS	0001	RO	LSS002	DSN2SQL	H-IS,P,C
-			MEMBER NAME DB2G			
TSPART	TS	0001	RO			H-S,PP,I
-			MEMBER NAME DB2G			
TSPART	TS	0002	RW	LSS001	DSN2SQL	H-IS,P,C
-			MEMBER NAME DB1G			
TSPART	TS	0002	RW			H-S,PP,I
-			MEMBER NAME DB1G			
TSPART	TS	0003	RW	LSS002	DSN2SQL	H-IS,P,C
-			MEMBER NAME DB2G			
TSPART	TS	0003	RW			H-S,PP,I
-			MEMBER NAME DB2G			
TSPART	TS	0004	RW			

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			

**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
-----
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
IAIQUQ01 IX              RW,ICOPY
IAIQUQ02 IX              RW,ICOPY
IAIQUQ03 IX              RW,ICOPY
IAIQUQ04 IX              RW,ICOPY
IPIQUQ01 IX      0001 RW,ICOPY
      -THRU      0004
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
IX486A    IX      L0004
IX486B    IX      0004
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
IX486X  IX   L0001 RW
IX486X  IX   L0002 LSTOP
- THRU      L0003
IX486X  IX   L0004 LSTOP
- THRU      L0010
IX486Y  IX   0001 RW
- THRU      0010
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
INDEX2    IX      L*   RW
INDEX3    IX      D0001 RW
-THRU     D0002
INDEX4    IX      L0001 RECP
INDEX4    IX      L0002 RW

```

#### 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

```

### 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      0002 RW,AREO*
DB2TSP    TS      0004 RW
DB2TSP    TS      0001 RW,AREO*
DB2TSP    TS      0005 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
TPIQUQ01 TS      002 RW,AUXW
TPIQUQ01 TS      003 RW,AUXW
TPIQUQ01 TS      004 RW,ICOPY
```

### 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
CHKOUT2	TSB1	002	RW
CHKOUT2	TSB1	003	RW
XDEPT1	IXB1	001	RECP
XDEPT1	IXB1	002	RW
XDEPT1	IXB1	003	RW
CHKOUT2	TSC2	001	RO
CHKOUT2	TSC2	002	RW
CHKOUT2	TSC2	003	RW
XDEPT1	IXC2	001	ICOPY
XDEPT1	IXC2	002	RW
XDEPT1	IXC2	003	RW
CHKOUT3	TS	001	RO
CHKOUT3	TS	002	RO
CHKOUT3	TS	003	RO
XDEPT3	IX	001	RW
XDEPT3	IX	002	RW
XDEPT3	IX	003	RO

**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
- 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 DDF [ALIAS(alias-name)] [DETAIL] ►►
```

### 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

The DISPLAY DDF command displays the following output:

#### **STATUS**

The operational status of DDF.

**LOCATION**

The location name of DDF.

**LUNAME**

The fully qualified LUNAME of DDF.

**GENERICLU**

The fully qualified generic LUNAME of DDF.

**IPADDR**

The IP address of DDF.

**TCPPORT**

The SQL listener port used by DDF.

**RESPORT**

The resync listener port used by DDF.

**DOMAIN**

The SQL and resync domains used by DDF.

**SECPORT**

The TCP/IP SQL listener port used by DDF to accept inbound SSL connections.

**ALIAS**

Information related to location alias names that are defined.

**Options:**

A list of the option values that are currently in effect for DDF.

**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 TCPPORT=446  SECPORT=0    RESPORT=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 TCPPORT=446  SECPORT=447  RESPORT=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 TCPPORT=446      SECPORT=447      RESPORT=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
DSNL091I MCONQN=      -1  MCONQW=      0
DSNL092I ADBAT=      0  QUEDBAT=      0  INADBAT=      0  CONQUED=      0
DSNL093I DSCDBAT=      0  INACONN=      0
DSNL094I WLMHEALTH=100  CLSDCONQN=      0  CLSDCONQW=      1
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 TCPPORT=446      SECPORT=0        RESPORT=5001 IPNAME=-NONE
DSNL085I IPADDR=:9.30.115.135
DSNL085I IPADDR=2002:91E:610:1::5
DSNL086I SQL      DOMAIN=v7ec135.svl.ibm.com
DSNL087I ALIAS           PORT      SECPORT 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 (alias1) 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 (alias1) 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


```

*Example:* The following command is used to display a DDF alias detail report:

```
-DISPLAY DDF ALIAS(alias01) DETAIL
```

This command produces output similar to the following output:

**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

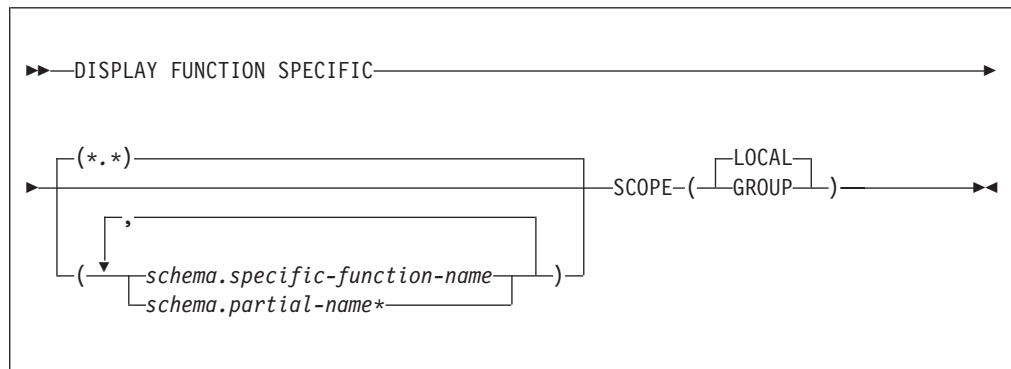
If you specify DISPLAY FUNCTION SPECIFIC \*.\* or *schema.partial-name \**, you must use a privilege set of the process that includes one of the following authorities:

- 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

This command displays one line of output for each function that a DB2 application has accessed.

Information returned by this command reflects a dynamic status. By the time DB2 displays the information, the status might have changed.

**Sample output:** The DISPLAY FUNCTION SPECIFIC command generates the following output:

```
DSNX975I = DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT FOLLOWS -
```

```
----- SCHEMA=PAYROLL
FUNCTION      STATUS ACTIVE  QUED  MAXQ  TIME  FAIL  WLM_ENV
APPL1         STARTED    1    0    0    0    0  PAYROLL
APPL2         STARTED    1    0    0    0    1  PAYROLL
APPL3         STARTED    0    1    2    0    0  PAYROLL
APPL5         STOPREJ    0    0    0    0    0  SANDBOX
APPL6         STOPABN    0    0    0    0    1  PAYROLL
FUNC1         STOPQUE    0    0    0    0    0  SANDBOX
```

```
DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT COMPLETE
DSN9022I = DSNX9COM '-DISPLAY FUNC' NORMAL COMPLETION
```

**DISPLAY FUNCTION SPECIFIC command output:** The DISPLAY FUNCTION SPECIFIC command displays the following output:

### FUNCTION

The specific name of the function.

### STATUS

The status of the function. The possible values are:

#### STARTED

Requests for the function can be processed.

#### STOPQUE

Requests are queued.

#### STOPREJ

Requests are rejected.

#### STOPABN

Requests are rejected because of abnormal termination.

### ACTIVE

The number of threads that are currently running the function.

### QUED

The number of threads that are waiting for the function to be scheduled.

### MAXQ

The maximum number of threads that have waited concurrently for the function to be scheduled since the DB2 subsystem was started.

## TIMEOUT

The number of times an SQL statement timed out while waiting for a request for the function to be scheduled.

**FAIL** The number of times a procedure has failed. DB2 resets this value to 0 each time you run the START FUNCTION command.

## WLM\_ENV

The WLM environment where the function runs.

Message DSNX971I lists a range of functions that are stopped because a STOP FUNCTION SPECIFIC command included a partial name with the pattern-matching character (\*). See the topic Chapter 94, “-STOP FUNCTION SPECIFIC (DB2),” on page 535 for more information.

## 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

```
»»—DISPLAY GROUP—[DETAIL]—««
```

### 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 DSN7106I 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 the same DB2 version:

```
-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      DB2 SYSTEM  IRLM
MEMBER   ID  SUBSYS  CMDPREF  STATUS  LVL NAME  SUBSYS  IRLMPROC
-----
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 the same DB2 version:

```
-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      DB2 SYSTEM  IRLM
MEMBER   ID  SUBSYS  CMDPREF  STATUS  LVL NAME  SUBSYS  IRLMPROC
-----
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:

```
-DB1A DISPLAY GROUP
```

The output is similar to the following output:

```

DSN7100I -DB1A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(.....) CATALOG LEVEL(101) MODE(N )
        PROTOCOL LEVEL(...) GROUP ATTACH NAME(....)
-----
DB2      DB2 SYSTEM  IRLM
MEMBER   ID  SUBSYS  CMDPREF  STATUS  LVL NAME  SUBSYS  IRLMPROC
-----
.....   0  DB1A    -DB1A    ACTIVE  101 MVSA  DJ1A    DB1AIRLM

```

```
-----
*** END DISPLAY OF GROUP(DSNDB10)
DSN9022I -DB1A DSN7GCMDB10 '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 @ DSN7GCMDB10 488
*** BEGIN DISPLAY OF GROUP(DSNDB10 ) 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
-----
VA1A      NO              NO
-----
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(DSNDB10 )
DSN9022I @ DSN7GCMDB10 'DISPLAY GROUP ' NORMAL COMPLETION
```

Output

Message DSN7100I indicates the beginning of the output of the command.

**DSN7100I** *csect-name*

**Explanation:** This message displays multiple lines of information in response to a DISPLAY GROUP command.

*csect-name*

The name of the control section that issued the message.

The output from the command consists of one or more (but not necessarily all) of the following sections, in the indicated order:

- Report heading
- Data sharing members
- Member support for Sysplex query parallelism
- New-function mode enablement
- Subgroup attachment
- Shared communications area
- LOCK1 structure
- SPT01 table space information

| • End of report

| **Report heading:** The name of the control section is followed by heading information for the report:

| **GROUP** (*text*)

|     The name of the data sharing group.

| **CATALOG LEVEL** (*text*)

|     A string of three numeric characters that indicates the highest release with which any DB2 subsystem in the group has been started:

|     • Version

|     • Release

|     • Modification level

| **MODE** (*text*)

|     A two- or three-character indicator of the current mode of the system:

|     **Cn**     Conversion mode. All members in the group are on the same DB2 release.

|     **Cn\***    Conversion mode, but the system detects that it has been to a higher level.

|     **En**     Enabling-new-function mode.

|     **En\***    Enabling-new-function mode, but the system detects that it has been to a higher level.

|     **N**     New-function mode.

|     **X**     Coexistence mode. Either members in the data sharing group are on different DB2 releases or a member has performed a fallback from a higher release.

| **PROTOCOL LEVEL** (*num*)

|     A single numeric character that indicates the current locking protocol used in the data sharing group.

| **GROUP ATTACH NAME** (*text*)

|     The group attach name for the data sharing group.

| **Data sharing members:** The report heading is followed by a section that reports information about the data sharing members:

| **DB2 MEMBER** *text*

|     The name of the DB2 data sharing member.

| **ID** *num*

|     The internally assigned ID for each data sharing member.

| **SUBSYS** *text*

|     The subsystem name of the data sharing member.

| **CMDPREF** *text*

|     The command prefix of the data sharing member.

| **STATUS** *text*

|     The state of the data sharing member. Expected values are:

|     **ACTIVE**

|         The member is currently up and running.

|     **AI**     The member is active with additional conditions. Indoubt or postponed abort units of recovery are unresolved, so retained locks are held.

|     **DEACT**

|         The member has been deactivated.

|     **FAILED**

|         The member has terminated abnormally.

|     **QUIESCED**

|         The member is down normally.

|     **QI**     The member is down with indoubt units of recovery.

|     **QR**     The member is down with resynchronization responsibility.

|       **QC**       The member is down with group buffer pool castout responsibility.

|       **QCR, QIC, QIR, or QICR**

|           A possible combination of the quiesced states that are listed above:

|           • QCR is the equivalent of QC and QR

|           • QIC is the equivalent of QI and QC.

|           • QIR is the equivalent of QI and QR

|           • QICR is the equivalent of QC, QI, and QR

|       **DB2 LVL** *num*

|           A string of three numeric characters that identifies the DB2 product version:

|           • DB2 version

|           • DB2 release

|           • DB2 modification level

|       **SYSTEM NAME** *text*

|           The name of the z/OS system on which the data sharing member runs.

|       **IRLM SUBSYS** *text*

|           The accompanied IRLM's subsystem name for this DB2 member.

|       **IRLMPROC** *text*

|           The accompanied IRLM's startup procedure name.

|       **Member support for Sysplex query parallelism:** If the DISPLAY GROUP command is issued with the DETAIL option, and there are active DB2 members that support Sysplex query parallelism, the following additional information is provided:

|       **DB2 MEMBER** *text*

|           The name of the DB2 data sharing member.

|       **PARALLEL COORDINATOR** *text*

|           An indicator of whether or not this DB2 member can coordinate parallel processing. Expected values are "YES" or "NO".

|       **PARALLEL ASSISTANT** *text*

|           An indicator of whether or not this DB2 member can assist with parallel processing. Expected values are "YES" or "NO".

|       If no members support Sysplex query parallelism, these lines are omitted from the report.

|       **New-function mode enablement:** If the DISPLAY GROUP command is issued with the DETAIL option, and the DB2 catalog mode is indicated as "E" in the report heading, then the following additional information is provided:

|       **TABLE SPACE** *text*

|           The name of the table space.

|       **ENABLED FOR NEW FUNCTION** *text*

|           An indicator of the readiness of the table space for new-function mode. Expected values are:

|           **YES**       Either the table space was created in Unicode or it has been converted to Unicode.

|           **NO**        The table space is not enabled for new-function mode. Table spaces are converted to Unicode during enabling-new-function mode.

|       **Subgroup attachment:** If the DISPLAY GROUP command is issued with the DETAIL option, subgroup attachment information is provided:

|       **GROUP ATTACH** *text*

|           The name of the group attachment group for the data sharing member that is printing the report.

|       **MEMBERS OF SUBGROUP ATTACH** *text*

|           The name of the subgroup attachment group, followed by one or more lines listing the members of the subgroup. Each line contains up to eight members.

|           If more than 105 subgroup attachment groups are found, the following message displays:

MAXIMUM NUMBER OF SUBGROUP ATTACH GROUPS  
DISPLAYED

In this case, no additional groups are reported.

**Shared communications area:** The report provides the following information about the shared communications area (SCA):

**STRUCTURE SIZE** *num*

The size of the shared communications area (SCA), measured in kilobytes (KB).

**STATUS** *tx*

A two-character indicator of the status of the shared communications area. Possible values are:

<b>AC</b>	SCA is active.
<b>CC</b>	SCA rebuild connect function completed.
<b>CS</b>	SCA rebuild connect function started.
<b>DS</b>	SCA loss of connectivity, delayed action started.
<b>DC</b>	SCA loss of connectivity, delayed action completed.
<b>EC</b>	SCA rebuild stop complete function completed.
<b>ES</b>	SCA rebuild stop complete function started.
<b>IN</b>	SCA is in initialization phase.
<b>LC</b>	SCA rebuild cleanup function completed.
<b>LS</b>	SCA rebuild cleanup function started.
<b>OC</b>	SCA rebuild connect complete function completed.
<b>OS</b>	SCA rebuild connect complete function started.
<b>PC</b>	SCA rebuild process complete function completed.
<b>PS</b>	SCA rebuild process complete function started.
<b>QC</b>	SCA rebuild quiesce function completed.
<b>QS</b>	SCA rebuild quiesce function started.
<b>RC</b>	SCA rebuild resource manager invocation completed.
<b>RS</b>	SCA rebuild resource manager invocation started.
<b>SC</b>	SCA rebuild stop function completed.
<b>SS</b>	SCA rebuild stop function started.
<b>TE</b>	SCA is in termination phase.
<b>UC</b>	SCA rebuild resource manager invocation completed.
<b>US</b>	SCA rebuild resource manager invocation started.
<b>VC</b>	SCA volatility change function completed.
<b>VS</b>	SCA volatility change function started.

**SCA IN USE** *num*

Either the percentage of the number of entries that are in use in the SCA or the percentage of the number of elements that are in use in the SCA, whichever is bigger. If the SCA structure is very large or if only a small number of SCA entries are in use, the percentage that is in use might be less than 1%. In this case, the percentage is indicated as:

SCA IN USE: < *num* %

**LOCK1 structure:** The report provides the following information about the LOCK1 structure:

**STRUCTURE SIZE** *num*

The size of the LOCK1 coupling facility structure, measured in kilobytes (KB).

## DSN7100I

- | **LOCK ENTRIES** *num*  
|       The maximum number of lock table entries in the LOCK1 coupling facility structure.
- | **LIST ENTRIES** *num*  
|       The maximum number of modify lock list entries in the LOCK1 coupling facility structure.
- | **LIST ENTRIES IN USE** *num*  
|       The number of modify lock list entries in the LOCK1 coupling facility structure that are currently in use.  
|       These entries are sometimes referred to as "record data".
- | **SPT01 table space information:** The report contains information about the size of the inline length of the SPT01  
| column SPTSEC\_DATA:
- | **SPT01 INLINE LENGTH** *num*  
|       The current size of the inline length of the SPTSEC\_DATA column.
- | **End of report:** The report normally ends with the following message:  
| \*\*\* END DISPLAY OF GROUP(*text*)
- | **GROUP**  
|       The name of the group that the preceding lines of the report refer to.
- | If the output exceeds the maximum length, the following message marks the end of the report instead:  
| DISPLAY TERMINATED DUE TO INSUFFICIENT MESSAGE  
|       SPACE.
- | **System action:** Processing continues.
- | **User response:** No action is required.
- | **Related reference:**  
| Chapter 30, "-DISPLAY GROUP (DB2)," on page 219



---

## 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

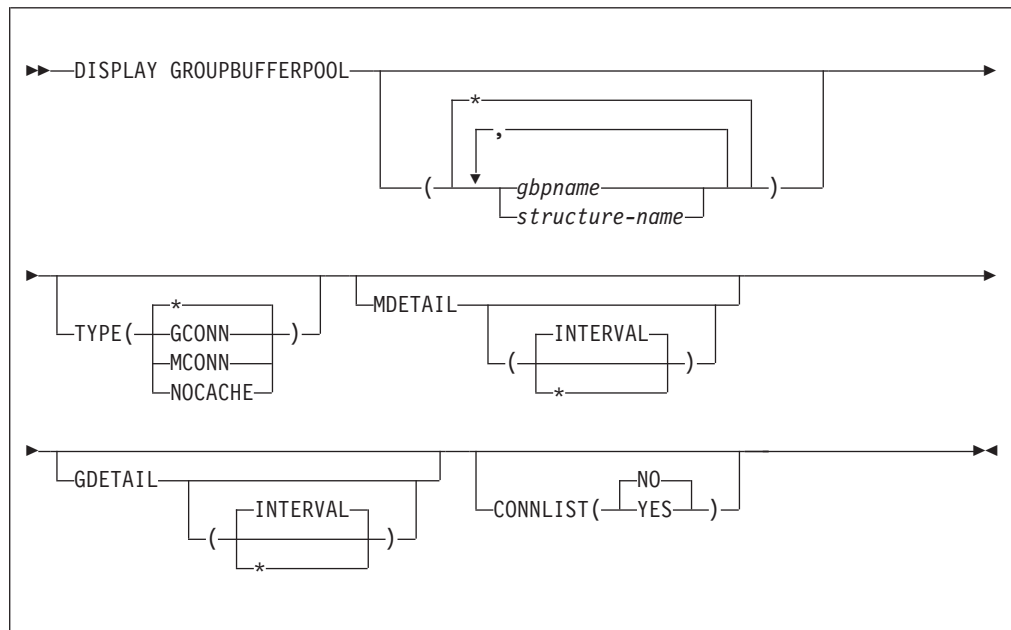
### Authorization

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**

The DISPLAY GROUPBUFFERPOOL command can produce the following the three report types:

- A summary report
- A group detail report
- A member detail report

**Summary report**

You can display summary information about group buffer pools. The report indicates whether this DB2 subsystem is actively connected to the group buffer pools for which you requested information. The summary report also shows the following information:

*Group buffer pool characteristics:*

- Threshold values
- Directory-to-data entry ratio (both pending and current)
- Checkpoint interval
- Recovery status (whether damage assessment is pending)

***CFRM policy information about the group buffer pool:***

- The allocation value specified in the CFRM policy and whether the group buffer pool is currently allocated in the coupling facility.
- The actual allocated size (which can be different from that specified in the CFRM policy) and volatility status. DB2 requests nonvolatile storage; however, it can allocate in a volatile structure.
- The actual number of directory entries, data pages, and connections to the group buffer pool.

The summary report contains additional information as follows:

**AUTOMATIC RECOVERY**

Indicates whether automatic recovery is allowed for this group buffer pool.

**DUPLEX**

Indicates the current duplexing option for the group buffer pool that is specified in the active CFRM policy.

**REBUILD STATUS**

Indicates whether a rebuild is in progress for this group buffer pool. If so, the phase of the rebuild is indicated as either QUIESCE, CONNECT, or CLEANUP. If the rebuild is in the process of stopping, the status indicates STOPPING.

**DUPLEXING STATUS**

Indicates the current state of the group buffer pool with respect to duplexing.

**CFNAME**

Indicates the name of the coupling facility in which the group buffer pool is allocated. If the group buffer pool is duplexed, this is the coupling facility name associated with the primary group buffer pool.

**CFLEVEL**

Indicates the level of the coupling facility in which the group buffer pool is allocated. If the group buffer pool is duplexed, this is the coupling facility level associated with the primary group buffer pool.

Both the operational and actual levels of the coupling facility are shown. The operational level indicates the capabilities of the coupling facility from the DB2 subsystem perspective. The actual level is the level as displayed by the z/OS D CF command.

**LAST GROUP BUFFER POOL CHECKPOINT**

Indicates the date and time of the last group buffer pool checkpoint, the LRSN that was recorded at that checkpoint, and the member name of the group buffer pool structure owner.

**Group detail report**

The group detail report shows detailed statistical information reflecting the activity of the entire group for the specified group buffer pools. This statistical information

is helpful in tuning the size and other characteristics of group buffer pools. The group detail report includes the same information as the summary report in addition to the following information:

#### **READS**

Information about reads.

This is a detailed accounting of the number of reads against the group buffer pool, including the following:

- The number of reads where data was returned.
- The number of reads where data was not returned, broken down to include more detailed information about whether the page was cached in the coupling facility or not, and whether directory entries needed to be created to fulfill requests for data.

#### **WRITES**

Information about writes.

This includes the number of writes for clean pages and changed pages, and how many writes failed because there was not enough storage in the group buffer pool.

#### **CHANGED PAGES SNAPSHOT VALUE**

The number of changed pages currently in the group buffer pool (a snapshot value).

#### **RECLAIMS**

The number of reclaims of directory entries and data pages.

#### **CASTOUTS**

The number of castouts.

#### **CROSS INVALIDATIONS**

The number of cross-invalidations that occurred because of directory reclaims and because of writes.

#### **DUPLEXING STATISTICS FOR GBP *n***

This section of output indicates detailed duplexing statistics as follows:

##### **CHANGED PAGES**

Indicates the number of changed pages that are written to the secondary group buffer pool. If the group buffer pool has been duplexed for the entire reporting interval, this number approximates the CHANGED PAGES counter that is reported in message DSNB786I for the primary group buffer pool. The counts might not be exactly the same, due to timing periods for gathering the counter information for display or previous transaction failures that might have occurred.

##### **FAILED DUE TO LACK OF STORAGE**

Indicates the number of writes to the secondary group buffer pool that failed due to a lack of storage.

##### **CHANGED PAGES SNAPSHOT VALUE**

Indicates the number of changed pages that are currently cached in the secondary group buffer pool. This number approximates the CHANGED PAGES SNAPSHOT VALUE for the primary group buffer pool, but is probably not identical due to the asynchronous nature of gathering statistics for the two different coupling facility structures.

## Member detail report

The member detail report includes the summary report and additional information about how a particular member's system is responding to the current environment. It categorizes reads and writes as synchronous or asynchronous. A large number of synchronous reads or writes can indicate that you need to tune your group buffer pools.

### SYNCHRONOUS WRITES

Lists local synchronous write and cross-invalidation statistics for the group buffer pool.

#### CHANGED PAGES

Number of changed pages that were written synchronously, under a user's execution unit, to the coupling facility.

#### CLEAN PAGES

Number of clean pages that were written synchronously, under a user's execution unit, to the coupling facility

### ASYNCHRONOUS WRITES

Lists local asynchronous write and cross-invalidation statistics for the group buffer pool.

#### CHANGED PAGES

Number of changed pages that were written asynchronously, under a system execution unit, to the coupling facility

#### CLEAN PAGES

Number of clean pages that were written asynchronously, under a system execution unit, to the coupling facility.

#### FAILED DUE TO LACK OF STORAGE

Coupling facility write requests that were not completed because of a lack of storage in the group buffer pool.

### GBP CHECKPOINTS TRIGGERED

The number of checkpoints that occurred for this group buffer pool.

### PARTICIPATION IN REBUILD

The number of times this member participated in a rebuild for this group buffer pool.

### CASTOUTS

This section of output indicates detailed statistics for castout processing as follows:

#### PAGES CAST OUT

Indicates how many data pages were cast out of the group buffer pool by this member.

#### UNLOCK CASTOUT

The number of times that DB2 issued an unlock request to the coupling facility for castout I/Os that completed. As pages are cast out to disk, they are "locked for castout" in the coupling facility. The castout lock ensures that only one system is doing castout for a given page.

DB2 usually includes multiple pages in the write I/O request to disk for castout. Therefore, the UNLOCK CASTOUT counter should always be less than or equal to the value of the PAGES CASTOUT counter; it should be significantly less if multiple pages

are written per I/O. For example, if there are four pages written per castout write I/O on average, PAGES CASTOUT should be four times larger than UNLOCK CASTOUT.

#### **READ CASTOUT CLASS**

Number of requests made to the group buffer pool to determine which pages belonging to a given page set or partition are cached in the group buffer pool as changed pages and thus need to be cast out.

READ CASTOUT CLASS is issued by the page set or partition castout owner, and it is also issued by the group buffer pool structure owner when the GBPOOLT threshold has been reached.

#### **READ CASTOUT STATISTICS**

The number of requests that are issued by the group buffer pool structure owner when the GBPOOLT threshold is reached. This determines which castout classes have changed pages. Generally READ CASTOUT STATISTICS is issued only once or twice for each occurrence of the GBPOOLT threshold.

#### **READ DIRECTORY INFO**

The number of requests to read the directory entries of all changed pages in the group buffer pool. The group buffer pool structure owner issues these requests at group buffer pool checkpoints. The purpose of the request is to determine the oldest recovery LRSN to use in case the group buffer pool fails. This recovery LRSN is displayed in message DSNB798I.

The request to read directory information might be issued several times for each group buffer pool checkpoint. If you see an abnormally high number here, it might be that the requests are being cut short by the model-dependent timeout criteria of the coupling facility. To help alleviate this problem, upgrade those coupling facilities to CFLEVEL=2 or above.

#### **OTHER INTERACTIONS**

This section of the output lists details of other interactions that this DB2 has with this group buffer pool.

##### **REGISTER PAGE**

The number of times that DB2 registered interest to the group buffer pool for a single page. These are register-only requests, meaning that DB2 is not requesting that any data be returned for the page because no data is cached in the group buffer pool for this page. The REGISTER PAGE request is made only to create a directory entry for the page for cross-invalidation when downgrading the P-lock on a page set or partition from S mode to IS mode, or from SIX mode to IX mode.

##### **UNREGISTER PAGE**

The number of times that DB2 reversed registered interest from the group buffer pool for a single page. This is generally done as DB2 uses pages from the local buffer pool that belong to partitions or page sets that are group buffer pool dependent.

##### **DELETE NAME**

The number of times that DB2 issued a request to the group buffer pool to delete directory and data entries that were associated with a given page set or partition. DB2 issues this request:

- When it converts a page set or partition from group buffer pool dependent to non group buffer pool dependent.
- When the first DB2 member opens the object for GBPCACHE ALL objects.

#### **READ STORAGE STATS**

The number of times that DB2 requested statistics information from the group buffer pool. This number should be relatively low. It is issued once per group buffer pool checkpoint by the group buffer pool structure owner. It is also issued for DISPLAY GROUPBUFFERPOOL GDETAIL requests and to record IFCID 0254.

#### **DUPLEXING STATISTICS FOR GBP0-SEC**

This section of the output lists details of other interactions that this DB2 has with this group buffer pool.

#### **CHANGED PAGES**

Indicates the number of changed pages written to the secondary group buffer pool. This number approximates the sum of the synchronous writes of changed pages to the primary group buffer pool and the asynchronous writes of changed pages to the primary group buffer pool. The counts might not be exactly the same, due to timing periods for gathering the counter information for display or previous transaction failures that might have occurred.

#### **FAILED DUE TO LACK OF STORAGE**

Indicates the number of writes to the secondary group buffer pool that failed due to a lack of storage.

#### **COMPLETION CHECKS SUSPENDED**

Indicates the number of times DB2 checked for the completion of the write of a changed page to the secondary group buffer pool, but the write had not yet completed; DB2 suspends the execution unit until the write to the secondary group buffer pool completes.

#### **DELETE NAME LIST**

Indicates the number of DELETE NAME LIST requests to delete a set of pages from the secondary group buffer pool that have just been cast out to disk from the primary group buffer pool.

#### **READ CASTOUT STATISTICS**

Indicates the number of READ CASTOUT STATISTICS requests to check for orphaned data entries in the secondary group buffer pool. The DB2 member that is the group buffer pool structure owner periodically issues these requests to determine whether garbage collection is necessary.

#### **DELETE NAME**

Indicates the number of DELETE NAME requests to delete orphaned data entries from the secondary group buffer pool. The DB2 member that is the group buffer pool structure owner issues these requests if it determines that garbage collection is necessary.

## **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 CFPM 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
              OPERATIONAL CFLEVEL                  = 5
              ACTUAL CFLEVEL                      = 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
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 CFPM 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
              OPERATIONAL CFLEVEL             = 5
              ACTUAL CFLEVEL                  = 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
              OPERATIONAL CFLEVEL             = 5
              ACTUAL CFLEVEL                  = 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
	OPERATIONAL CFLEVEL	= 5
	ACTUAL CFLEVEL	= 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	
	PAGES RETRIEVED	= 0
	FAILED READS DUE TO LACK OF STORAGE	
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	
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
                ASYNCHRONOUS 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

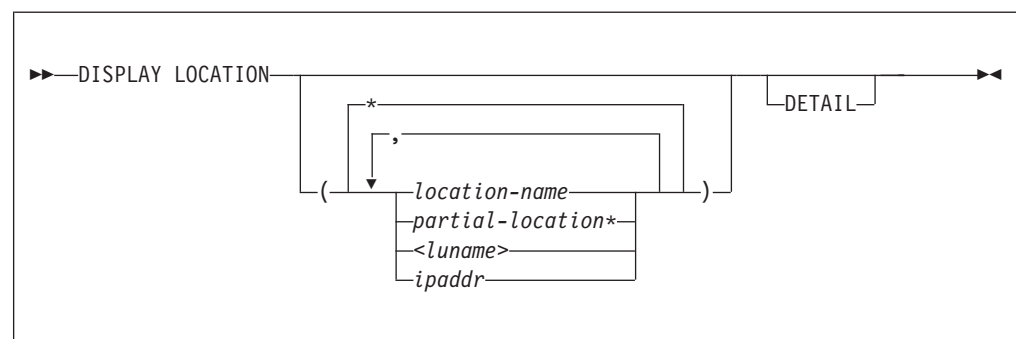
### Authorization

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 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
| LUND1
| LUND2
| PRDID      T ATT CONNS
| DSN10015 R      1
| 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:

## DSNL200I

```
|          DSNL200I  @ DISPLAY LOCATION REPORT FOLLOWS-
|          LOCATION                                PRDID    T ATT CONNS
|          ::FFFF:9.30.115.135                    DSN10015 R          1
|                                                    TRS      1
|          ::FFFF:9.30.115.135                    DSN10015 S          1
|                                                    TRS      1
|          DISPLAY LOCATION REPORT COMPLETE
```

---

### DSNL200I DISPLAY LOCATION REPORT FOLLOWS-

**Basic report:** This message displays multiple lines of information in response to a DISPLAY LOCATION command. The report heading is followed by a line that identifies the values that follow in one or more lines that report information about connections with remote locations:

#### LOCATION

The identifier for the remote location.

- For TCP/IP requester connections, the IP address concatenated with the port number of the remote location. For TCP/IP server connections, the IP address of the remote location.
- The LU name of the remote location, for SNA connections.

**PRDID** An alphanumeric string that identifies the product, in the form *pppvrrm*:

*ppp* A three-character code that identifies the product:

**AQT** IBM DB2 Analytics Accelerator for z/OS

**ARI** DB2 for VSE & VM

**DSN** DB2 for z/OS

**JCC** IBM Data Server Driver for JDBC and SQLJ

**QSQ** DB2 for i

**SQL** DB2 for Linux, UNIX, and Windows

*vv* A two-digit number that identifies the product version number, such as 09 or 10.

*rr* A two-digit number that identifies the product release level, such as 01 or 05.

*m* A one-digit number that identifies the product modification level, such as 0 or 1.

**T** The connection type with the remote location. Expected values are:

**R** A requester connection from local subsystem accessing a remote system.

**S** A server connection from remote system accessing local subsystem.

#### ATT and CONNS

The number of connections with the partner location. When the ATT column is blank, the value of CONNS is the total number of connections with the partner location. The ATT column is blank unless the DETAIL option is specified.

**No locations found:** When the DISPLAY LOCATION command is unable to find any locations associated with the specified list of locations, a message line is shown to indicate that no locations were found.

If all locations were specified, there is no distributed activity. If a location list was specified, reissue the DISPLAY LOCATION command using the LOCATION(\*) parameter to display information for all locations.

**Truncated report:** If the report was generated in response to a command from an MVS console and more than 254 response messages were generated, a message line is shown to indicate that the report is truncated.

You can reissue the DISPLAY LOCATION command and specify a specific location or list of locations to see a complete report for that context.

**Detail report:** When a DISPLAY LOCATION command is issued with the DETAIL option, the report might contain additional message lines for each remote location.

The report might contain one or more lines that provide information about the number of connections from or to remote systems that have certain attributes:

## LOCATION

The identifier for the remote location.

- For TCP/IP requester connections, the IP address concatenated with the port number of the remote location. For TCP/IP server connections, the IP address of the remote location.
- The LU name of the remote location, for SNA connections.

**PRDID** An alphanumeric string that identifies the product, in the form *pppvrrm*:

*ppp* A three-character code that identifies the product:

**AQT** IBM DB2 Analytics Accelerator for z/OS

**ARI** DB2 for VSE & VM

**DSN** DB2 for z/OS

**JCC** IBM Data Server Driver for JDBC and SQLJ

**QSQ** DB2 for i

**SQL** DB2 for Linux, UNIX, and Windows

*vv* A two-digit number that identifies the product version number, such as 09 or 10.

*rr* A two-digit number that identifies the product release level, such as 01 or 05.

*m* A one-digit number that identifies the product modification level, such as 0 or 1.

**T** The connection type with the remote location. Expected values are:

**R** A requester connection from local subsystem accessing a remote system.

**S** A server connection from remote system accessing local subsystem.

## ATT and CONNS

The number of connections that are associated with the remote location and have the attribute that is identified by the ATT column. Because more than one attribute might apply to the same connection, the total number of connections is not a sum of the numbers of connections for the different attributes. The Expected values for ATT are:

**AES** Using security credentials with AES encryption

**IPS** Using IPsec

**TLS** SSL using Application Transparent-Transport Layer Security (AT-TLS)

**TRS** Using trusted context

**WLB** Using workload balancing connections

**XA** Using XA transaction manager processing

The report might also include lines that contain detailed information about the connections associated with the DB2 system thread. A line identified by the partial message number L203 provides headings that describe the values in one or more following lines that are identified by the partial message number L204.

## SYSTASK

Describes the DB2 system threads. The expected values are:

### SYSPLX-x

Indicates SNA-related SYSPLEX processing where a DB2 systems periodically create connections to a remote data sharing group to obtain WLM balancing information related to all members of the group.

The expected values of *x* are:

**O** The local DB2 system is contacting a remote member of a data sharing group to request WLM balancing information.

**I** The local DB2 system is receiving a connection that requests WLM balancing information.

**RESYNC**

A connection used to resynchronize a unit of work that encountered a previous thread or communication error. There can be one entry representing each thread needing resynchronization with the remote site.

**SESSID**

The identifier of the session:

- For SNA network connections, the VTAM-defined session instance identifier of the session on which the conversation is executing. If the session identifier is not applicable, this field contains zeros.

You can use the VTAM DISPLAY NET,ID=db2-luname,SCOPE=ACT command to obtain the full session ID. The DISPLAY NET command lists all sessions (SID) for the DB2 logical unit. Scan the DISPLAY NET output for the complete SID.

- For TCP/IP network connections, a string of the form *local:remote* where:

*local* specifies the local DB2 system's TCP/IP port number

*remote* specifies the remote partner's TCP/IP port number

**A** The connection activity:

**N** The connection is active in the network.

**W** The connection is suspended in DB2 waiting for VTAM or TCP/IP notification that the function is complete.

**blank** No specific activity condition exists.

**ST** The status of the connection:

**A** connection is in allocation.

**C** Session limits are being negotiated (CNOS) with the SNA partner prior to connection allocation.

**D** connection is in deallocation.

**R** Receiving.

**S** Sending.

**blank** No specific status condition exists.

**TIME** The timestamp of the last message sent or received on the connection, in the format *yydddhmmssstth*.

*yy* The last two digits of the year.

*ddd* The number of the day in the year. For example, February 1 is represented as 032.

*hhmmssstth*

The time of day in 24-hour format, to hundredths of the second.

**Report conclusion:** The following message line identifies the end of the output of the DISPLAY LOCATION command:

DISPLAY LOCATION REPORT COMPLETE

**System action:** Processing continues.

**User response:** No action is required.

**Related tasks:**

 Displaying information about connections with other locations (DB2 Administration Guide)

**Related reference:**

Chapter 32, “-DISPLAY LOCATION (DB2),” on page 241

---

## 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).

### 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 - DSNJCO0A 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)
DSNJ371I - DB2 RESTARTED 14:06:23 MAY 22, 2011
RESTART RBA 0000039A8000
DSN9002I - 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.

---

## 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

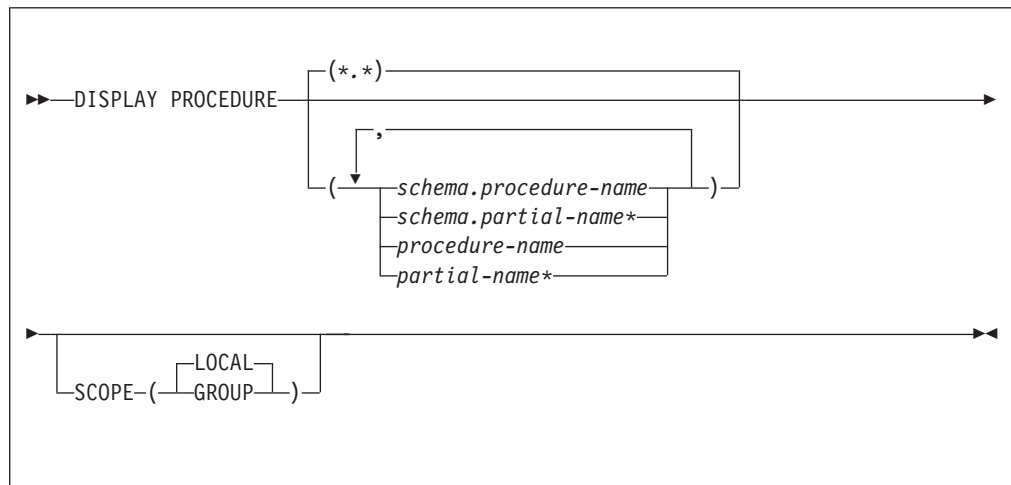
If you specify DISPLAY PROCEDURE \*.\* or *schema.partial-name \**, the privilege set of the process must include one of the following authorities:

- 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

*Sample output:* The DISPLAY PROCEDURE command generates output similar to the following output:

DSNX940I = DSNX9DIS DISPLAY PROCEDURE REPORT FOLLOWS -

PROCEDURE	STATUS	ACTIVE	QUED	MAXQ	TIMEOUT	FAIL	WLM_ENV
APPL1	STARTED	1	0	0	0	1	SANDBOX
APPL2	STARTED	1	0	0	0	0	SANDBOX
APPL2	STARTED	0	1	2	0	0	SANDBOX
APPL5	STOPREJ	0	0	0	0	0	SANDBOX
APPL6	STOPABN	0	0	0	0	0	SANDBOX
PROC1	STOPQUE	0	0	0	0	0	SANDBOX

DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE

DSN9022I = DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION

*Description of output:* Each output line displays the following information:

### PROCEDURE

The name of the stored procedure.

### STATUS

The status of the stored procedure of the stored procedure. The possible values are:

#### STARTED

Requests for the procedure can be processed.

#### STOPQUE

Requests are queued.

#### STOPREJ

Requests are rejected.

#### STOPABN

Requests are rejected because of abnormal termination.

### ACTIVE

The number of threads that are currently running the load module.

### QUED

The number of threads that are waiting for the procedure to be scheduled.

### MAXQ

The maximum number of threads that have waited concurrently for the procedure to be scheduled since DB2 was started. DB2 resets this value to 0 each time you run the START PROCEDURE command.

### TIMEOUT

The number of times an SQL CALL statement timed out while waiting for a request for the procedure to be scheduled. DB2 resets this value to 0 each time you run the START PROCEDURE command.

### FAIL

The number of times a procedure has failed. DB2 resets this value to 0 each time you run the START PROCEDURE command.

### WLM\_ENV

The WLM environment where the procedure runs.

Message DSNX943I lists a range of procedures that are stopped because a STOP PROCEDURE command included a partial name with a pattern-matching character (\*), as in the following example:

-STOP PROCEDURE(ABC\*)

Message DSNX950I is returned when DISPLAY PROCEDURE is issued for a procedure name that has not been accessed by a DB2 application.

**Displaying native SQL procedures:** Native SQL procedures are not displayed in the DISPLAY PROCEDURE output unless you run the procedure in DEBUG mode. If you do run the 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 native SQL procedures as '0' if the native SQL procedures are under the effect of a STOP PROCEDURE command.

## 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-
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 indicates the beginning of the output of the command.

---

| **DSNT753I**    *csect-name* **DISPLAY PROFILE REPORT FOLLOWS:**

| **Explanation:** This message displays multiple lines of information in response to a DISPLAY PROFILE command.

|    *csect-name*

|        The name of the control section that issued the message.

| The output from the command consists of one or more (but not necessarily all) of the following sections, in the indicated order:

## DSNT753I

| **Current profiling status:** The report contains information about the current profiling status:  
| STATUS=*profile-status*

| **STATUS** *profile-status*  
|     The profiling status. Expected values are:

|     **ON**     Profiling is active.

|     **OFF**     Profiling is inactive.

|     **SUSPENDED**  
|         Profiling was active, but is suspended now due to some error conditions. Typical error conditions  
|         include that the maximum push-out limit has been reached, or that the table space is full.

|     **STARTING**  
|         Profiling is being started, but has not yet completed.

|     **STOPPING**  
|         Profiling is being stopped, but has not yet completed.

| **Information about when profiling started:** If the profile status is ON or SUSPENDED, the report contains  
| information about when profiling started:  
| TIMESTAMP=*profile-timestamp*

| **TIMESTAMP** *profile-timestamp*  
|     The date and time when the profiling status started, in the format YYYY-MM-DD-HH.MN.SS.NNNNNN.

| **Information about pushouts:** The report contains information about reports issued for monitored SQL statements:  
| PUSHOUT=*current-pushouts* OUT OF *maximum-pushouts*

| **PUSHOUT** *current-pushout*  
|     The current number of reports issued for all monitored SQL statements since the profile was started. This  
|     value is applicable only for profiles that monitor SQL statements. This value is reported only when the  
|     profile status is ON or SUSPENDED.

| **OUT OF** *maximum-pushouts*  
|     The maximum number of total reports allowed for all (normal and exception) statement monitoring during  
|     the lifetime of a monitor profile.

| **End of report:** The report normally ends with the following message:  
| DISPLAY PROFILE REPORT COMPLETE

| **System action:** Processing continues.


| **System programmer response:** Take the action indicated for the *profile-status* value:

| **ON**     No action is required. To stop profiling, issue STOP PROFILE.

| **OFF**     No action is required. To start profiling, issue START PROFILE.

| **STARTING or STOPPING**  
|     Wait until the command execution is complete.

| **SUSPENDED**  
|     Correct the problem that is causing the suspension of profiling activities, then stop or restart profiling.

| **Related concepts:**  
|  Profiles (DB2 Performance)

---

## 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:

- 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 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
```



---

## 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.

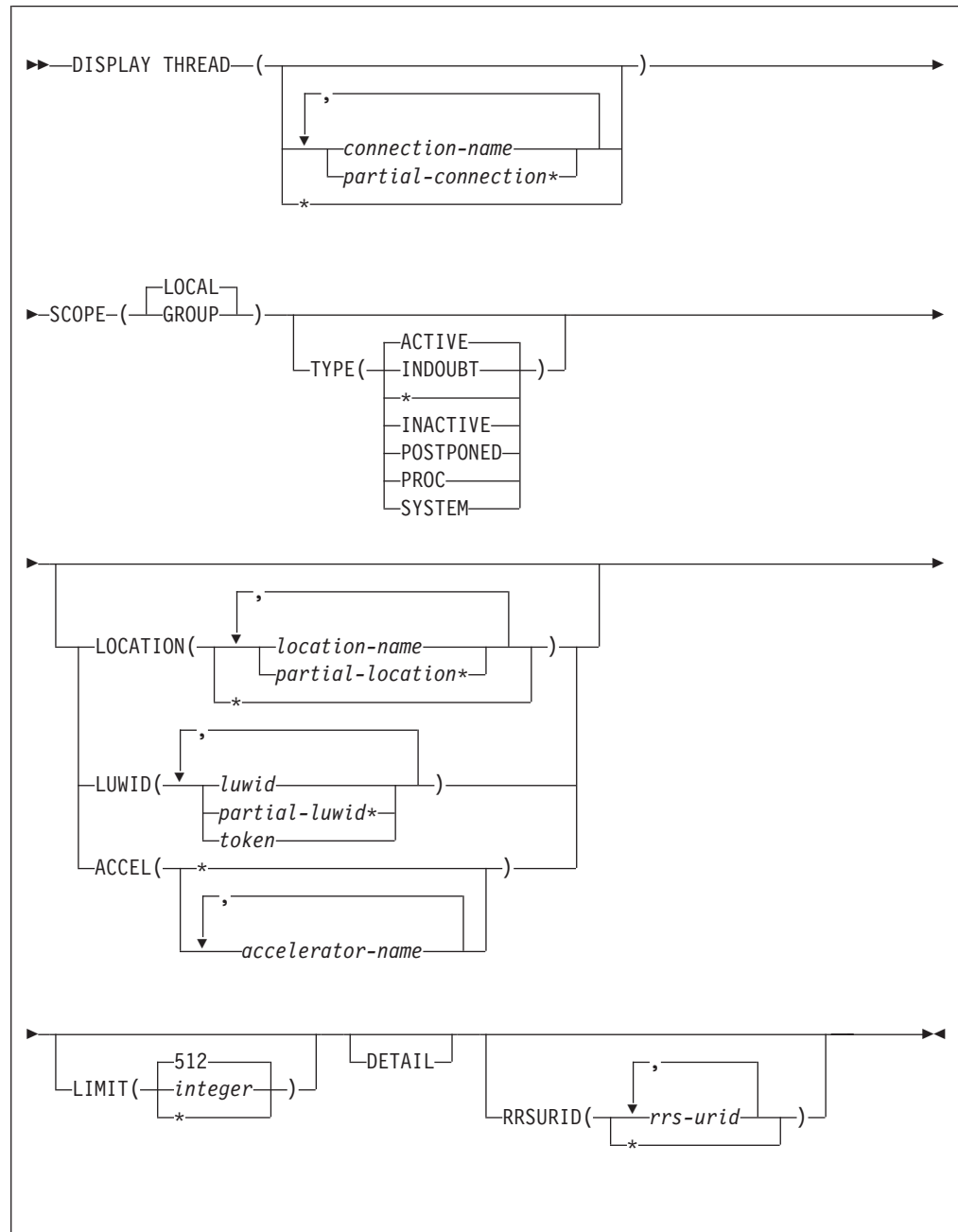
### Authorization

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
- 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

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 messages DSNV403I and DSNV404I.

**(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 RRSF. 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 DSNV404I. 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.

#### LUWID( *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.

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 255 lines (at which time a DSNV421I or DSNV422I 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, ACCEL, 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 DSNV404I messages.

The command preprocessor (GCPC) console commands tasks can be canceled, and a token will be provided for these threads in a DSNV404I 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 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      0522008555443
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.C0C44BBCCFF4=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 B4D0FC267EB020000000001101010000
- 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 C4D4FA267EB040000000001201020000
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.GlmTsk00  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)

#### | DSNV401I DISPLAY THREAD REPORT FOLLOWS -

##### | Explanation:

| This message marks the beginning of multiple lines of information in response to a DISPLAY THREAD command.  
| Some lines in the output have their own message numbers or alphanumeric identifiers to assist with identification.

| View this topic in the information center.

| The output from the command consists of one or more (but not all) of the following sections:

- | • If SCOPE (GROUP) was specified, any thread-specific information is preceded by relevant information about the member: Member-specific information (DSNV473I, DSNV747I, DSNV475I, DSNV476I, DSNV503I)
- | • The member-specific information, or the DSNV401I message line if SCOPE(GROUP) was not specified, is followed by an information block that reports on threads by type:
  - | – Information about active, inactive, stored procedure, and system threads (DSNV402I, DSNV424I, and DSNV497I)
  - | – Information about indoubt and postponed threads (DSNV406I and DSNV431I)

| If TYPE(\*) was specified, the output contains multiple information blocks.

- | • Depending on other command options or system conditions, some additional detail lines are present within the report block:

- | – V429: Information about stored procedures
- | – V436: Information about SQL statements
- | – V437: Information about the user and application
- | – V440: Information about threads that are managed by an XA transaction manager
- | – V441: User accounting information
- | – V442: Information about correlation tokens
- | – V443: Information about parallelism
- | – V444: Information about distributed threads (for active, inactive, stored procedure, and system threads)
- | – V445: Information about database access threads
- | – V447 and V448: Information about conversations
- | – V449: Information about IMS and CICS connections and RRS
- | – V450: Information about downstream participants that might be indoubt
- | – V451: Information about incomplete resolution of indoubt threads
- | – V452: Information about heuristic commit decisions
- | – V453: Information about heuristic abort decisions



## DSNV401I

|   – V457: Information about resync protocol errors with participants  
|   – V458: Connection information for indoubt threads  
|   – V459: Information about syncpoint protocol errors with coordinators  
|   – V460: Information about syncpoint protocol errors with participants  
|   – V461: Information about heuristic damage  
|   – V462: Information about cold starts by participants  
|   – V463: Information about cold starts by coordinators  
|   – V464: Information about resync protocol errors with coordinators  
|   – V465: Information about threads that remain in the prepared state  
|   – V466: Information about how long a thread was indoubt  
|   – V467: Information about distributed threads (for indoubt and postponed threads)  
|   – V471: Information about available agents  
|   – V480 and V481: Information about the unit of recovery  
|   – V482: WLM information  
|   – V485: Information about trusted contexts  
|   – V490: Information about suspended threads  
|   – V492: Information about storage  
|   – V501: Information about commands  
|   – V502: Information about DB2 exit routines  
|   – V504: Information about threads that are terminating  
|   – V505: Information about threads that are waiting  
|   – V507: Information about monitoring tasks  
|   • Information that indicates NOT FOUND situations  
|   • Other information reported in the DISPLAY THREAD output  
|   • End of report markers  
|   • Error messages  
| **Member-specific information (DSNV473I, DSNV747I, DSNV475I, DSNV476I, DSNV503I):**  
| If you specify the SCOPE(GROUP) option, the output includes member-specific information before the thread-specific  
| information for that member.  
| Depending on the DISPLAY THREAD options and the threads that are found, the output includes one or more of the  
| following lines. Each line is followed by thread-specific information for the specified type of thread on that member:  
|   DSNV473I ACTIVE THREADS FOUND FOR MEMBER: *text*  
|   DSNV474I INDOUBT THREADS FOUND FOR MEMBER: *text*  
|   DSNV475I INACTIVE THREADS FOUND FOR MEMBER: *text*  
|   DSNV476I POSTPONED ABORT THREADS FOUND FOR  
|       MEMBER: *text*  
|   DSNV503I SYSTEM THREADS FOUND FOR MEMBER: *text*  
| *text*     Member name for which thread information is being displayed.  
| **Information about active, inactive, stored procedure, and system threads (DSNV402I, DSNV424I, and**  
| **DSNV497I):**  
| The DISPLAY THREAD command reports the same type of information about active, inactive, stored procedure, and  
| system threads.  
| The TYPE(ACTIVE) and TYPE(PROC) options generate a DSNV402I message.  
| The TYPE(INACTIVE) option generates a DSNV424I message.  
| The TYPE(SYSTEM) option generates a DSNV497I message.  
| Each thread that meets the report criteria is listed on a separate line. Additional information might be provided for  
| individual threads.



| **DB2** *db2-statestate-phase*

|       The state of DB2 as follows:

|       *db2-state*

|           The state of DB2. Possible values are STARTING, STOPPING, or ACTIVE.

|       *state-phase*

|           The phase of restart or shutdown, if available. If *db2-state* is ACTIVE, no phase information is provided.

| **NAME** *text*

|       The connection name that was used to establish the thread.

|       If the connection status value is "D", the connection name is either the connection name for the allied address space or the DB2 subsystem name.

|       For distributed database access threads, this field contains the following information:

|       *requester-value*

|           For threads that use system-directed access or application-directed access from a DB2 requester, this value is the connection name of the thread at the requesting location.

|       **SERVER**

|           For threads that use application-directed access from a non-DB2 requester, no connection name is identified.

|       **blank**   For threads where the originating task is running on the same DB2 subsystem, this value is blank. When the connection name value is blank, the expected connection status value is "PT".

| **ST** *t*   A one- or two-letter code that indicates connection status. Possible values are:

|       **AC**       A thread is executing in an accelerator server. This status is displayed until accelerator processing concludes and returns control to DB2.

|       **D**        The thread is in the process of termination as a result of the termination of the associated allied task. If this thread is also the last (or only) DB2 thread for the address space, the associated allied task is placed in a wait state.

|           For an active thread, this value can indicate possible problems with DB2. If the activity indicator value is \*, use the information in message DSN3201I to identify and resolve any problems.

|       **DA**       The database access thread slot is currently not associated with a remote connection and is available to be assigned to a type 2 inactive thread.

|       **DI**       The thread is disconnected from an execution unit. No TCB is associated with the DB2 thread.

|           This value is possible is only when the connection name is "RRSAF".

|       **N**        The thread is in either IDENTIFY or SIGNON status.

|           This value is accompanied by a token value of "0".

|       **ND**       The thread is in either IDENTIFY or SIGNON status, and the thread is currently not associated with any TCB.

|       **PT**       A parallel task thread was established (plan allocated).

|       **QD**       The thread is queued for termination as a result of the termination of the associated allied task. If this thread is also the last (or only) DB2 thread for the address space, the associated allied task is placed in a wait state.

|           For an active thread, this value can indicate possible problems with DB2. If the activity indicator value is \*, use the information in message DSN3201I to identify and resolve any problems.

|       **QT**       The CREATE THREAD request was queued. The associated allied task is placed in a wait state.

|       **R2**       A distributed thread is accessing a remote site on behalf of a request from another location. The thread is currently an inactive connection (type 2 inactive thread) and is waiting for an agent to become available.

|       **RA**       A distributed thread is accessing a remote site on behalf of a request from another location.

<b>RK</b>	A distributed thread is accessing a remote site on behalf of a request from another location. The thread is performing an operation that invoked Kerberos services. This status is displayed until Kerberos services returns control to DB2.
<b>RN</b>	<p>A distributed thread is accessing a remote site on behalf of a request from another location. The request accesses data at another DB2 location. Therefore, the thread was suspended until DB2 connects to the partner location (establishes DB2 system conversations with the partner).</p> <p>When another DB2 site is being contacted for the first time using system-directed access, DB2 must establish DB2 system conversations with the partner location. A request is presented to a DB2 DDF service task that establishes a system conversation. The thread is suspended until the request is processed. If the thread remains in this status for an extended period of time, the DB2 service task that establishes the system conversations might be busy processing other requests. The DB2 DISPLAY LOCATION command (DISPLAY LOCATION(*) DETAIL) shows conversation activity for this DB2 system conversation (SYSCON-O) service task.</p>
<b>RQ</b>	<p>A distributed thread is accessing a remote site on behalf of a request from another location.</p> <p>The thread is suspended for one of the following reasons:</p> <ul style="list-style-type: none"> <li>• The maximum number of active database access threads was reached. The number of threads is described by the MAX REMOTE ACTIVE value of the DSN6SYSP macro in the DB2 startup parameter, usually DSNZPARM.</li> <li>• The system profile monitoring threshold was reached. The threshold is described in the DSN_PROFILE_ATTRIBUTES table.</li> </ul> <p>Database access agents (DBAAs) are queued until other DBAAs deallocate or go inactive, providing an available slot. The DBAA resumes when a slot becomes available, and the DBAA is next in the queue.</p> <p>Consider increasing the MAX REMOTE ACTIVE value.</p>
<b>RX</b>	The distributed thread is executing an XA transaction on behalf of a request from another location.
<b>SP</b>	A thread is running within a stored procedure. This status value is displayed until the stored procedure terminates and returns control to DB2.
<b>SW</b>	A thread is waiting for a stored procedure to be scheduled. This status value is displayed until the stored procedure begins to run.
<b>T</b>	An allied, nondistributed thread was established (plan allocated).
<b>TD</b>	An allied thread was established (plan allocated), and the thread is currently not associated with any TCB.
<b>TN</b>	<p>An allied thread was distributed to access data at another DB2 location, but it was suspended because DB2 was not connected to the partner location.</p> <p>When another DB2 site is being contacted for the first time using system-directed access, DB2 must establish DB2 system conversations with the partner location. A request is presented to a DB2 DDF service task that establishes a system conversation. The thread is suspended until the request is processed. If the thread remains in this status for an extended period of time, the DB2 service task that establishes the system conversations might be busy processing other requests. The DB2 DISPLAY LOCATION command (DISPLAY LOCATION(*) DETAIL) shows conversation activity for this DB2 system conversation (SYSCON-O) service task.</p>
<b>TR</b>	An allied thread was distributed to access data at another location.
<b>A t</b>	An indicator that the thread is active within DB2. <i>t</i> can have one of the following values:
<b>*</b>	The thread is active.
<b>blank</b>	The thread is inactive.
<b>REQ num</b>	A counter that shows the number of DB2 requests. This counter restarts at 1 when the count exceeds 32767 requests.
<b>ID text</b>	The recovery correlation ID that is associated with the thread.

For distributed database access threads, this field contains one of the following values:

**028.DBAA** *nn*

This value indicates that the database access agent (DBAA) is performing connection processing, which consists of establishing the DBAA thread and validating and verifying the user's ID.

RACF requests are serialized on one of the DB2 RACF service tasks. This serialization might increase the validation time when multiple DBAAs are being created concurrently.

This value is possible only while the thread is being created.

*requester-value*

For threads that use system-directed access or application-directed access from a DB2 requester, this value is the correlation ID of the thread at the requesting location.

This value is possible only after the thread has been created.

*external-name*

For threads that use application-directed access from a non-DB2 requester, this value is the first 12 characters in the DDM external name (EXTNAM) parameter of the DDM EXCSAT command that is received as part of the SQL CONNECT statement.

This value is possible only after the thread has been created.

**AUTHID** *text*

The authorization ID that is associated with a signed-on connection.

**PLAN** *text*

The plan name that is associated with the thread.

For distributed database access threads, this field contains one of the following values:

**DISTSERV**

For threads that use application-directed access from a non-DB2 requester, no plan name is identified.

*requester-value*

For threads that use system-directed access or application-directed access from a DB2 requester, this value is the plan name that is being executed at the requesting location.

For RRSF connected threads that did not specify a plan name at connect time, the value ?RRSAF is displayed.

If a thread was not established, this field is blank.

If this value is a system plan, see the information about system plans in message DSNT376I.

**ASID** *hex*

The address space ID (ASID) of the home address space. This value is a hexadecimal number of up to four characters.

**TOKEN** *num*

The thread token that is assigned to the thread. For threads with a connection status value of "N", this value is "0".

The information about types of threads ends with a REPORT COMPLETE line.

**Information about indoubt and postponed threads (DSNV406I and DSNV431I):**

The DISPLAY THREAD command reports the same type of information about indoubt and postponed threads.

Information about indoubt threads begins with a DSNV406I message.

Information about postponed abort threads begins with a DSNV431I message.

Each thread that meets the report criteria is listed on a separate line. Additional information might be provided for individual threads.

**COORDINATOR** *text*

The name of the two-phase commit coordinator. The value depends on the type of thread, as follows:

**Allied threads**

The coordinator name is one of the following values:

- The IMS connection name
- The CICS connection name
- RRS (for Resource Recovery Services attachment facility connected threads)
- The location name of the local DB2 subsystem.

Additional information is displayed for IMS and CICS connections and for RRS.

**Database access threads**

The coordinator name is one of the following values:

- *location:luname*, where *location* is the relational database name for the coordinator and *luname* is the SNA LU name for the coordinator.
- *location:port*, where *location* is the relational database name for the coordinator and *port* is the TCP/IP port number for two-phase commit resynchronization for the coordinator.
- *<luname>:luname*, where *luname* is the SNA LU name for the coordinator.
- *nnn.nnn.nnn.nnn:port*, where *nnn.nnn.nnn.nnn* is the IP address for the coordinator and *port* is the TCP/IP port number for two-phase commit resynchronization for the coordinator.

If the thread is distributed, additional information is displayed.

**STATUS** *text*

The status of the thread.

If the status of a thread for a logical unit of work at a participant is INDOUBT, COMMITTED-H, or ABORTED-H, and the coordinator is a DB2 location that has not cold started, the lack of any information about the thread in a DISPLAY THREAD(INDOUBT) report indicates that the decision at the coordinator was to abort the logical unit of work.

This field can have one of the following values:

**ABORT-P**

A postponed abort unit of recovery. Objects for which this unit of recovery has backout work pending are inaccessible (restart-pending status) until the abort is completed (for example, by means of the RECOVER POSTPONED command).

**ABORT-PSTRT**

A postponed abort unit of recovery that is currently undergoing RECOVER POSTPONED processing or automatic DB2 backout processing (requested by restarting with system parameter LBACKOUT = AUTO).

**ABORTED**

A coordinator status that indicates that DB2 has one or more downstream participants that are using the Presumed Nothing (PN) protocol and might be indoubt. The thread is included in the DISPLAY THREAD output until all downstream PN participants complete indoubt resolution. The detail line V450 lists the downstream participants that have pending resolution. Database locks that were held for the thread are released.

If a participant resolved the indoubt thread with a heuristic decision, and the decision was COMMIT, detail line V452 is displayed. This detail line contains the name of the participant and the heuristic decision that it made. Because the coordinator aborted and the participant committed, this situation indicates heuristic damage. Information about the thread is displayed until it is deleted with the RESET INDOUBT command. The database administrators at all involved locations need to know that heuristic damage occurred and at which location.

**ABORTED-H**

An indoubt thread that was heuristically resolved with the RECOVER INDOUBT command. Information about the thread is displayed until the coordinator is informed of the heuristic decision. Additional detail lines might be included in the report.

**COMMITTED**

A thread that is committed. This status is displayed when DB2 is the coordinator and has downstream participants that might be indoubt. Information about the thread is included in the

DISPLAY THREAD output until all downstream participants complete indoubt resolution. Detail line V450 lists the downstream participants that have pending resolution. Database locks that were held for the thread are released.

If a participant resolved the indoubt thread with a heuristic decision, and the decision was ABORT (rollback), detail line V453 is displayed. This detail line contains the name of the participant and the heuristic decision that it made. Because the coordinator committed and the participant aborted, this situation indicates heuristic damage. Information about the thread is displayed until it is deleted with the RESET INDOUBT command. The database administrators at all involved locations need to know that heuristic damage occurred and at which location.

#### COMMITTED-H

An indoubt thread that was heuristically resolved with the RECOVER INDOUBT command. Information about the thread is displayed until the coordinator is informed of the heuristic decision, and all downstream participants, if any, complete indoubt resolution. Additional detail lines might be included in the report to identify downstream participants that have pending resolution.

#### INDOUBT

A thread that is indoubt. Information about the thread is displayed until all indoubt resolution responsibilities are complete. Additional detail lines might be included in the report.

#### RESET *txt*

If this column contains "YES", the thread must be removed from the indoubt display. If necessary, use the RESET INDOUBT command to purge a thread. If this column is empty, the thread does not need to be removed.

#### URID *hex*

An RBA of the DB2 recovery log. This point is the beginning of recovery logging for this thread.

#### AUTHID *text*

The primary authorization ID that is associated with the thread.

The information about types of threads ends with a REPORT COMPLETE line.

#### V429: Information about stored procedures:

If a thread is running a stored procedure, the output includes the following information about the stored procedure.

#### PROCEDURE *text*

The name of the stored procedure.

#### LOAD MODULE *text*

The MVS load module that is associated with the stored procedure. This value is empty until the load module name is determined.

#### PROC *text*

The name of the JCL PROC that is used to start the address space where the stored procedure is running. This field is empty until the stored procedure is assigned to a specific stored procedure address space.

#### ASID *hex*

The MVS address space ID (ASID) of the address space where the stored procedure is running. This field contains the value 0000 until the stored procedure is assigned to a specific stored procedure address space.

#### WLM\_ENV *text*

The name of the WLM application environment where the stored procedure is running.

#### V436: Information about SQL statements:

If you specify a detail report and a thread is processing an SQL statement, the output includes the following information about the SQL statement and the program that contains the statement.

#### PGM *text*

The collection ID and associated package or DBRM member. This information is provided in the following format:

*collection.package*

If the SQL statement that is being executed is not associated with a package, *collection* is not applicable and is indicated by an asterisk (\*).

## SEC *num*

The SQL section number that is associated with the package or DBRM.

## STMT *num*

The SQL statement number that is associated with the package or DBRM.

## THREAD-INFO *text*

Information about the thread. The information is presented in a colon-delimited list that contains the following segments:

- The primary authorization ID that is associated with the thread.
- The name of the user's workstation.
- The ID of the user.
- The name of the application.
- The statement type for the currently executing statement: dynamic or static.
- The statement identifier for the currently executing statement, if available. The statement identifier can be used to identify the particular SQL statement.
  - For static statements, the statement identifier correlates to the STMT\_ID column in the SYSIBM.SYSPACKSTMT table.
  - For dynamic statements, the statement identifier correlates to the STMT\_ID column in the DSN\_STATEMENT\_CACHE\_TABLE table.
- The name of the role that is associated with the thread.
- The correlation token that can be used to correlate work at the remote system with work that is performed at the DB2 subsystem. The correlation token, if available, is enclosed in '<' and '>' characters, and contains three components, separated by periods:
  - A 3 to 39 character IP address
  - A 1 to 8 character port address
  - A 12 character unique identifier

An asterisk (\*) in any segment indicates that the information is not available.

## V437: Information about the user and application:

If the client system is able to provide specific information about the user that is associated with this thread, this information is included in the output. If the information is not provided by the client system, the relevant fields contain an asterisk (\*).

## WORKSTATION *text*

The workstation name

## USERID *text*

The user ID

## APPLICATION NAME *text*

The name of the client application

## V440: Information about threads that are managed by an XA transaction manager:

If the thread is related to a global transaction that is coordinated by an XID, the output includes XID information. The DISPLAY THREAD command must include the TYPE(ACTIVE) and DETAIL options, or the TYPE (INDOUBT) option for this information to be included.

Line V440 indicates that the thread is managed by an XA transaction manager, such as WebSphere® Application Server, which identifies the transaction with an *xid* value. The XID is provided to allow correlation with the XA transaction manager. DB2 uses both the logical unit of work identifier, *luwid*, and the XA transaction identifier, *xid*, to coordinate and recover transactions. DB2 is a participant in an XA transaction that is executing on behalf of an XA transaction manager.

## XID *hex*

The XID is provided in its hexadecimal representation and consists of the following elements:

- A 4-byte format ID.
- A 4-byte global transaction identifier length (the gtrid\_length field).
- A 4-byte branch qualifier length (bqual\_length).

- Variable-length data that is composed of at most two contiguous components: A global transaction identifier length (*gtrid\_length*), and a branch qualifier length (*bqual\_length*). The *gtrid\_length* element specifies the number of bytes (1-64) that constitute the *gtrid*, starting at the first byte of the data element. The *bqual\_length* element specifies the number of bytes (up to 64) that constitute *bqual*, starting at the first byte after *gtrid*.

#### V441: User accounting information:

If the client system is able to provide user accounting information for the thread, this information is included in the output. The DISPLAY THREAD command must include the DETAIL option and either the TYPE(ACTIVE), TYPE(INACTIVE), or TYPE(INDOUBT) option for this information to be included.

#### ACCOUNTING *text*

The client end-user accounting information. This information is provided if the appropriate information is provided by the client system, and if DB2 recognizes the format of the information.

#### V442: Information about correlation tokens:

If the client system is able to provide a correlation token that can be used to correlate work at the remote system with work that is performed at the DB2 subsystem, this information is included in the output.

#### CRTKN *text*

The correlation token, which is made up of three components separated by periods:

*ip-address.port-address.unique-id*

*ip-address*

The IP address is 3 to 39 characters in length.

*port-address*

The port address is 1 to 8 characters in length.

*unique-id*

The ID is 12 characters in length.

•

#### V443: Information about parallelism:

If a parallel task thread is running on an assisting DB2 subsystem, the output includes information about the DB2 member name and the thread token for the originating task.

#### COORDINATOR *text*

The coordinating DB2 member where the originating task thread is running.

#### ORIGINATING TOKEN *num*

The originating task thread token.

#### V444: Information about distributed threads (for active, inactive, stored procedure, and system threads):

If any threads were distributed to other locations, the output includes the logical unit of work identifier for each distributed thread.

#### *text=num* ACCESSING DATA AT

Information to identify the distributed thread as follows:

*text* The global logical unit of work ID (LUWID).

*num* The local token that identifies the thread. This token can be used in place of the LUWID in any DB2 command that accepts LUWID as input.

If the thread becomes indoubt, a new token is assigned for the indoubt thread.

#### *text-address..number*

The remote DBMS at which data is being accessed.

*text* The partner location name.

*address* The partner LU name for an SNA connection, or the dotted decimal IP address for a TCP/IP connection (nnn.nnn.nnn.nnn).

*number* An SQL port number. This token is included only if the second token is an IP address.



**V445: Information about database access threads:**

The output includes the following information about database access threads:

*text=num*

Information to identify the distributed thread as follows:

*text*      The global logical unit of work ID (LUWID).

*num*      The local token that identifies the thread. This token can be used in place of the LUWID in any DB2 command that accepts LUWID as input.

If the thread becomes indoubt, a new token is assigned for the indoubt thread.

**ACCESSING DATA FOR *text***

The network address of the remote location. This value can be blank.

- If the connection with the requester is through SNA, this field contains the relational database name of the requester or the VTAM LU name of the requester, a dash (-) delimiter, and the LU name of the requester.
- If the connection with the requester is through TCP/IP, this field contains the IP address of the requester.

**V447 and V448: Information about conversations:**

The output contains conversation information for active and inactive threads. Conversation information is displayed for a distributed access thread, distributed inactive connection, or distributed allied thread when a detail report is specified. Information is displayed for each conversation that is associated with the thread or connection. The report lists detailed information regarding the connection.

**INDEX (*text*)**

The index value from the location information in the V445 output line. This value provides correlation of the conversation detail information to the associated location.

**SESSID *text***

The session identifier. The session identifier is either a pair of TCP/IP port numbers or a VTAM-defined session instance identifier.

**TCP/IP connections**

The local DB2 TCP/IP port number followed by the TCP/IP port number for the partner. The two port numbers are separated by a colon. For example: 5001:28191.

**SNA partners**

The VTAM-defined session instance identifier of the session on which the conversation is running. The field contains zeros if the session identifier is not applicable.

VTAM might not supply the entire session ID (SID) to DB2. The first two digits of the SID might be incorrect, in which case the SID, as presented in this message, is not acceptable in VTAM commands. If this occurs, use the VTAM DISPLAY NET, ID=*db2-luname*, SCOPE=ACT command to obtain the full SID. The DISPLAY NET command lists all sessions for the DB2 logical unit. Review the DISPLAY NET report for the complete session ID.

**A *text***    A single alphabetic character that indicates activity. This field can contain one of the following values:

**N**      The conversation is active in the network.

**W**      The conversation is suspended in DB2 while it waits for the network notification that the function is complete.

**Blank**    All other cases.

**ST *text***    The status of the conversation. This information is presented as two characters and can contain the following values:

**First character**

**A**      Conversation is in allocation.

**C**      Session limits are being negotiated with the partner (CNOS) prior to conversation allocation.

**D**      Conversation is in deallocation.



<b>R</b>	Receiving.
	<ul style="list-style-type: none"> <li>At the requesting site, this value indicates that the conversation is receiving a response from a request.</li> <li>At the server site, this value indicates that the conversation is receiving or waiting for a request.</li> </ul>
<b>S</b>	Sending.
	<ul style="list-style-type: none"> <li>At the requesting site, this value indicates that the conversation can send requests to the server.</li> <li>At the server site, this value indicates that the conversation is sending or preparing to send a response.</li> </ul>
<b>X</b>	Exchanging log name information with the partner before conversation allocation to determine if the partner supports protected conversations. If the thread remains in this status for an extended period of time, the DB2 service task that exchanges log names might be busy processing other requests. The DB2 DISPLAY LOCATION command (DISPLAY LOCATION(*) DETAIL) shows you conversation activity for this (RESYNC) service task.
<b>Blank</b>	All other cases.

#### Second character

<b>1</b>	Unprotected conversation using system-directed access is active.
<b>2</b>	Unprotected conversation using application-directed access is active.
<b>3</b>	Protected conversation using system-directed access is active.
<b>4</b>	Protected conversation using application-directed access is active.
<b>Blank</b>	All other cases.

#### TIME *timestamp*

The timestamp (*yyddhlmmssth*), in local time, of the last message that was sent or received on the conversation. The timestamp consists of the following segments:

<i>yy</i>	Two-digit indicator for the year.
<i>ddd</i>	Three-digit indicator for the day of the year.
<i>hlmmssth</i>	Indicator for the time of the day.

#### V449: Information about IMS and CICS connections and RRS:

When the coordinator is an IMS or CICS connection or is RRS, the output includes the following additional information about the coordinator.

#### NID *text*

The network ID that is assigned by IMS or CICS, or, if the connection type is RRSAF, the RRS unit of recovery ID.

**ID *text*** The correlation ID that is assigned by IMS or CICS, or, if the connection type is RRSAF, the correlation ID that is assigned by the connected application.

#### V450: Information about downstream participants that might be indoubt:

If you specify TYPE(INDOUBT) in the DISPLAY THREAD command, the output also includes the following information about any downstream participants that might be indoubt. The indoubt participants that are listed in the output might be the result of an abnormal termination of this DB2 subsystem. The LUWID of the indoubt thread is displayed in the COORDINATOR field prior to this message.

#### HAS PARTICIPANT INDOUBT AT *text address number*

The name of a remote DBMS at which data is being accessed or where a thread is indoubt.

- text* is the partner location name.
- address* is the partner LU name for an SNA connection, or the dotted decimal IP address for a TCP/IP connection (nnn.nnn.nnn.nnn).

- *number* is a resynchronization port number. This token is included only if the second token is an IP address.

#### V451: Information about incomplete resolution of indoubt threads:

Line V451 indicates that an indoubt thread was manually resolved with the RECOVER INDOUBT command, but indoubt resolution with the coordinator is not complete. The thread remains indoubt. The LUWID of the indoubt thread is displayed in the COORDINATOR field prior to this message.

You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

#### V452: Information about heuristic commit decisions:

Line V452 indicates that a decision was made at the participant to force the indoubt thread to commit without waiting for automatic resynchronization to be performed. The LUWID of the indoubt thread is displayed in the COORDINATOR field prior to this message.

#### HEURISTIC COMMIT BY PARTICIPANT AT LOCATION *text*

The location (relational database name) where the HEURISTIC COMMIT was performed.

You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

This information is displayed until DB2 resolves the indoubt status with the coordinator. If the coordinator decision is the same as the heuristic decision of the partner, the detail line is no longer displayed.

#### V453: Information about heuristic abort decisions:

Line V453 indicates that a decision was made at the participant to force the indoubt thread to abort without waiting for automatic resynchronization to be performed. The LUWID of the indoubt thread that was forced to abort is displayed in the COORDINATOR field prior to this message.

#### HEURISTIC ABORT BY PARTICIPANT AT LOCATION *text*

The location (relational database name) where the HEURISTIC ABORT was performed.

You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

This information is displayed until DB2 resolves the indoubt status with the coordinator. If the coordinator decision is the same as the heuristic decision of the partner, this detail line is no longer displayed.

If the coordinator status is COMMITTED and the heuristic decision of the participant is ABORT, the message continues to be displayed until it is deleted with the RESET INDOUBT command. The database administrators at all involved locations need to know that heuristic damage occurred and at which location.

#### V457: Information about resync protocol errors with participants:

Line V457 indicates that a protocol error was detected during attempted automatic indoubt resolution with the participant. Manual resolution of an indoubt thread at the participant location might be required. The LUWID of the thread that might require manual resolution is displayed in the COORDINATOR field prior to this message.

#### RESYNC PROTOCOL ERROR WITH PARTICIPANT *text*

The location where manual resolution might be needed. The location is either the relational database name or the logical unit name (LU name) of the participant.

You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

#### V458: Connection information for indoubt threads:

If you specify TYPE(INDOUBT) in the DISPLAY THREAD command, the output contains the following connection information :

#### CONNECTION-NAME *text*

The connection name that is assigned by the coordinator if the coordinator is another DB2 subsystem. The field contains the constant SERVER if the coordinator is not a DB2 subsystem.

#### CORRELATION-ID *text*

The correlation ID that is assigned by the coordinator if the coordinator is another DB2 subsystem. The field contains the first 12 characters of the EXTNAME parameter of the EXCSAT command that is sent by the coordinator if the coordinator is not a DB2 subsystem.

| **V459: Information about syncpoint protocol errors with coordinators:**

| Line V459 indicates that a protocol error was detected during sync point processing with the coordinator. Manual resolution of an indoubt thread at this location is required. The LUWID of the thread that requires manual resolution is displayed in the COORDINATOR field prior to this message.

| You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

| **V460: Information about syncpoint protocol errors with participants:**

| Line V460 indicates that a protocol error was detected during sync point processing with the participant. Manual resolution of an indoubt thread at the participant location might be required. The LUWID of the thread that might require manual resolution at the participant is displayed in the COORDINATOR field prior to this message.

| **SYNCPOINT PROTOCOL ERROR WITH PARTICIPANT** *text*

|       The location where manual resolution might be needed. The location is either the relational database name or the logical unit name (LU name) of the participant.

| You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

| **V461: Information about heuristic damage:**

| Line V461 indicates that indoubt resolution with the coordinator completed. Heuristic damage was detected at this location because the heuristic decision that was made at this location is different than the decision that was made at the coordinator. The thread remains indoubt. The LUWID of the thread is displayed in the COORDINATOR field prior to this message. The STATUS field shows the heuristic decision that was made at this location.

| This message is displayed until it is removed by a RESET INDOUBT command.

| You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

| **V462: Information about cold starts by participants:**

| Line V462 indicates that DB2 has coordinator responsibility for an indoubt thread at the participant, and the participant informed DB2 that it performed a cold start operation and lost all knowledge of indoubt threads. Heuristic damage might have occurred at the participant. The LUWID of the indoubt thread is displayed in the COORDINATOR field prior to this message.

| Use the RESET INDOUBT command to purge this message from the indoubt report. The cold-starting participant might have been coordinating downstream participants that are now indoubt. The database administrator at these participants might need to know the status of the thread at the coordinator, so that correct heuristic decisions can be made.

| **COLD START BY PARTICIPANT AT LOCATION** *text*

|       The location (relational database name) where the cold start was performed.

| You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

| **V463: Information about cold starts by coordinators:**

| Line V463 indicates that one of the following situations occurred:

- | • The coordinator for a thread that is indoubt at a DB2 subsystem was cold started and therefore lost all knowledge of indoubt threads.
- | • The knowledge of the local system about the coordinator was cold started, for example, by the RESET INDOUBT FORCE command.

| Manual resolution of an indoubt thread at this location is required. The LUWID of the thread for which manual resolution is required is displayed in the COORDINATOR field prior to this message.

| You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

| **V464: Information about resync protocol errors with coordinators:**

| Line V464 indicates that a protocol error was detected during attempted automatic indoubt resolution with the coordinator. Manual resolution of an indoubt thread at this location is required. The LUWID of the thread that requires manual resolution is displayed in the COORDINATOR field prior to this message.

| You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

| **V465: Information about threads that remain in the prepared state:**

| Line V465 indicates that the thread was prepared for the specified period of time. DB2 is the participant in the logical unit of work. DB2 has completed the first phase of commit and is waiting for the commit or abort decision to be transmitted from the coordinator for the specified period of time. This message is not included in the output until the thread is in the prepared state for 60 seconds. The thread is identified by the line that precedes this message.

| You must specify TYPE (ACTIVE) in the DISPLAY THREAD command for this information to be displayed.

| *timestamp*

|       The length of time that the thread was prepared, in the format *hh:mm:ss* (hours, minutes and seconds). The maximum number of reported hours is 65535, which is roughly 7.5 years. After this time period elapses, the count is reset to zero.

| You can take the following actions to try to resolve this thread manually (that is, take heuristic action) if you need to release locks that are held by this thread:

- | 1. Use the CANCEL DDF THREAD command to force the thread from the prepared state to the indoubt state. When the thread is indoubt, it is displayed in the DISPLAY THREAD TYPE(INDOUBT) report.
- | 2. Determine the proper decision (commit or abort) by contacting the coordinator location, as shown in the display of indoubt threads.
- | 3. Use the RECOVER INDOUBT command to heuristically commit or abort the thread.

| **V466: Information about how long a thread was indoubt:**

| Line V466 indicates that the thread entered the indoubt state at the specified time. DB2 is the participant in the logical unit of work, has completed the first phase of commit, and is waiting for the commit or abort decision to be transmitted from the coordinator. However, DB2 lost communications with the coordinator. The LUWID of the thread is displayed in the COORDINATOR field prior to this message.

| **THREAD HAS BEEN INDOUBT FOR** *timestamp*

|       The length of time that the thread was indoubt, in the format *hh:mm:ss* (hours, minutes and seconds). The maximum number of reported hours is 65535, which is roughly 7.5 years. After this time period elapses, the count is reset to zero.

| If necessary, you can release locks that are held by this thread by performing the following actions:

- | 1. Determine the proper decision (commit or abort) by contacting the coordinator location, as shown in the display of indoubt threads.
- | 2. Use the RECOVER INDOUBT command to heuristically commit or abort the thread.

| You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

| **V467: Information about distributed threads (for indoubt and postponed threads):**

| For indoubt and postponed distributed threads, the output includes the following additional information.

| **LUWID** *text num*

|       The global logical unit of work ID and the local token that identifies the thread.

| You must specify TYPE(INDOUBT) in the DISPLAY THREAD command for this information to be displayed.

| **V471: Information about available agents:**

| Line V471 indicates that the agent is currently not associated with a remote location. In the following situations, the agent can be assigned to a process:

- | • The completion of a queued receive request on a type 2 inactive thread.
- | • A request that a new connection be established, after MAX REMOTE ACTIVE has been reached.

| *luwid=token*

|       The global logical unit of work ID and the local token that identifies the agent.

| **V480 and V481: Information about the unit of recovery:**

| When the DISPLAY THREAD command is issued with the RRSURID option, additional information is provided about the unit of recovery.

| Information about the participant is provided in line V480 when the TYPE(ACTIVE) option is used.

| **URID** *hex*  
 |       The Recoverable Resource Services (RRS) unit of recovery ID in which this thread participates.

| Information about the coordinator is provided in line V481 when either the TYPE(ACTIVE) or the TYPE(INDOUBT) option is used.

| **URID** *hex*  
 |       The Recoverable Resource Services (RRS) unit of recovery ID for which this thread is coordinator.

| When the V481 message is included in an INDOUBT thread report, the preceding lines indicate the status of the DB2 unit of recovery that contains the RRS unit of recovery. If manual recovery of the RRS unit of recovery is required, the outcome of the DB2 unit of recovery can determine whether to commit or rollback the RRS unit of recovery.

| **V482: WLM information:**

| If you request a detail report and the thread is associated with a z/OS Workload Manager (WLM) enclave, the output includes an indication that WLM is currently applying its performance goals to the thread.

| *text*       Information about the WLM characteristics that are associated with the thread. This field includes the values of the following identifiers when known, separated by colons:

- |       • The name of the service class.
- |       • The service class period number.
- |       • The importance level of the period.
- |       • The performance index of the service class period, scaled by 100. A value of 100 corresponds to a value of 1.0 in the corresponding RMF™ report.

|       An asterisk (\*) indicates that the value is not known or is not applicable.

| **V485: Information about trusted contexts:**

| If a thread is associated with a trusted context, the output includes the trusted context name, system authorization ID, and role.

| This information is included when a non-detail report is specified.

| **TRUSTED CONTEXT** *text*  
 |       The trusted context name.

| **SYSTEM AUTHID** *text*  
 |       The system authorization ID that is associated with the user in the trusted context.

| **ROLE** *text*  
 |       The role that is associated with the user in the trusted context. If the role is not defined for the trusted context, this field contains an asterisk (\*).

| **V490: Information about suspended threads:**

| Line V490 indicates that a thread in your system is currently suspended. This message is informational only.

| *timestamp*  
 |       The suspend date and time. The format is YYDDD-HH:MM:SS.HSEC.

| *smodule soffset and smodptf-level*  
 |       Serviceability information.

| You must specify TYPE(SYSTEM) in the DISPLAY THREAD command for this information to be displayed.

| **V492: Information about storage:**

| The V492 line is an informational message for use by IBM Software Support.

| **V501: Information about commands:**

| Line V501 indicates that the system thread is currently executing the specified command. The thread is identified by the preceding lines of output.

| *text*       The command that the thread is currently running.

| **V502: Information about DB2 exit routines:**

| If a system thread is currently executing a DB2 exit routine that you specified, the output includes the following additional information about the thread. The thread is identified by the preceding lines of output.

| *text*      The name of the DB2 exit routine that the thread is running.

| **RMID=***text*, **FCODE=***text text*  
|      Serviceability information.

|      **RMID=***dec*  
|      Resource manager identifier (RMID).

|      **FCODE=***text*  
|      Feature code.

|      *text*      A text description, if available.

### | **V504: Information about threads that are terminating:**

| If a system thread is currently terminating another thread, the output includes the following additional information about the thread. The thread is identified by the preceding lines of output.

| **CORRID=***text*  
|      The specified correlation ID of the thread that is being terminated.

| **CONNID=***text*  
|      The connection ID of the thread that is being terminated.

| **HASID=***text*  
|      The home address space ID of the thread that is being terminated.

### | **V505: Information about threads that are waiting:**

| Line V505 indicates that a system thread is experiencing a long wait time for reading or writing data for the specified object. The thread is identified by the preceding lines of output.

| **DBNAME=***text*  
|      The name of the database.

| **SPACE NAME=** *text*  
|      The name of the table space or index space.

| **PART=** *num*  
|      A number that identifies the partition. For non-partitioned data sets, no value is displayed.

| More detailed information about the suspended thread is reported in line V490.

### | **V507: Information about monitoring tasks:**

| If the thread is currently an internal DB2 monitoring task, the output includes the following additional information about the thread.

| *text* **MONITOR**  
|      The type of monitoring task: active or inactive.

| *monintervals, monstg, monboosts, and text*  
|      Serviceability information.

| **REGION** *num*  
|      The size of the region, as a number and unit of measurement.

| **AVAIL** *num*  
|      The amount of available storage, as a number and unit of measurement.

| **CUSHION** *num*  
|      The size of the storage cushion, as a number and unit of measurement. The storage cushion is a defined portion of total storage that is available for use, but whose use triggers alerts. The purpose of the alerts is to notify administrators of unexpected storage requirements and help them manage the condition.

| You must specify DETAIL and TYPE(SYSTEM) in the DISPLAY THREAD command for this information to be displayed.



**Information that indicates NOT FOUND situations:**

If no threads or connections of the requested type are found, one of the following messages is displayed.

**NO CONNECTIONS FOUND**

The DISPLAY THREAD TYPE(\*), TYPE(ACTIVE) or TYPE(INACTIVE) command was unable to locate any connections within the subsystem.

**NO CONNECTION FOUND FOR NAME=*text***

The DISPLAY THREAD TYPE(ACTIVE) or TYPE(INACTIVE) command was unable to locate any connections that are associated with the indicated connection name. This message is generated once for each connection name for which no connection was found.

**NO THREADS FOUND FOR NAME=*text***

The DISPLAY THREAD TYPE(ACTIVE) or TYPE(INACTIVE) command was unable to locate any threads that are associated with the indicated connection name. This message is generated once for each connection name for which no threads were found.

**NO SYSTEM THREADS FOUND**

The DISPLAY THREAD command did not find any system threads that can be displayed.

**NO SYSTEM THREADS FOUND FOR NAME= *connection-name***

The DISPLAY THREAD command did not find any system threads for the named connection.

**NO INDOUBT THREADS FOUND**

The DISPLAY THREAD command found no indoubt threads within the system. This message is generated when the command requests information about all (\*) threads that are currently residing within an indoubt state (TYPE=INDOUBT) within the system, but no threads currently exist within this state.

*csect* **NO INDOUBT THREADS FOUND FOR NAME=*text***

The DISPLAY THREAD TYPE(INDOUBT) command was unable to locate any indoubt threads that are associated with the indicated connection name. This message is generated once for each connection name for which no indoubt threads were found.

**NO POSTPONED THREADS FOUND FOR NAME=*text***

The DISPLAY THREAD command found no postponed abort URs for the named connection. This message is generated when the command requests information about threads for the given connection name that are currently residing within a postponed abort state, but no threads currently exist within this state.

**DSNV434I *csect* NO POSTPONED ABORT THREADS FOUND**

The DISPLAY THREAD TYPE(POSTPONED) command found no postponed abort threads.

**Other information reported in the DISPLAY THREAD output:**

**ARCHIVE LOG QUIESCE CURRENTLY ACTIVE**

An ARCHIVE LOG MODE(QUIESCE) command is currently active. Updates against DB2 resources have been temporarily suspended, which might result in active threads being suspended until termination of the quiesce period.

**SCOPE(GROUP) SUPPRESSED DURING RESTART AND SHUTDOWN**

The DISPLAY THREAD command with the SCOPE(GROUP) option was issued during restart or shutdown. SCOPE(GROUP) is not supported during restart or shutdown. The command runs locally.

**ONLY SYSTEM THREAD DATA AVAILABLE**

The DISPLAY THREAD command was issued during restart or shutdown and specified a value for the TYPE option other than SYSTEM. This message indicates that only system thread information is available during restart and shutdown.

**End of report markers:**

The DISPLAY THREAD output normally ends with message DSN9022I.

If the command is not able to complete the report, the output ends with message DSN9023I and one of the following explanations:

**DISPLAY THREAD MESSAGE POOL SIZE EXCEEDED**

The amount of CSA or ECSA virtual storage that is needed to generate DISPLAY THREAD output exceeded the maximum size of the message buffer pool.

This situation can happen when many threads exist in DB2, and the pattern-matching character '\*' is used in a DISPLAY THREAD command, as shown in the following example:

```
-DISPLAY THREAD(*)
```

In this case, processing is terminated.

To correct the error, reduce the amount of CSA or ECSA virtual storage that is required for the DISPLAY THREAD command by performing one of the following actions:

- Specify qualifiers, such as TYPE(INDOUBT), TYPE(ACTIVE), or TYPE(INACTIVE).
- Specify specific *connection-name*, *location-name*, or *luwid* values.

#### MESSAGE LIMIT EXCEEDED. DISPLAY IS TRUNCATED.

The number of DISPLAY THREAD command messages exceeds the maximum number of permitted messages. The maximum number of messages is specified by the LIMIT option or the default value for the LIMIT option.

#### DISPLAY THREAD OUTPUT TO BE CONTINUED

DISPLAY THREAD command output directed to an MVS console must be continued in another group of messages. The continuation notice contains a unique command ID that is also displayed in the first line of the continuation.

Compare the command ID values to confirm that the continuation is from the same report.

#### Error messages:

If the DISPLAY THREAD command fails, you receive one of the following messages:

*csect-name* **DISPLAY THREAD***pkwname* **TERMINATED, DEFAULT UNAVAILABLE**

The DISPLAY THREAD command statement did not contain all required parameters. Command processing terminates.

*csect-name*

The name of the control section that issued the message.

**pkwname**

The command primary keyword as entered by the requester.

One of the following items was not specified: *connection-name*, LOCATION, or LUWID.

Reenter the command with one of these parameters specified.

#### LOCATION OR LUWID KEYWORD INVALID WHEN DDF NOT STARTED

The DISPLAY THREAD TYPE(ACTIVE) command was issued with the keyword LOCATION or LUWID, but the distributed data facility (DDF) was not started. The LOCATION or LUWID keyword can be used only if DDF is in operation.

Command processing is complete.

#### INVALID PARAMETER WITH LUWID KEYWORD *luwid-parameter*

The DISPLAY THREAD TYPE(ACTIVE) or DISPLAY THREAD TYPE(INACTIVE) command was issued with the keyword LUWID and its required parameter list. However, one of the parameters was syntactically invalid as a logical unit of work identifier.

The command returns without processing.

Reenter the command with valid LUWID values.

#### SCOPE(GROUP) DISPLAY TERMINATED DUE TO IRLM TRUNCATION

The SCOPE(GROUP) keyword was specified, and IRLM truncated the responses gathered from other members. This error is a DB2 internal error.

The DISPLAY THREAD report is terminated and processing abnormally terminates.

Notify the system programmer and contact IBM Software Support to report the problem.

#### SCOPE(GROUP) DATA UNAVAILABLE



The SCOPE(GROUP) keyword was specified, and the IRLM NOTIFY to other members has returned a bad return code.

*retcode* Return code from IRLM on the Notify request.

*reason* Reason code diagnostic information from IRLM on the Notify request.

**System action:** Processing continues.

**Operator response:** Review the information that is provided for each line of output and take any necessary action.

**Related concepts:**

 TCP/IP terminology (DB2 Installation and Migration)

 Using service class periods (System Programmer's Guide to: Workload Manager)

 Using importance levels (System Programmer's Guide to: Workload Manager)

 Performance index (System Programmer's Guide to: Workload Manager)

**Related reference:**


Chapter 32, “-DISPLAY LOCATION (DB2),” on page 241

Chapter 37, “-DISPLAY THREAD (DB2),” on page 259

Chapter 69, “-RESET INDOUBT (DB2),” on page 433

 Performance Index (MVS Programming: Workload Management Services)

**Related information:**

 DSN3201I (DB2 Messages)

 DSN9022I (DB2 Messages)

 DSN9023I (DB2 Messages)

 Defining Service Classes and Performance Goals (z/OS MVS Planning: Workload Management)

 Setting up a Service Definition (z/OS MVS Planning: Workload Management)

 Using Performance Periods (z/OS MVS Planning: Workload Management)



---

## 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.

This command is also available in conversion mode.

**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.

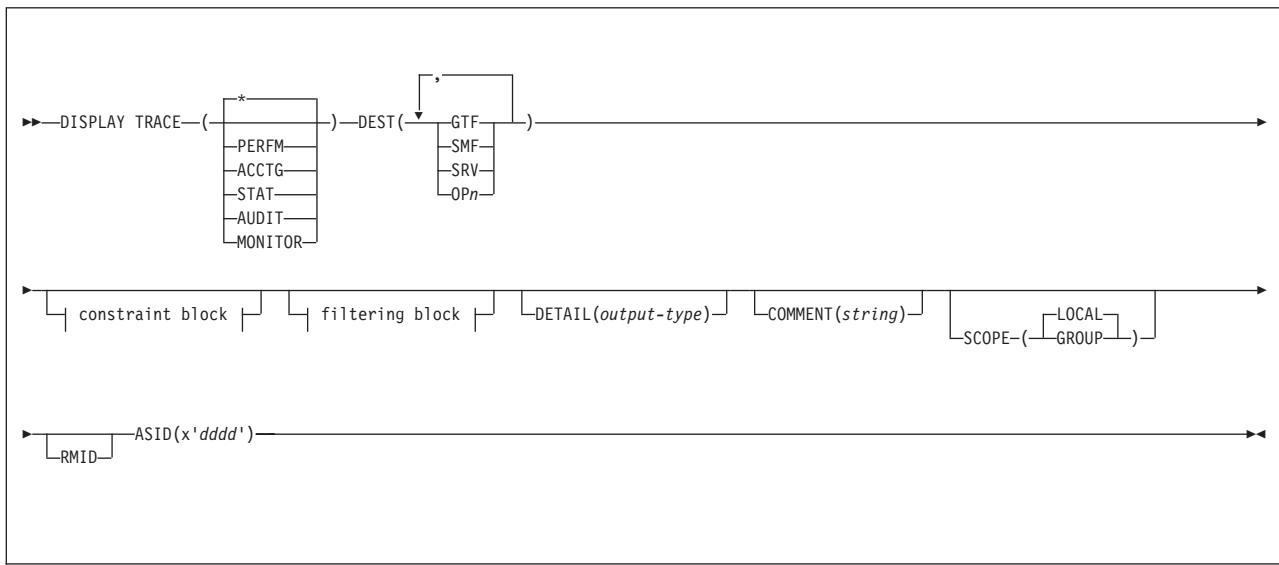
### Authorization

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

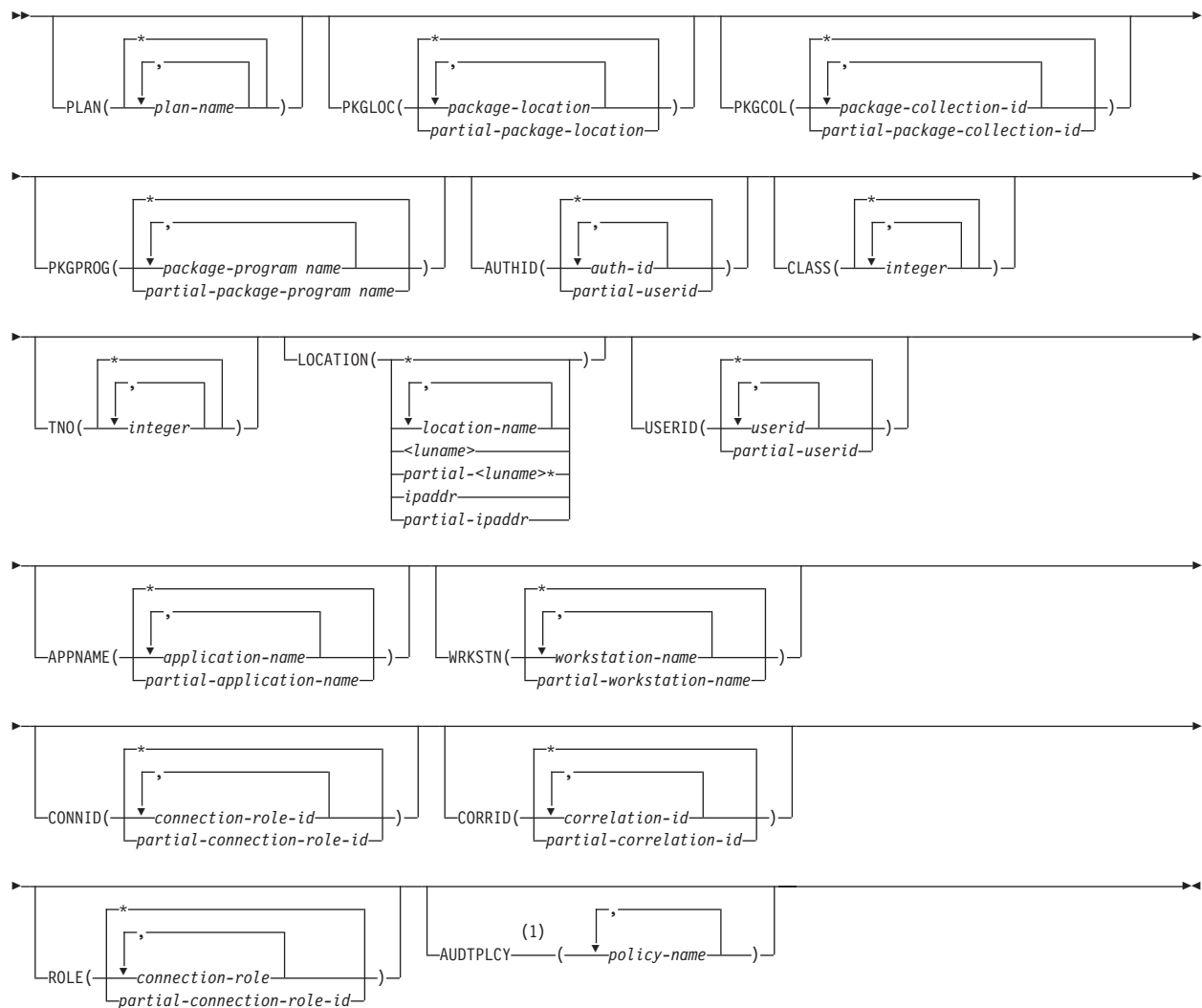
- DISPLAY privilege
- SQLADM authority
- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority
- SECADM authority

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



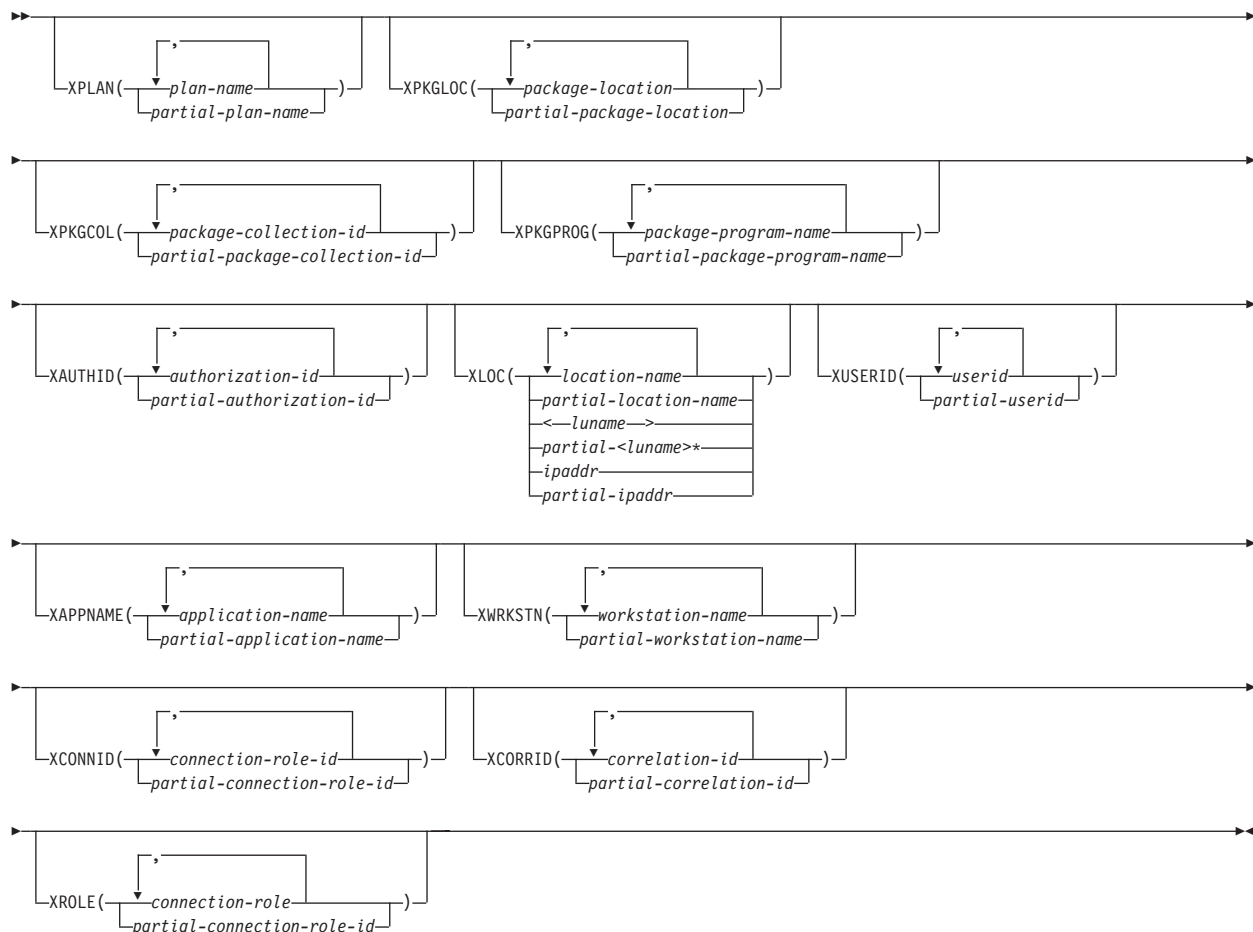
## constraint block



### Notes:

- 1 AUDTPLCY applies to trace type AUDIT. You cannot specify CLASS or IFCID with AUDTPLCY.

## filtering block



## 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 <i>n</i>	A specific destination. <i>n</i> 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.

**APPNAME or XAPPNAME**

Specifies the application name. Use APPNAME to constrain the trace to the specified applications or XAPPNAME to exclude the specified applications.

**WRKSTN or XWRKSTN**

Specifies the workstation name. Use WRKSTN to constrain the trace to the specified workstations or XWRKSTN to exclude the specified workstations.

#### **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 utilize 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 (ADBIOP, 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.

## 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:**

Chapter 99, “-STOP TRACE (DB2),” on page 549

Chapter 87, “-START TRACE (DB2),” on page 497

Chapter 60, “-MODIFY TRACE (DB2),” on page 401



---

## 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

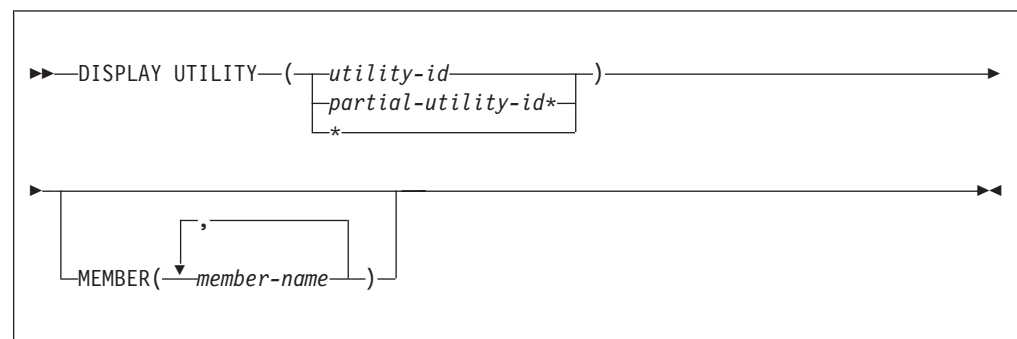
This command can be issued from a z/OS console, a DSN session, a DB2I panel (DB2 COMMANDS), an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

**Data sharing scope:** Group or member, depending on which option you choose

### 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

The output from the DISPLAY UTILITY command consists of informational messages only.

**Output during any phase of REORG with SHRLEVEL CHANGE or SHRLEVEL REFERENCE:** During *any* phase of REORG with SHRLEVEL CHANGE or SHRLEVEL REFERENCE, the output of DISPLAY UTILITY includes the information in DSNU347I. During *any* phase of REORG with SHRLEVEL CHANGE, the output of DISPLAY UTILITY includes information in DSNU384I.

### DEADLINE

Indicates a timestamp according to the value of DEADLINE that is currently in effect.



**MAXRO**

Indicates the number of seconds, according to the value of MAXRO that is currently in effect.

**LONGLOG**

Indicates either CONTINUE, TERM, or DRAIN according to the value of LONGLOG that is currently in effect.

**DELAY**

Indicates the number of seconds according to the value of DELAY that is currently in effect.

*Output during LOG phase of REORG with SHRLEVEL CHANGE:* During the LOG phase of REORG with SHRLEVEL CHANGE, the output of DISPLAY UTILITY includes the additional information found in message DSNU383I.

**CURRENT ITERATION NUMBER**

Indicates the current iteration number.

**WRITE ACCESS ALLOWED IN CURRENT ITERATION**

Indicates "YES" or "NO" according to whether write access is allowed in the current iteration of log processing.

**ITERATION BEFORE PREVIOUS ITERATION**

Indicates the ELAPSED TIME so far, and the NUMBER OF LOG RECORDS PROCESSED in the iteration. Their value is zero if the current iteration number is one or two.

**PREVIOUS ITERATION**

Indicates the ELAPSED TIME and the NUMBER OF LOG RECORDS PROCESSED for the previous iteration. Their value is zero if the current iteration number is one.

**CURRENT ITERATION:**

Indicates the ESTIMATED ELAPSED TIME, the ACTUAL ELAPSED TIME SO FAR and the ACTUAL NUMBER OF LOG RECORDS BEING PROCESSED.

**CURRENT ESTIMATE FOR NEXT ITERATION**

For the next iteration, indicates the currently ELAPSED TIME and the currently estimated NUMBER OF LOG RECORDS TO BE PROCESSED.

*Output during UNLOAD phase of REORG with SHRLEVEL CHANGE or SHRLEVEL REFERENCE:* During the UNLOAD phase of REORG with SHRLEVEL CHANGE or SHRLEVEL REFERENCE, DSNU111I messages are issued for the following subtasks: unloading nonpartitioned indexes, building shadow nonpartitioned indexes, sort, build, and inline statistics. The phase for subtasks for unloading nonpartitioned indexes is UNLOADIX. The phase for subtasks during the LOG phase is LOGAPPLY.

*Progress of utility processing:* The DISPLAY UTILITY command provides an estimate of how much processing the utility has completed. The output displays information from message DSNU105I, and includes the following information:

**COUNT**

COUNT *n* is the number of pages or records processed in a utility phase. COUNT has different meanings for different utilities. For utilities not mentioned in the following list, ignore this field.

- For the LOAD utility, COUNT represents the total number of records that have been loaded into all partitions when the command is issued.

The count is zero from the time the RELOAD phase starts until the first LOAD subtask begins loading records into the first partition assigned to that subtask.

- For the CHECK INDEX, RECOVER INDEX, and REORG utilities, COUNT represents the number of records processed.
- For the COPY, MERGE COPY, RECOVER (restore phase), and RUNSTATS utilities, COUNT represents the number of pages processed.
- For the STOSPACE utility, COUNT represents the number of table spaces or indexes processed.

**Progress of the RECOVER utility:** During the LOGAPPLY phase, you can use the DISPLAY UTILITY command to check the progress status of the RECOVER utility. Message DSNU116I provides an estimate of the log processing that has been completed for the recovery job. This function is available in conversion mode.

## 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.

The TSO command DSN enables you to issue the following DSN subcommands:

- ABEND
- BIND
- DCLGEN
- END
- FREE
- REBIND
- RUN
- SPUFI

During a DSN session, you can enter DB2 commands or comments. DB2 commands must start with a hyphen (-). Comments must start 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
```

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.

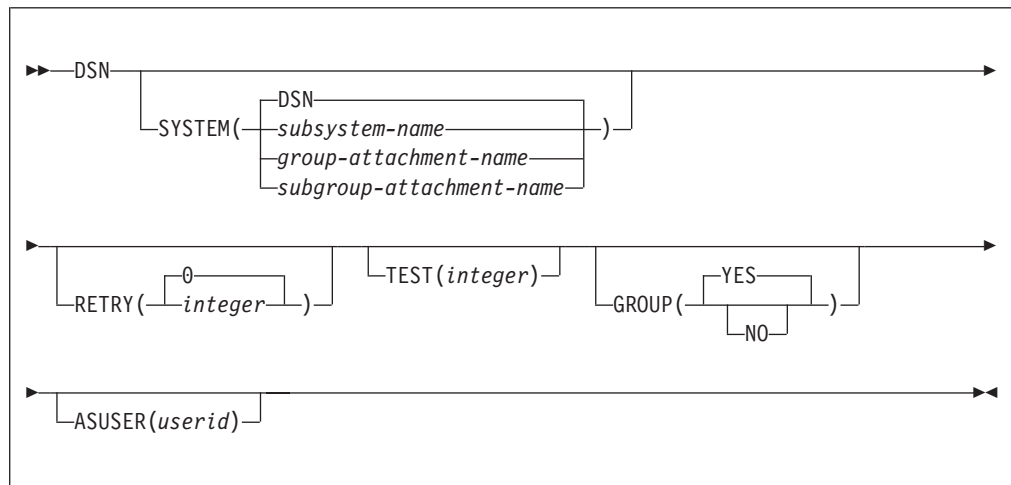
You can also start a DSN session from a CLIST running in either foreground or background mode.

**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 doing one of the following:
  - 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(SSTR)
DSN prompt : DSN
USER enters: RUN PROGRAM(MYPROG)
DSN prompt : DSN
USER enters: END
TSO prompt : READY
```



---

## Chapter 41. DSNC (CICS attachment facility)

The CICS attachment facility DSNC command allows you to enter DB2 commands from CICS.

### Environment

This command can be issued only from a CICS terminal.

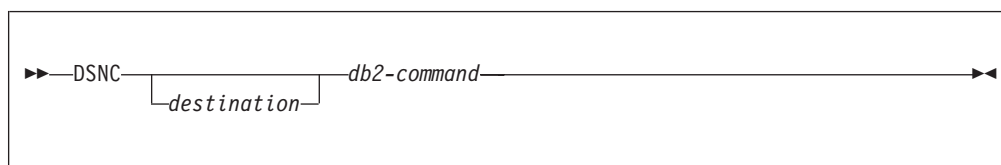
**Data sharing scope:** Member

### Authorization

This command requires the appropriate level of CICS authority.

Entering the DSNC command requires no privileges from DB2 security. For a description of the privileges required to issue a DB2 command using the DSNC command, see the command's description.

### Syntax



### Option descriptions

#### *destination*

Identifies another terminal to receive display information. It must be a valid terminal that is defined to CICS and supported by CICS basic mapping support (BMS).

#### *db2-command*

Specifies the exact DB2 command that you want to enter from a CICS terminal. It must be preceded by a hyphen.

### Usage note

**Screen scrolling:** The CICS SIT table keyword SKRxxxx can be used to support the scrolling of DSNC DB2 commands from your terminal.

### Examples

#### **Example: Displaying thread information from a CICS terminal**

The following command issues the DB2 command DISPLAY THREAD from a CICS terminal.

```
DSNC -DISPLAY THREAD
```



---

## Chapter 42. DSNC DISCONNECT (CICS attachment facility)

The CICS attachment facility command DSNC DISCONNECT disconnects threads.

The command provides manual control to release resources being shared by normal transactions so that special purpose processes, such as utilities, can have exclusive access to the resources.

**Abbreviation:** DSNC DISC

### Environment

This command can be issued only from a CICS terminal.

**Data sharing scope:** Member

### Authorization

This command requires an appropriate level of CICS authority.

### Syntax

▶▶—DSNC DISCONNECT—*plan-name*————▶▶

### Option description

*plan-name*

Specifies a valid application plan.

### Usage notes

**Preventing creation of threads:** The command DSNC DISCONNECT does not prevent threads from being created on behalf of transactions. The command only causes currently connected threads to be terminated as soon as they are not being used by a transaction. To interrupt a transaction and cancel a thread faster, you can use the DB2 command CANCEL THREAD.

You can stop the transactions associated with a particular plan ID in CICS with the MAXACTIVE setting for TRANCLASS. This prevents new instances of the transaction from causing a re-creation of a thread.

**Alternative for protected threads:** You might want to deallocate a plan for rebinding or for running a utility against the database. If you are using a protected thread, use DSNC MODIFY rather than DSNC DISCONNECT. Modify the THRDA value of the plan to zero to send all the threads to the pool. The protected thread will terminate on its own within 60 seconds and DISCONNECT is unnecessary.

## Examples

### Example: Disconnecting active threads for a plan

The following command disconnects active threads for plan PLAN1.

```
DSNC DISCONNECT PLAN1
```

---

## Chapter 43. DSNB DISPLAY (CICS attachment facility)

The CICS attachment facility command DSNB DISPLAY displays information on CICS transactions accessing DB2 data, or statistical information associated with DB2ENTRYs and the DB2CONN.

**Abbreviation:** DSNB DISP

### Environment

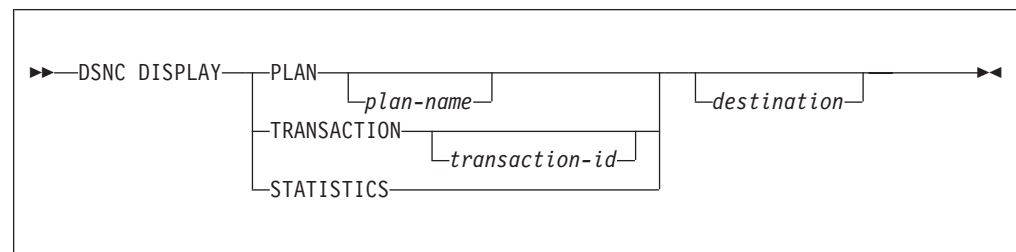
This command can be issued only from a CICS terminal.

**Data sharing scope:** Member

### Authorization

This command requires an appropriate level of CICS authority.

### Syntax



### Option descriptions

#### **PLAN** *plan-name*

Displays information about transactions by plan name.

*plan-name* is a valid plan name for which information is displayed.

**Default:** If you do not specify *plan-name* (or if you specify an asterisk, \*), information is displayed for all active transactions.

#### **TRANSACTION** *transaction-id*

Displays information about transactions by transaction ID.

**Abbreviation:** TRAN

*transaction-id* is a valid transaction ID for which information is displayed.

**Default:** If you do not specify a transaction ID, information is displayed for all active transactions.

#### **STATISTICS**

Displays one line of the statistical counters that are associated with each DB2ENTRY. The counters correspond to the usage of the available connections of the CICS attachment facility to DB2.

**Abbreviation:** STAT

If you issue this command from CICS while the CICS attachment facility is active but the DB2 subsystem is not, CICS produces a statistics display with no obvious indication that the DB2 subsystem is not operational. Message DFHDB2037 appears in the CICS message log to indicate that the attachment facility is waiting for DB2 to start.

*destination*

Specifies the identifier of another terminal that is to receive the requested display information. It must be a valid terminal that is defined to CICS and supported by CICS basic mapping support (BMS).

## Usage notes

**Entering parameters:** Because the optional destination is sometimes preceded by an optional plan name or transaction ID in the command, each parameter must be unique and separately identifiable as either a name or a terminal identifier. If only one parameter is entered, it is first checked to see whether it is a plan name or a transaction ID, and it is then checked as a destination. To use a character string that is both a plan name or transaction ID and also a valid terminal identifier, you must use both the name and destination parameters to display the information at the terminal that you want.

**Acknowledging display information sent to an alternative destination:** When an alternative destination is specified to receive the requested display information, the following message is sent to the requesting terminal:

DFHDB2032 THE DISPLAY COMMAND IS COMPLETE

## Output

For each created thread, the output for the DSNCR DISPLAY (PLAN or TRANSACTION) command displays the following information:

**DB2ENTRY**

The name of the DB2ENTRY, \*POOL for the pool, or \*COMMAND for DSNCR command calls.

**S** The status field. A status of A indicates the thread is active within a unit of work. A status of I indicates that a protected thread is waiting for work.

**PLAN** The plan that is associated with the thread. Command threads have no plan.

**PRI -AUTH**

The primary authorization ID for the thread.

**SEC -AUTH**

The secondary authorization ID (if any) for the thread.

**CORRELATION**

The 12-byte thread correlation ID in the form of *eeeeetttnnnn*, where *eeee* is either COMD, POOL, or ENTR, indicating a command, pool, or DB2ENTRY thread; *tttt* is the transaction ID; *nnnn* is a unique number.

If the thread is active within a unit of work, its CICS transaction name (TRAN), task number (TASK), and CICS local unit of work ID (UOW-ID) are also displayed.

The output of a DSNCR DISPLAY STATISTICS command displays the following information:

**DB2ENTRY**

The name of the DB2ENTRY, \*COMMAND for DSNB command calls, or \*POOL for pool statistics.

**PLAN** The plan name that is associated with this entry. Eight asterisks in this field indicate that this transaction is using dynamic plan allocation. The command processor transaction DSNB does not have a plan associated with it because it uses a command processor.

**CALLS**

The total number of SQL statements that are issued by transactions that are associated with this entry.

**AUTHS**

The total number of sign-on invocations for transactions associated with this entry. A sign-on does not indicate whether a new thread is created or an existing thread is reused. If the thread is reused, a sign-on occurs only if the authorization ID or transaction ID has changed.

**W/P** The number of times that all available threads for this entry were busy. This value depends on the value of THREADWAIT for the entry.

An overflow to the pool is displayed in the transaction statistics only and is not reflected in the pool statistics.

If THREADWAIT is set to YES, the output reflects the number of times that the thread had to wait. If the number of started tasks has reached THREADLIMIT, the output also reflects the number of times the thread could not attach a new subtask.

The only time W/P is updated for the pool is when a transaction had to wait for a pool thread and a new subtask could not be attached for the pool. The W/P statistic is useful for determining if a sufficient number of threads are defined for the entry.

**HIGH** The maximum number of threads that are required by transactions that are associated with this entry at any time since the connection was started. It provides a basis for setting the maximum number of threads for the entry. For releases of CICS Transaction Server before release 1.2, this number includes the transactions that were forced to wait or diverted to the pool. For release 1.2 or later, the HIGH keyword is associated only with the threads that are actually created on the entry.

**ABORTS**

The total number of units of recovery that were rolled back. It includes both abends and SYNCPOINT ROLLBACKS, including SYNCPOINT ROLLBACKS generated by -911 SQL codes.

**COMMITTS**

One of the following two fields increments each time a DB2 transaction that is associated with this entry has a real or implied (such as EOT) syncpoint. Units of recovery that do not process SQL calls are not reflected here.

**1-PHASE**

The total number of single phase commits for transactions that are associated with this entry. This total does not include any two-phase commits (see the explanation for 2-PHASE). This total does include read-only commits and single-phase commits for units of recovery that have performed updates. A two-phase commit is needed only when CICS is the recovery coordinator for more than one resource manager.

## 2-PHASE

The total number of two-phase commits for transactions that are associated with this entry. This number does not include one-phase commit transactions.

## Examples

### Example: Sending information about all active threads to a designated terminal

The following command displays information about all active threads, and sends the output to a terminal that is designated as MT02.

```
DSNC DISP PLAN * MT02
```

The output is similar to the following output:

```
IYK4Z2G1 DISPLAY REPORT FOLLOWS FOR THREADS
ACCESSING DB2 DB3A

DB2ENTRY S PLAN      PRI-AUTH SEC-AUTH CORRELATION  TRAN TASK  UOW-ID
*POOL    A TESTC05   JTILLI1          POOLXC050001 XC05 01208 AEEEC03-1ACDCE00
XC06     A TESTC06   JTILLI1          ENTRXC060003 XC06 01215 AEEEC04-2F8EFE01
XP05     A TESTP05   JTILLI1          ENTRXP050002 XP05 01209 AEEEC03-35230C00
XP05     I TESTP05   JTILLI1          ENTRXP050004
DFHDB2020 07/09/98 15:26:47 IYK4Z2G1 THE DISPLAY COMMAND IS COMPLETE.
```

### Example: Displaying statistics counters that show usage of the available connections of the CICS attachment facility to DB2

The following commands displays statistical counters that are associated with each DB2ENTRY.

```
DSNC DISP STAT
```

The output is similar to the following output:

```
IYK4Z2G1 STATISTICS REPORT FOLLOWS

DB2ENTRY PLAN      CALLS  AUTHS  W/P  HIGH  ABORTS  -----COMMIT-----
*COMMAND                                1-PHASE  2-PHASE
*POOL    POOL              0      0      0    0      0          0          0
XC01     DSNXC01          22      1     11    2      0          7          5
XC02     DSNXC02           0      0      0    0      0          0          0
XA81     DSNA81           0      0      0    0      0          0          0
XCD4     DSNCED4           0      0      0    0      0          0          0
XP03     DSNTPO3           1      1      0    1      0          1          0
XA20     DSNTA20           1      1      0    1      0          0          1
XA88     *****          0      0      0    0      0          0          0
DFHDB2020 07/09/98 15:45:27 IYK4Z2G1 THE DISPLAY COMMAND IS COMPLETE
```



---

## Chapter 44. DSNC MODIFY (CICS attachment facility)

The CICS attachment facility command DSNC MODIFY modifies the message queue destination of the DB2CONN, or modifies the maximum active thread value for the pool, for DSNC commands, or for DB2ENTRY.

**Abbreviation:** DSNC MODI

### Environment

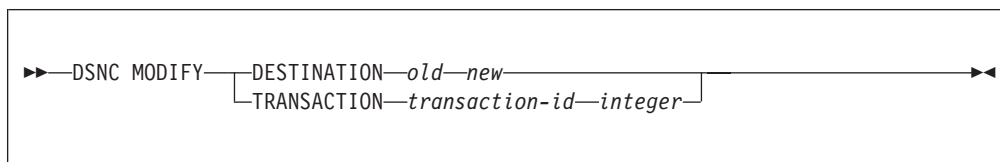
This command can be issued only from a CICS terminal.

**Data sharing scope:** Member

### Authorization

This command requires an appropriate level of CICS authority.

### Syntax



### Option descriptions

#### DESTINATION

Specifies that the MSGQUEUE parameter of the DB2CONN is to be changed, replacing the *old* destination ID with the *new* destination ID.

**Abbreviation:** DEST

*old*

Specifies any destination ID that is currently active in the MSGQUEUE of the DB2CONN.

*new*

Specifies a new destination identifier. CICS verifies the new destination to ensure that it is an existing transient data entry in the destination control table.

#### TRANSACTION

Specifies that the maximum active thread value that is associated with the specified transaction or group is to be modified.

**Abbreviation:** TRAN

*transaction-id*

Specifies a valid transaction identifier.

To change the maximum active thread value, use one of the following transaction IDs:

- For the pool: CEPL

- For command threads: DSNC
- For DB2ENTRY: the ID of any transaction that is defined to use DB2ENTRY

*integer*

Specifies a new maximum value.

## Usage notes

**Protected threads:** If you increase the active thread value by using the command DSNC MODIFY TRANSACTION, the attributes of the DB2ENTRY are used.

Issuing DSNC MODIFY TRANSACTION to increase the total number of threads that are permitted allows creation of unprotected threads. For example, assume PROTECTNUM(2) and THREADLIM(2). If the total number of permitted threads increases, the additional threads are unprotected.

The command DSNC MODIFY TRANSACTION can also allow creation of protected threads. If PROTECTNUM(2) and THREADLIM(2) and you modify the thread limit to 1, one of the protected threads is eliminated. If the thread limit is then modified back to 2, the thread that is re-created is protected.

**TRANSACTION thread limit:** The lowest possible value is zero.

### Example: Changing the destination for the message queue

The following command changes the message queue destination in the DB2CONN definition from MTO1 to MTO2.

```
DSNC MODIFY DESTINATION MT01 MT02
```

### Example: Changing the pool thread limit

The following command changes the thread limit for the pool (CEPL) to 12.

```
DSNC MODIFY TRANSACTION CEPL 12
```

### Example: Changing the command thread limit

The following command changes the thread limit for commands (DSNC) to 3.

```
DSNC MODIFY TRANSACTION DSNC 3
```

### Example: Changing the DB2ENTRY thread limit

The following command changes the thread limit for the DB2ENTRY that transaction XP05 uses to 8.

```
DSNC MODIFY TRANSACTION XP05 8
```

---

## Chapter 45. DSNC STOP (CICS attachment facility)

The CICS attachment facility command DSNC STOP stops the attachment facility.

### Environment

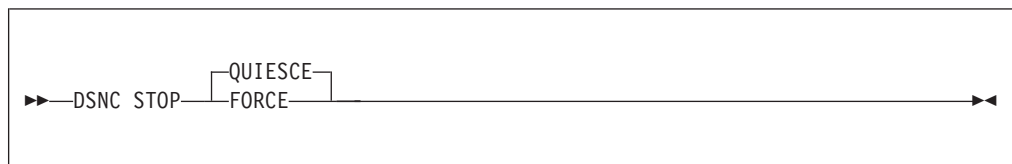
This command can be issued only from a CICS terminal.

**Data sharing scope:** Member

### Authorization

This command requires an appropriate level of CICS authority.

### Syntax



### Option descriptions

#### QUIESCE

Specifies that the CICS attachment facility is to be stopped after CICS transactions that are currently running terminate.

**Abbreviation:** Q

#### **FORCE**

Specifies that the CICS attachment facility is to be stopped immediately by forcing disconnection with DB2, regardless of any transactions that are running.

### Usage notes

**Requirements for restarting:** Using FORCE can leave threads in an indoubt situation. Restarting requires reconnection of CICS and DB2 to resolve any indoubt situations. In a data sharing environment, resolution of indoubt situations requires that the CICS be reconnected to the same DB2 member.

**Output destinations:** Output from the command DSNC STOP is sent to the requesting terminal, which remains locked until shutdown is complete.

### Examples

#### **Example: Stopping the CICS attachment facility**

The following command stops the CICS attachment facility immediately, and forces disconnection from DB2.

```
DSNC STOP FORCE
```



---

## Chapter 46. DSNC STRT (CICS attachment facility)

The DSNC STRT command starts the CICS attachment facility, which allows CICS application programs to access DB2 databases.

### Environment

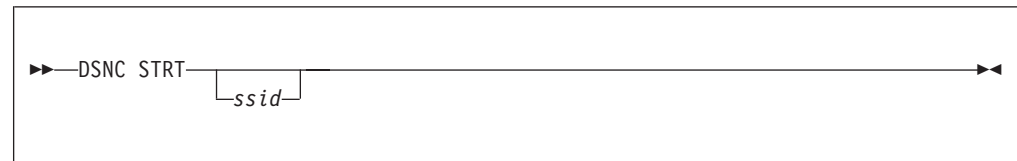
This command can be issued only from a CICS terminal.

**Data sharing scope:** Member

### Authorization

This command requires an appropriate level of CICS authority.

### Syntax



### Option description

*ssid*

Specifies the subsystem ID (SSID) that is to override the ID that is specified in the CICS DB2CONN.

**Default:** The DB2ID that is specified in the last installed DB2CONN. If the DB2CONN contains a blank DB2ID, the default is the SSID that is specified in the CICS INTIPARM parameter.

### Usage note

**Output destinations:** Output from the DSNC START command is sent to the requesting terminal. If no DB2CONN is installed when you issue DSNC STRT, error message DFHDB2031 is sent to the terminal.

### Examples

**Example: Starting the CICS attachment facility for the default DB2 subsystem**

The following command starts the CICS attachment facility for the DB2 subsystem that is specified in the DB2CONN definition.

```
DSNC STRT
```

**Example: Starting the CICS attachment facility for a specified DB2 subsystem**

The following command starts the CICS attachment facility for DB2 subsystem DB2P.

```
DSNC STRT DB2P
```



---

## Chapter 47. 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 16. 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
<b>Note:</b> The step names are used in the heading of Table 18 on page 331.	

Subsections:

- “Environment”
- “Authorization” on page 330
- “Syntax” on page 330
- “Usage notes” on page 353
- “Examples” on page 354

### 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.

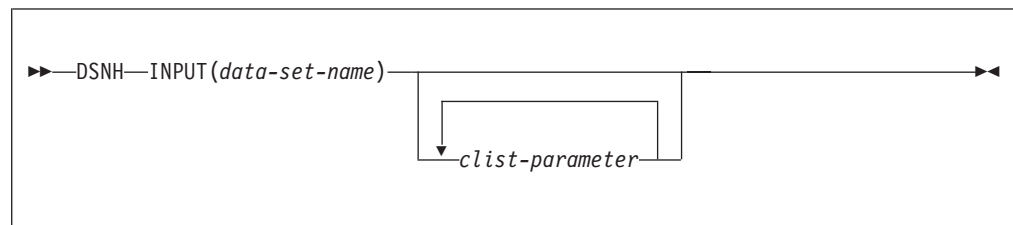
**Data sharing scope:** Member

## 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 18 on page 331. 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 17. 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 16 on page 329 for an explanation of the two-letter step names.

*Table 18. 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 18. 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 18. 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 18. 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
SOMDLI						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 19 on page 335. For a summary of:

- BIND PLAN parameters, refer to Table 20 on page 348
- BIND PACKAGE parameters, refer to Table 21 on page 351

The only parameter that is required on the DSNH statement is INPUT; the others are optional. In Table 19 on page 335:

- 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 19. 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 <b>default</b> 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 <b>default</b> 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 <b>default</b> 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 <b>default</b> 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 <b>default</b> is "'CBC.SCCNCMP(CCNDVR)'".
CCMSGSGS	<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 <b>default</b> 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.
CCPMSGSGS	<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 <b>default</b> is "'CEE.SCEEMSGP(EDCPMSG)'".
CCSLIB	<i>dsname</i>	Specifies a data set that contains the C compiler headers.  The <b>default</b> is "'CEE.SCEEH.H'".

Table 19. 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 <b>default</b> , 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 <b>default</b> is blank.
CICSLLIB	<i>dsname</i>	Specifies the CICS load library. To use this library, leave the CICSPRE parameter blank.  The <b>default</b> 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 <b>default</b> 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 <b>default</b> 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 <b>default</b> is YES. The DB2I panel <b>default</b> 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 <b>default</b> , NONE, to specify no data set.
COBICOMP	<i>dsname</i>	Specifies the IBM COBOL data set that is required for compilation.  The <b>default</b> is "IGY.SIGYCOMP".
COBILINK	<i>dsname</i>	Specifies the IBM COBOL data set that is required for link edit.  The <b>default</b> 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 <b>default</b> is "CEE.SCEERUN(EDCPRLK)".

Table 19. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
COBIPMSG	<i>dsname</i>	<p>Specifies a data set that contains the message library that is to be used by the IBM prelink editor when preparing COBOL programs.</p> <p><i>dsname</i> can include a member name.</p> <p>The <b>default</b> is "CEE.SCEEMSGP(EDCPMSGE)".</p>
COBLIB	<i>dsname</i>	<p>Specifies the linkage editor include library that is to be used for OS/VS COBOL routines.</p> <p>This parameter is obsolete.</p>
COBLOAD	<i>dsname</i>	<p>Specifies a data set that contains the OS/VS COBOL compiler load module.</p> <p>This parameter is obsolete.</p>
COBSOM	<i>dsname</i>	<p>Specifies the IBM System Object Model (SOM) data set that is required for access to SOM objects.</p> <p>This parameter is obsolete.</p>
COB2CICS	<i>dsname</i>	<p>Specifies the linkage editor include library that is to be used for VS COBOL II CICS routines.</p> <p>This parameter is obsolete.</p>
COB2LIB	<i>dsname</i>	<p>Specifies the linkage editor include library that is to be used for the VS COBOL II or COBOL/370 routines.</p> <p>This parameter is obsolete.</p>
COB2LOAD	<i>dsname</i>	<p>Specifies a data set that contains the VS COBOL II or COBOL/370 compiler load module.</p> <p>This parameter is obsolete.</p>
COMPILE	<u>YES</u> NO	Specifies whether to execute the compiler or assembler if the precompile step is successful.
CONNECT	(1) <u>(2)</u>	<p>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.</p> <p>The <b>default</b> is CONNECT(2).</p>
CONTROL	<u>NONE</u> CONLIST LIST SYMLIST	<p>Specify to help you trace the allocation of non-existent data sets. Use this parameter if you have a problem without an obvious cause.</p> <p>CONLIST displays CLIST commands after substitution for symbols and before command execution.</p> <p>LIST displays TSO commands after substitution for symbols and before command execution.</p> <p>SYMLIST displays all executable statements (TSO commands and CLIST statements) before substitution for symbols.</p>
COPTION	<u>NONE</u> <i>string</i>	<p>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.</p> <p>NONE specifies no options.</p>

Table 19. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
CPPCLASS	<i>dsname</i>	Specifies the data set that contains C++ class libraries.  The <b>default</b> 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 <b>default</b> 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 <b>default</b> 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 <b>default</b> is "CEE.SCEEH.H".
CPPLLIB	<i>dsname</i>	Specifies a data set that contains the C++ prelink automatic call library.  The <b>default</b> 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 <b>default</b> is "CEE.SCEEMSGP(EDCPMSGE)".
CPPSLIB	<i>dsname</i>	Specifies the data set that contains C++ header files for class libraries.  The <b>default</b> is "CBC.SCLBH.HPP".
CPPUTIL	<i>dsname</i>	Specifies the data set that contains procedures to set up and execute the C++ compiler.  The <b>default</b> 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 <b>default</b> is the value that is supplied when DB2 is installed, and is written in the data-only application defaults load module.



Table 19. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
DBRMLIB	<u>DEFAULT</u>	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.
	<i>dsname(member)</i>	
	NONE	
		DBRMLIB specifies the DBRM library and member that is defined on the DBRMLIB DD statement during DB2 precompiler processing.
		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.
		<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.
		<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:
		<ul style="list-style-type: none"> <li>Given INPUT(<i>outname.DBRM(dbrmmem)</i>): <ul style="list-style-type: none"> <li><i>outname.DBRM(dbrmmem)</i> - If the member name is specified</li> <li><i>outname.DBRM(dbrm)</i> - If no member name is specified</li> </ul> </li> </ul>
		NONE indicates that no LIBRARY( <i>dsname</i> ) subcommand keyword is specified on invocation.
DECARTH	<u>DEFAULT</u>	Specifies the maximum precision of decimal numbers.
	15	
	31	
		DEFAULT designates the value chosen, during installation, for the DECIMAL ARITHMETIC field on the APPLICATION PROGRAMMING DEFAULTS panel.
		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.
		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.
		DECARTH is ignored for Fortran.
DECIMAL	COMMA	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.
	PERIOD	
		COMMA makes the indicator a comma.
		PERIOD makes the indicator a period.
		The <b>default</b> is the value of the DECIMAL POINT field, set on the DB2 APPLICATION PROGRAMMING DEFAULTS panel during installation.

Table 19. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
DELIMIT	<u>DEFAULT</u> APOST QUOTE	<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 <b>default</b> depends on the host language and the value of RUN.</p> <ul style="list-style-type: none"> <li>For the PL/I language, the ENTRY value default is: <ul style="list-style-type: none"> <li>NONE if the RUN value is CICS</li> <li>PLISTART for any other RUN value.</li> </ul> </li> <li>For assembler language, the ENTRY value default is DLITASM if the RUN value is IMS.</li> <li>For COBOL, the ENTRY value default is DLITCBL if the RUN value is IMS.</li> <li>For any other language, the ENTRY value default is NONE (no specified entry point) for any RUN value.</li> </ul>
FLAG	I C E W	<p>Specifies the messages that you want to see. Use one of the following values to show messages of the corresponding types:</p> <p><b>I</b> All informational, warning, error, and completion messages</p> <p><b>W</b> Only warning, error, and completion messages</p> <p><b>E</b> Only error and completion messages</p> <p><b>C</b> 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 <b>default</b> 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 <b>default</b> is "SYS1.VSF2VCOMP(FORTVS2)".</p>
HOST	ASM C CPP IBMCOB FORTRAN PLI	<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"> <li>A COBOL program that uses object-oriented extensions</li> <li>A C++ program that uses object-oriented extensions and consists of more than one compilation unit</li> </ul> <p>The <b>default</b> is the value of the LANGUAGE DEFAULT field, set on the DB2 APPLICATION PROGRAMMING DEFAULTS panel during installation.</p>

Table 19. 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 <b>default</b> 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 <b>default</b> is 60.
LINK	<u>YES</u> NO	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 L <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 L <i>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 L <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 <b>default</b> , 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 <b>default</b> 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 <b>default</b> , NONE, to give no options.
MACRO	<u>YES</u> NO	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 19. 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 <b>default</b>, 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 <b>default</b> is ONE or 1 for PL/I and C.</p> <p>The <b>default</b> 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 <b>default</b> is '(DSNHPC)'.</p>
PLAN	<i>plan-name</i>	<p>Specifies the application plan that is created by the bind process.</p> <p>The <b>default</b> plan name is the first of the following available choices defined in the INPUT data set:</p> <ul style="list-style-type: none"> <li>• DBRM member name</li> <li>• Leftmost qualifier</li> </ul> <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 19. 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 <b>default</b> 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 <b>default</b> 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 <b>default</b> 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 <b>default</b> is "CEE.SCEEMSGP(EDCPMSG)".
POPTION	<u>NONE</u> <i>string</i>	Specifies a list of the C compiler language prelink utility options  Use the <b>default</b> , 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 compile 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 19. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
PRINT	NONE	Specifies where to send printed output, including the lists of options, source, cross-reference, error, and summary information.
	<i>dsname</i> LEAVE TERM	<p>The <b>default</b>, 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"> <li>• SYSCPRT.LIST for PL/I macro listings (these listings are overwritten by the compiler listings)</li> <li>• PCLIST for precompiler listings</li> <li>• CXLIST for CICS command translator listings</li> <li>• LIST for compiler listings</li> </ul> <p>The PRINT parameter is ignored for the compile step when HOST(CPP) is specified.</p> <ul style="list-style-type: none"> <li>• SYSOUT.PRELLIST for C prelink utility listings</li> <li>• LINKLIST for link-edit listings</li> </ul> <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"> <li>• Dynamically</li> <li>• In the JCL that is used to run the DSNH CLIST (if in batch mode)</li> <li>• With the TSO ALLOCATE command (before running DSNH)</li> </ul> <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 <b>default</b> 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 <b>default</b> 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 <b>default</b> is 8.</p>

Table 19. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
RUN	TSO or YES	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.
	BATCH or NO	
	CAF	
	CICS	
	IMS	
	RRSAF	TSO or YES indicates that the application program is to be scheduled for execution in the TSO environment, and executes the compiled program.
		BATCH or NO indicates that the application program is not to be scheduled for execution, and defaults to TSO as the execution environment.
		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)
		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.
		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.
		RRSAF indicates that the application program is not to be scheduled for execution, and identifies RRSF as the run time execution environment. RRSF applications cannot run in TSO.
RUNIN	TERM	Specifies where to get input for the RUN step.
	<i>dsname</i>	
	LEAVE	
	NONE	
		The <b>default</b> , TERM, gets input from the terminal.
		<i>dsname</i> specifies a data set that is to be used for the input.
		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.
		NONE allocates no input file.
RUNOUT	TERM	Specifies where to send output from the RUN step.
	<i>dsname</i>	
	LEAVE	
	NONE	
		The <b>default</b> , TERM, sends output to the terminal.
		<i>dsname</i> specifies a data set to receive output.
		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.
		NONE allocates no output file for the RUN step.
SOMDLLI	<i>dsname</i>	Specifies the name that of the SOM/MVS DLL import library.
		This parameter is obsolete.
SOURCE	NO	Specifies whether the source code and diagnostics are to be printed with output from the precompiler, CICS command translator, and compiler.
	YES	

Table 19. 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> ALL	Specifies how to interpret SQL statements and check syntax for use by either DB2 for z/OS or other database management systems.
		The <b>default</b> , 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 <b>default</b> is SQL(DB2).
SQLDELIM	<u>DEFAULT</u> APOSTSQL QUOTESQL	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> NO	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 <b>default</b> is the installation-defined subsystem name (often DSN).



Table 19. General DSNH CLIST parameters (continued)

Parameter	Value	Comments
TERM	TERM	Specifies where to send terminal output, including error information, error statements, and summary information.
	dsname LEAVE NONE	<p>The <b>default</b>, 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"> <li>• PCTERM for precompiler output</li> <li>• LIST for compiler output</li> </ul> <p>LEAVE sends the output to the current allocation for SYSTERM.</p> <p>NONE omits terminal output.</p>
TIME	ISO	Specifies the format for time values that are to be returned, overriding the format that is specified as the location default.
	JIS USA EUR LOCAL	There is <b>no default</b> because this option overrides the default previously specified.
VERSION	version-id	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 <b>default</b> is no version ID if specified at precompiler invocation.</p>
WORKUNIT	unit	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 <b>default</b> in batch mode is any eligible device.</p> <p>The <b>default</b> in any other mode is the UADS unit name for the current TSO user.</p>
WSECSPAC	integer	Specifies the amount of secondary space to allocate for work data sets, in the units given by SPACEUN.
		The <b>default</b> is 20.
WSPACE	integer	Specifies the primary size of the work data sets in the units given by SPACEUN.
		The <b>default</b> is 20.
XLIB	dsname	Specifies the linkage editor include library that is to be used for DB2 routines.
		The <b>default</b> is " <i>prefix</i> .SDSNLOAD".
XREF	NO	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	
<b>Note:</b> Precompiler options do not affect ODBC behavior.		

## DSNH/DSN subcommand summary

Table 20 and Table 21 on page 351 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 20. 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> <sup>1</sup> <i>dbrm-member-name</i> NONE <sup>2</sup>	MEMBER	<i>dbrm-member-name</i>	<sup>1</sup> DBRM member name, which is obtained from one of the following sources, in the order listed: <ul style="list-style-type: none"> <li>• BDBRMLIB member name</li> <li>• DBRMLIB member name</li> <li>• INPUT member name, or generated using <i>dsname</i>.</li> </ul> <sup>2</sup> Keyword is not specified on subcommand invocation.
BIND	<u>YES</u> <sup>1</sup> NO <sup>2</sup>	( <i>command-verb</i> )		<sup>1</sup> Execute BIND PLAN subcommand.  <sup>2</sup> Do not execute BIND PLAN subcommand.
BLIB	<u>NONE</u> <sup>1</sup> <i>dsname</i>	LIBRARY	<i>dbrm-pds-name</i>	<sup>1</sup> Keyword is not specified on subcommand invocation.
BnLIB <sup>1</sup>	<u>NONE</u> <sup>2</sup> <i>dsname</i>	LIBRARY	<i>list of dbrm-pds-names</i>	<sup>1</sup> <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.  <sup>2</sup> No additional data set names.
BMEM <sup>1</sup>	<u>NONE</u> <sup>2</sup> <i>list of dbrm-member-names</i>	MEMBER	<i>list of dbrm-member-names</i>	<sup>1</sup> Specify the first DBRM member name using the BDMEM parameter and any additional member names individually using this parameter.  <sup>2</sup> No additional DBRM member names.

Table 20. DSNH CLIST/ BIND PLAN subcommand summary (continued)

DSNH CLIST		BIND PLAN subcommand		Comments
Parameter	Value	Keyword	Value	
CACHESIZE	<u>NONE</u> <sup>1</sup> <i>decimal-value</i> <sup>2</sup>	CACHESIZE	<i>decimal-value</i> <sup>2</sup>	<sup>1</sup> The size is provided by the subsystem.  <sup>2</sup> Specify a size from 0 to 4096 bytes.
CICS	<u>NONE</u> <sup>1</sup> <i>application-ids</i>	CICS	<i>application-ids</i>	<sup>1</sup> Keyword is not specified on subcommand invocation.
CURRENTDATA	YES <u>NO</u> NONE	CURRENTDATA	YES <u>NO</u>	
CURRENTSERVER	<u>NONE</u> <sup>1</sup> <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 Version 7.
BDBRMLIB	<u>DEFAULT</u> <sup>1</sup> <i>dsname(member)</i> NONE <sup>2</sup>	LIBRARY	<i>dbrm-pds-name</i>	<sup>1</sup> 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.  <sup>2</sup> Keyword is not specified on subcommand invocation.
DEFER	<u>NONE</u> <sup>1</sup> PREPARE	DEFER	PREPARE	<sup>1</sup> 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> <sup>1</sup> <i>list of connection-ids</i>	DLIBATCH	<i>connection-name</i>	<sup>1</sup> 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	
IMSBMP	<u>NONE</u> <sup>1</sup> <i>insid</i>	IMSBMP	<i>insid</i>	<sup>1</sup> Keyword is not specified on subcommand invocation.

Table 20. DSNH CLIST/ BIND PLAN subcommand summary (continued)

DSNH CLIST		BIND PLAN subcommand		Comments
Parameter	Value	Keyword	Value	
IMSMPP	<u>NONE</u> <sup>1</sup> <i>imsid</i>	IMSMPP	<i>imsid</i>	<sup>1</sup> 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> <sup>1</sup> PREPARE	NODEFER	<u>PREPARE</u>	<sup>1</sup> Keyword is not specified on subcommand invocation.
OPTHINT	(' ') (' <i>hint-id</i> ')	OPTHINT	(' ') (' <i>hint-id</i> ')	
OWNER	<u>NONE</u> <sup>1</sup> <i>authorization-id</i>	OWNER	<i>authorization-id</i>	<sup>1</sup> Keyword is not specified on subcommand invocation.
PATH	( <i>schema-name</i> ) ( <b>USER</b> )( <i>schema-name</i> , <b>USER</b> ...)	PATH	( <i>schema-name</i> ) ( <b>USER</b> )( <i>schema-name</i> , <b>USER</b> ...)	
PKLIST	<u>NONE</u> <sup>1</sup> <i>list of collection-ids and package- names</i>	PKLIST	<i>list of collection-ids and package- names</i>	<sup>1</sup> The package names are not specified on subcommand invocation.
PLAN	<i>plan-name</i> <sup>1</sup>	PLAN ( <i>primary-keyword</i> )	<i>plan-name</i>	<sup>1</sup> <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"> <li>• DBRM member name</li> <li>• Left-most qualifier</li> </ul> If no name is found, a plan is not created.
QUALIFIER	<u>NONE</u> <sup>1</sup> <i>implicit-qualifier</i>	QUALIFIER	<i>qualifier-name</i>	<sup>1</sup> 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> <sup>1</sup> VARS	REOPT	<u>NONE</u> ALWAYS <u>ONCE</u> AUTO	<sup>1</sup> Keyword is not specified on subcommand invocation.
RETAIN	<u>NO</u> <sup>1</sup> YES <sup>2</sup>	RETAIN		<sup>1</sup> Keyword is not specified on subcommand invocation.  <sup>2</sup> 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 21. 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> <sup>1</sup> YES <sup>2</sup>	(command-verb)		<sup>1</sup> Do not execute BIND PACKAGE subcommand.  <sup>2</sup> Execute BIND PACKAGE subcommand.
PCICS	<u>NONE</u> <sup>1</sup> application-ids	CICS	application-ids	<sup>1</sup> Keyword is not specified on subcommand invocation.
COPY	<u>NONE</u> <sup>1</sup> collection-id. package-id	COPY	collection-id. package-id	<sup>1</sup> Keyword is not specified on subcommand invocation.
COPYVER	version-id	COPYVER	version-id	
PCURRENTDATA	NO YES <u>NONE</u>	CURRENTDATA	<u>YES</u> NO	
PDBPROTOCO	NONE DRDA PRIVATE	DBPROTOCOL	<u>DRDA</u> PRIVATE	If you specifit PRIVATE, your application cannot include SQL statements that were added to DB2 after Version 7.
PDBRMLIB	<u>DEFAULT</u> <sup>1</sup> dsname(member) NONE <sup>2</sup>	LIBRARY	dbrm-pds-name	<sup>1</sup> 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.  <sup>2</sup> Keyword is not specified on subcommand invocation.
PDEFER	<u>NONE</u> <sup>1</sup> PREPARE	DEFER	PREPARE	<sup>1</sup> Keyword is not specified on subcommand invocation.
PDEGREE	<u>1</u> ANY	DEGREE	<u>1</u> ANY	
PDISABLE	NONE BATCH CICS DB2CALL IMS DLIBATCH IMSBMP IMSMPP REMOTE RRSF	DISABLE	NONE BATCH CICS DB2CALL IMS DLIBATCH IMSBMP IMSMPP REMOTE RRSF	
PDLIBATCH	<u>NONE</u> <sup>1</sup> list of connection-ids	DLIBATCH	connection-name	<sup>1</sup> Keyword is not specified on subcommand invocation.
PDMEM	<u>DEFAULT</u> <sup>1</sup> dbrm-member- name NONE <sup>2</sup>	MEMBER	dbrm-member- name	<sup>1</sup> DBRM member name, which is obtained from one of the following sources, in the order listed: <ul style="list-style-type: none"> <li>• PDBRMLIB member name</li> <li>• DBRMLIB member name</li> <li>• INPUT member name, or generated using dsname</li> </ul> <sup>2</sup> Keyword is not specified on subcommand invocation.

Table 21. DSNH CLIST/ BIND PACKAGE subcommand summary (continued)

DSNH CLIST		BIND PACKAGE subcommand		
Parameter	Value	Keyword	Value	Comments
PDYNAMICRULES	<u>NONE</u> RUN BIND <u>DEFINE</u> INVOKE	DYNAMICRULES	RUN BIND <u>DEFINE</u> 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</u> C E W	FLAG	<u>I</u> C E W	
PIMSBMP	<u>NONE</u> <sup>1</sup> <i>imsid</i>	IMSBMP	<i>imsid</i>	<sup>1</sup> Keyword is not specified on subcommand invocation.
PIMSMPP	<u>NONE</u> <sup>1</sup> <i>imsid</i>	IMSMPP	<i>imsid</i>	<sup>1</sup> Keyword is not specified on subcommand invocation.
PISOLATION	<u>NONE</u> <sup>1</sup> RR RS CS <u>UR</u> NC	ISOLATION <sup>1</sup>	RR RS CS <u>UR</u> NC	<sup>1</sup> 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> <sup>1</sup> PREPARE	NODEFER	PREPARE	<sup>1</sup> 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> <sup>1</sup> <i>authorization-id</i>	OWNER	<i>authorization-id</i>	<sup>1</sup> Keyword is not specified on subcommand invocation.
PACKAGE	<u>DEFAULT</u> <sup>1</sup> <i>location-name.</i> <i>collection-id</i>	PACKAGE	<i>location-name.</i> <i>collection-id</i>	<sup>1</sup> 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> ) ( <b>USER</b> )( <i>schema-name</i> , <b>USER</b> , ...)	PATH	( <i>schema-name</i> ) ( <b>USER</b> )( <i>schema-name</i> , <b>USER</b> , ...)	
PQUALIFIER	<u>NONE</u> <sup>1</sup> <i>implicit-qualifier</i>	QUALIFIER	<i>qualifier-name</i>	<sup>1</sup> Keyword is not specified on subcommand invocation.
PRELEASE	<u>NONE</u> <sup>1</sup> COMMIT <u>DEALLOCATE</u>	RELEASE <sup>1</sup>	COMMIT DEALLOCATE	<sup>1</sup> 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> <sup>1</sup> VARS	REOPT	<u>NONE</u> ALWAYS <u>ONCE</u> AUTO	<sup>1</sup> Keyword is not specified on subcommand invocation.
REMOTE	<u>NONE</u> <sup>1</sup> <i>location-name</i> , < <i>luname</i> >	REMOTE	<i>network-name</i>	<sup>1</sup> Keyword is not specified on subcommand invocation.

Table 21. DSNH CLIST/ BIND PACKAGE subcommand summary (continued)

DSNH CLIST		BIND PACKAGE subcommand		
Parameter	Value	Keyword	Value	Comments
REPLVER	<u>NONE</u> <sup>1</sup> <i>version-id</i>	REPLVER	<i>version-id</i>	<sup>1</sup> <i>version-id</i> is not specified on subcommand invocation.
SQLERROR	<u>NOPACKAGE</u> <u>CONTINUE</u>	SQLERROR	<u>NOPACKAGE</u> <u>CONTINUE</u>	
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 22. 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
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 (*dsnhdcp*):** The following steps are required to enable DSNH CLIST to load your user-supplied *dsnhdcp* module rather than the DB2-supplied *dsnhdcp* module:

1. The JOBLIB or STEPLIB concatenation of any job or TSO userid that calls DSNH must allocate the library where the user-supplied *dsnhdcp* module resides (usually *prefix.SDSNEXIT*) before it allocates the library where the DB2-supplied *dsnhdcp* 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(IELOAA).
- 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</i> .PROG.PCLIST
Compiler listings	<i>prefix</i> .PROG.LIST
Link-edit listings	<i>prefix</i> .PROG.LIST

- 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(IELOAA)') -
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
SYSCPRT <sup>1</sup>	Used in the compile step
SYSUT10 <sup>1</sup>	Used in the compile step
SYSOUT	Used in the prelink-edit step

#### Note:

1. SYSUT10 and SYSCPRT are always allocated to the same data set or destination.

- SYSTERM is used in the compile 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:**

Chapter 17, "BIND PLAN (DSN)," on page 81

Chapter 70, "RUN (DSN)," on page 437

Chapter 16, "BIND PACKAGE (DSN)," on page 73

"The DSN command and its subcommands" on page 9



---

## Chapter 48. 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 49. 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

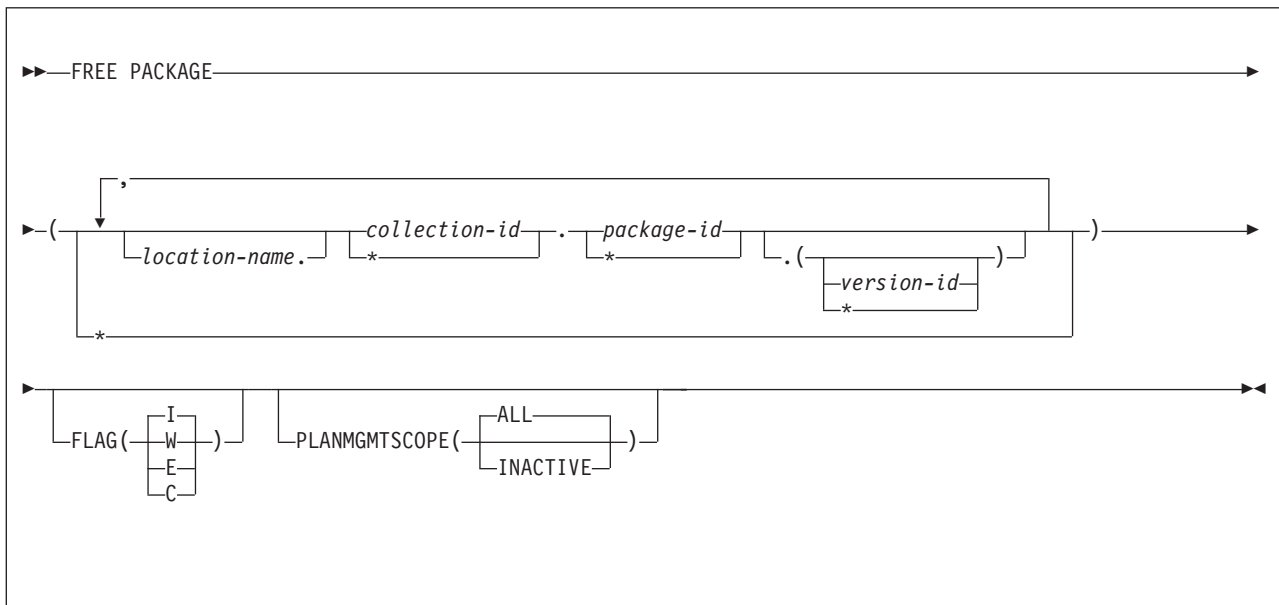
### Authorization

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 free a trigger package using the FREE PACKAGE subcommand.

## Examples

**Example 1:** 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)
```

**Example 2:** Free all packages at the local server in the collection named TESTCOLLECTION.

```
FREE PACKAGE (TESTCOLLECTION.*)
```



---

## Chapter 50. 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.

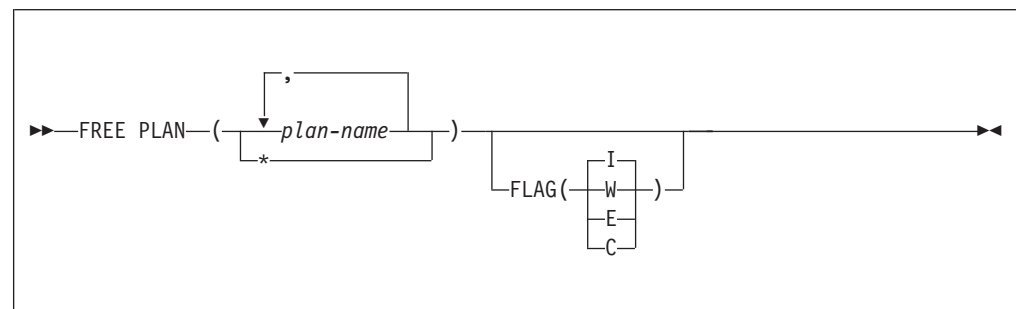
**Data sharing scope:** Group

### 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 51. FREE QUERY (DSN)

The DSN subcommand FREE QUERY removes one or more queries from the access path repository. 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. The location name 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. If the location name is specified, the name of the local DB2 subsystem must be defined.

The default 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. You cannot use the \* to free remote packages.

*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. You cannot use the \* to free remote packages.

*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)
```





---

## Chapter 52. 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 53. 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 do one of the following:

- 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 DSNA672I 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 DSNA672I MODIFY COMMAND FOR ADMIN SCHEDULER V91AADMT  
NORMAL COMPLETION
```

# Chapter 54. -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.

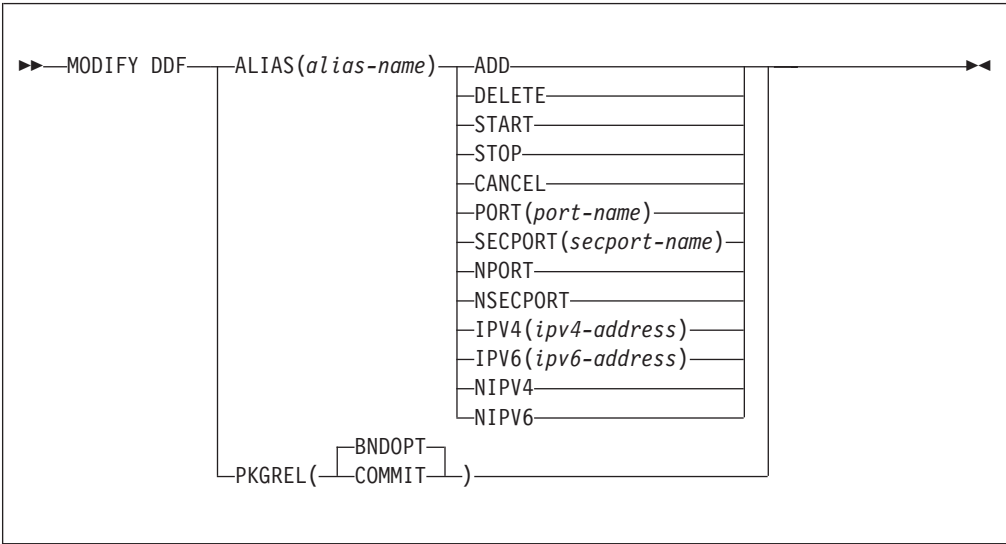
## Authorization

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

- DISPLAY 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

### 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 IPv4 address that can be returned to clients as part of the weighted server list when the clients connect to the specified

alias. DB2 does not activate this address. This address must be entered in the dotted decimal form. The IPV4 option is supported only when a member-specific IP address was previously specified by the DSNJU003 utility with the IPV4 or IPV6 keyword.

**IPV6(IPv6-address)**

Adds or replaces an existing IPv6 address that can be returned to clients as part of the weighted server list when the clients connect to the specified alias using an IPv6 address. DB2 does not activate this address. This address must be entered in the colon hexadecimal form. The IPV6 option is supported only when a member-specific IP address was previously specified by the DSNJU003 utility with the IPV4 or IPV6 keyword.

**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. BNDOPT is the default value of the MODIFY DDF PKGREL command.

**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) option during normal production operating hours. This option offers improved performance by reducing the amount of CPU costs for allocating and deallocating 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.

| **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 55. 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

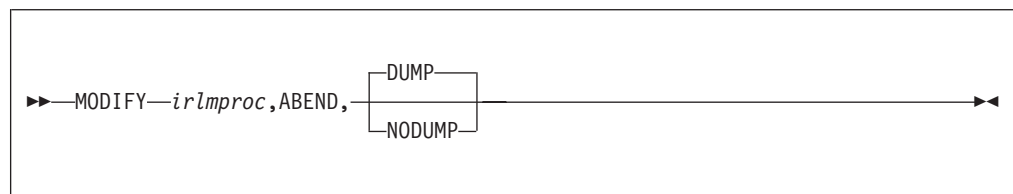
This command can be issued only from a z/OS console.

**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 56. 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

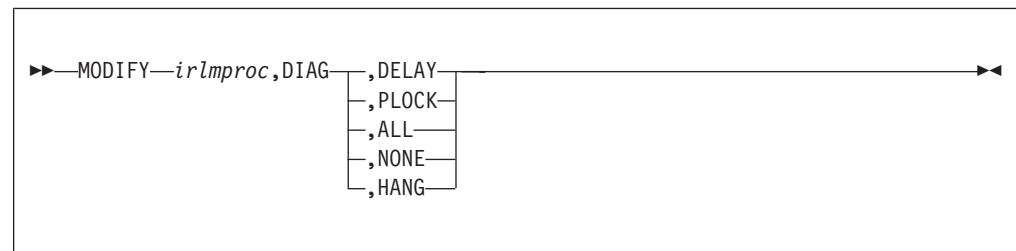
This command can be issued only from a z/OS console.

**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 57. 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 58. 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

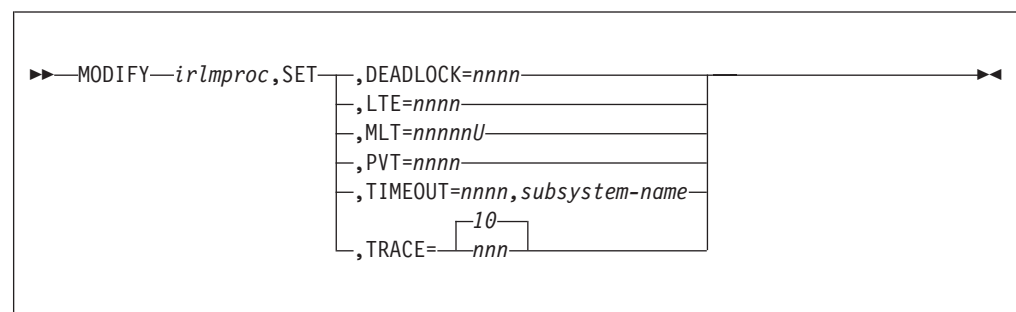
This command can be issued only from a z/OS console.

**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=** *nnnnnU*

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 59. 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

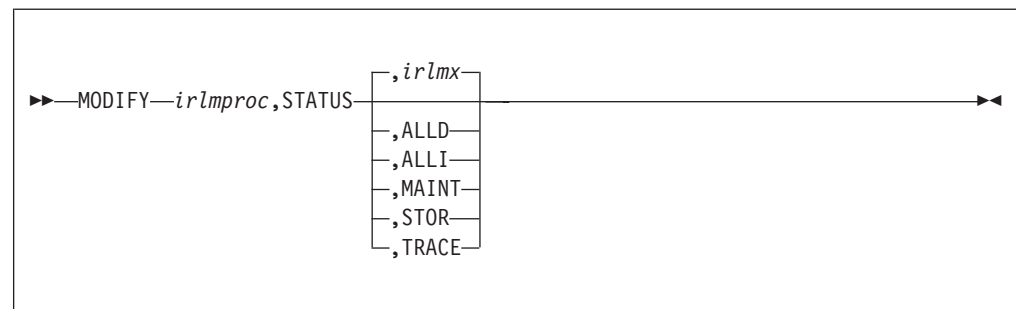
This command can be issued only from a z/OS console.

**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.

## 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” on page 396 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_LEVL
      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” on page 397 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” on page 397 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” on page 398 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” on page 398 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 LIEW: 32 LIE: 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” on page 394 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” on page 399 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” on page 398 to interpret the output.

## Output

Output from the command is presented in a DXR message. The command options determine which DXR message is used.

- “DXR100I” on page 394
- “DXR101I” on page 396

## DXR100I

- “DXR102I” on page 397
- “DXR103I” on page 398
- “DXR104I” on page 398
- “DXR179I” on page 399

---

### DXR100I     *irlmx* STOR STATS

**Explanation:** Message DXR100I is issued in response to the following command:

**F** *irlmproc,STATUS,STOR*

The message number is followed by multiple lines of output, including some or all of the following information.

**PC:** *text*

The parameter as specified in the *irlmproc*. Its value is YES. Monitor the PVT and AHW value when this is YES.

**LTEW:** *text*

The lock table entry width, which is the number of bytes needed for each lock table entry in the lock structure. The more users in the group, the more bytes are needed to manage each lock table entry. Expected values are:

- 2 (for up to six users)
- 4 (for up to 22 users)
- 8 (for up to 32 users)
- N/A (if the IRLM is not connected to a group)

**LTE:** *num* **M**

The number of lock table entries in units of 1,048,576 that were available in the coupling facility the last time this IRLM was connected to the group. If the value is less than one unit, the value is zero.

You can set this value initially with the LTE parameter in the IRLMPROC. Use the IRLM modify command **F irlmproc,SET,LTE=** to change this value.

This field is valid only for SCOPE=GLOBAL or NODISCON.

**RLE:** *num*

The number of record list entries available in the coupling facility the last time this IRLM was connected to the group.

This field is valid only for SCOPE=GLOBAL or NODISCON.

**RLEUSE:** *num*

The number of record list entries that were in use in the coupling facility at the time the **MODIFY** command is issued. If the IRLM is disconnected from the CF, this number represents those in use when last updated prior to DISCONNECT.

This field is valid only for SCOPE=GLOBAL or NODISCON.

**BB PVT:** *num* **M**

The extended private (below the bar) storage region limit threshold that is monitored for control blocks. When this limit is reached, new lock requests receive reason code 08 out-of-storage reason code unless they are must-complete requests. The default value is determined from the size of the extended private storage value minus 10% for a non-lock buffer to be used for IRLM and system required storage.

Use the IRLM MODIFY command **MODIFY irlmproc,SET,PVT=** to change this value.

**AB PVT (MEMLIMIT):** *num* **P**

Displays the current limit to private storage above the bar, also known as the MEMLIMIT, that is managed by MVS. This storage is used for locks. IRLM monitors the use of above-the-bar storage, and if it exceeds ninety percent of MEMLIMIT, then non-must-complete requests fail with reason code 08, the out-of-storage reason code.

Use the IRLM MODIFY command **MODIFY irlmproc,SET,MLT=** to change this value.



**CSA USE:**

Displays information about the Common Service Area.

**ACNT:** *num K*

Is the current CSA + ECSA usage for lock control structures. This storage is called “accountable” because it is accountable against the MAXCSA value. Its value is specified as either K for kilobytes or M for megabytes. When PC=YES, this value will be 0K.

**AHWM:** *num K*

Is the high water mark or greatest amount of CSA + ECSA allocated by IRLM during this initialization period for lock control structures. Its value is specified as either K for kilobytes or M for megabytes.

**CUR:** *num K*

Is the current CSA + ECSA usage. Its value is specified as either K for kilobytes or M for megabytes. This value accounts for CSA + ECSA storage obtained in IRLM. IRLM often gets storage for locks under an application's ASID and manages this storage regardless of the status of the owning ASID. The display of *cur* storage accounts for all of these under this IRLM's CSA usage.

**HWM:** *num K*

Is the high water mark or greatest amount of CSA + ECSA allocated by IRLM during this initialization period. Its value is specified as either K for kilobytes or M for megabytes.

**ABOVE 16M:** *num num K*

The number of IRLM control block segments above 16M and the amount of storage allocated to those segments. The amount of storage is specified as either K for kilobytes or M for megabytes.

**BELOW 16M:** *num num K*

The number of IRLM control block segments below 16M and the amount of storage allocated to those segments. The amount of storage is specified as either K for kilobytes or M for megabytes.

**PVT USE:**

Displays information about private storage.

**BB CUR:** *num K*

Is the current below the bar private usage. Its value is specified as either K for kilobytes or M for megabytes.

**BB CUR:** *num M*

Is the current above the bar private usage. Its value is specified as either M for megabytes or G for gigabytes.

**CLASS** *text*

A tabular summary of storage allocation, in the format:

```
CLASS  TYPE  SEGS    MEM
```

**CLASS** *text*

Specifies the category of storage. Expected values are:

**ACCNT**

The storage allocated to IRLM lock control blocks.

**PROC**

The storage allocated to IRLM structures used for IRLM processing (including requests). This storage includes CSA, ECSA, and IRLM private storage.

**MISC**

The storage allocated for the rest of IRLM's needs. This includes trace buffers and other diagnostic structures. This storage includes CSA, ECSA, and IRLM private storage.

**TYPE** *text*

Specifies the subcategory for CLASS. For example, T-1 is storage for resource block structures, T-2 is storage for resource request structures, and T-3 is storage for requestor structures. The storage for types T-1, T-2, and T-3 is allocated from ECSA when PC=NO or IRLM private storage when PC=YES.

**SEGS** *num*

The number of storage segments allocated

## DXR101I

### MEM *num* K

The storage specified as . Its value is specified in one of the following units: K for kilobytes, M for megabytes, or G for gigabytes.

**System action:** Processing continues normally.

**Sample output:** See sample output in the information center: DXR100I

---

DXR101I    *irlmx* STATUS SCOPE=*nnnnnn* DEADLOCK: *dddd* SUBSYSTEMS IDENTIFIED

```
NAME  STATUS UNITS HELD WAITING
ssname zzz      wun   reh  rew
                RET_LKS
                rtlks
```

**Explanation:** This message is issued in response to the following command:

F irlmproc,STATUS

A list of DBMSs identified to the IRLM is displayed. Work unit and lock information is displayed for each DBMS.

The message variables are:

*nnnnnn* One of the following:

#### LOCAL

SCOPE=LOCAL was specified as an EXEC parameter when the IRLM was started.

#### GLOBAL

SCOPE=GLOBAL was specified in the IRLMPROC and the IRLM is connected to a data sharing group.

#### DISCON

SCOPE=GLOBAL or SCOPE=NODISCON was specified in the IRLMPROC and the IRLM is not connected to a data sharing group.

#### NoDISC

SCOPE=NODISCON was specified in the IRLMPROC and the IRLM is connected to a data sharing group.

**GINIT** SCOPE=GLOBAL was specified in the IRLMPROC and the IRLM is joining a data sharing group.

#### GTERM

SCOPE=GLOBAL was specified in the IRLMPROC and the IRLM is terminating from a data sharing group.

*dddd* The deadlock detention interval time for IRLM. The value is measured in milliseconds.

*ssname* An eight-character DBMS name.

*tout* DBMS specified timeout value for the resource timeout. The value is measured in seconds.

*zzz* One of the following:

**UP** The DBMS is active.

#### DOWN

The DBMS failed.

**UP-RO** The DBMS subsystem is active and is currently identified to the IRLM as a 'read-only' subsystem.

**UP-NS** The DBMS subsystem is active and is currently identified to the IRLM as a 'no-share' subsystem.

#### CLEANUP

The IRLM drove the DBMS STATUS exit, indicating a failure condition, and is waiting for that DBMS to respond with a PURGE, indicating cleanup is complete.

**SFAIL** The IRLM to which the DBMS is identified disconnected from the data sharing group. The modify locks for all the subsystems on that IRLM were retained by IRLM. All DBMSs that are identified to IRLMs disconnected from a data sharing group are listed as SFAIL by the surviving IRLMs.

*wun* The number of work units under the DBMS that are holding or waiting for a lock. No work units exist for a failed DBMS. The field is set to '...' for a failed DBMS.

*reh* The number of resources locked by the DBMS.

*rew* The number of waiting lock requests. No waiting requests are permitted for a failed DBMS. The field is set to '...' for a failed DBMS.

*rtlks* The number of retained locks held by a subsystem that failed or was running on an IRLM that failed. In most cases, retained locks are purged when a DBMS reidentifies to IRLM; therefore the number is zero for active DBMSs. However, it is possible for a DBMS to hold both active and retained locks if it is recovering from a previous failure.

If no DBMSs are identified to this IRLM, the line beginning with *ssname* is replaced with NO INFORMATION AVAILABLE.

**System action:** Processing continues normally.

---

**DXR102I**     *irlmx* STATUS SUBSYSTEMS IDENTIFIED

NAME

*ssname*

STATUS

*zzz*

RET-LKS

*rtlks*

IRLMID

*id*

IRLM\_NAME

*iname*

IRLM\_LEVL

*ilevel*

**Explanation:** This message is issued in response to the following commands:

F irlmproc,STATUS,ALLD

F irlmproc,STATUS,irlmx

where *irlmx* is the IRLMX + IRLMID fields as specified in the IRLMPROC whose status is requested.

A list of database subsystems identified to the IRLMs in the data sharing group is displayed.

The message variables are:

*ssname* The eight-character name of the database subsystem.

*zzz* One of the following:

**UP** The database subsystem is active.

**DOWN**

The database subsystem failed.

**UP-RO** The database subsystem is active and is currently identified to the IRLM as a read-only subsystem.

**UP-NS** The database subsystem is active and is currently identified to the IRLM as a no-share subsystem.

**CLEANUP**

The IRLM drove the database subsystem STATUS exit, indicating a failure condition, and is waiting for that database subsystem to respond with a PURGE, indicating cleanup is complete.

**SFAIL** The IRLM to which the database subsystem is identified was disconnected from the data sharing group. The modify locks for all the database subsystems on that IRLM were retained by IRLM. All database subsystems that are identified to IRLMs disconnected from a data sharing group are listed as SFAIL by the surviving IRLMs.

*rtlks* The number of retained locks held by a subsystem that failed or was running on an IRLM that failed. In most cases, retained locks are purged when a database subsystem reidentifies to IRLM; therefore, the number is zero for active database subsystems. However, it is possible for a database subsystem to hold both active and retained locks if it is recovering from a previous failure.

*id* The ID of the IRLM to which the database subsystem is identified.

*iname* The name of the IRLM to which the database subsystem is identified.

If no database subsystems are identified to any IRLM or to the IRLM specified, or the IRLM specified is not known, the line beginning with *ssname* is replaced with NO INFORMATION AVAILABLE.

*ilevel* The IRLM function level that the database subsystem requests. This IRLM level is the level that the database subsystem requires to perform locking operations. This IRLM level might be less than the actual IRLM function level; however, IRLM still operates at the actual function level. If the database subsystem failed and

## DXR103I • DXR104I

its IRLM is the last IRLM to disconnect from the group, the value will be zero when displayed by any peer member that joins until the failed database subsystem is restarted.

**System action:** Processing continues normally.

---

**DXR103I**     *irlmx* STATUS IRLMS PARTICIPATING IN DATA SHARING GROUP FUNCTION LEVEL *glv m*

IRLM-NAME

*iname*

IRLMID

*id*

STATUS

*zzz*

LEVEL

*lv*

SERVICE

*s*

MIN\_LEVEL

*mlv*

MIN\_SERVICE

*ms*

**Explanation:** This message displays the active IRLMs in response to the following command:

F irlmproc,STATUS,ALLI

A list of IRLMs in the group that are actively data sharing is displayed. If not data sharing, the single IRLM is displayed.

*glv*     The IRLM function level in use by all the IRLM(s) in the data sharing group.

*m*       A decimal number such as 1, 2, and so on.

When more than 10 lines are required in response to a status command, multiple messages are issued with *m* incremented by one in each successive message.

*iname*   A 4-character IRLM name. If there is a '\*' appended to the *iname*, that *iname* is the GDM.

*id*       The ID of the IRLM.

*zzz*       One of the following:

**UP**        The IRLM is active.

**DOWN**

          The IRLM failed.

*lv*        The current IRLM function level.

*s*         The IRLM service or release that corresponds to the function level given in *lv*.

*mlv*       The minimum IRLM function level this IRLM can coexist with.

*ms*        The IRLM service or release that corresponds to the function level given in *mlv*

**System action:** Processing continues normally.

---

**DXR104I**     *irlmx* MAINTENANCE LEVELS

**Explanation:** This display is produced by the MODIFY irlmproc ,STATUS,MAINT command. Maintenance levels of all IRLM load modules are displayed to the console in two column format except for DXRRL183, DXRRLFTB, DXRRLM50, and DXRRL186. , Modules show the most recent APAR level and the compile date applied to each CSECT.

**System action:** IRLM processing continues normally.

**Operator response:** Review the maintenance level for any suspected module. If you are unable to correct the problem, contact your IBM Support Center for assistance.

**System programmer response:** No action is required.

---

**DXR179I**    *irlmx* TRACE USAGE TRACE BUFFER STORAGE IN USE: *nnnnnn*KB MAXIMUM NUMBER OF TRACE BUFFERS ALLOWED PER TRACE TYPE: *nnn* TRACE TYPE ACTIVE BUFFERS IN USE CTRACE WRITER

**Explanation:** This message is issued in response to the following command:

F irlmproc,STATUS,TRACE

The message indicates the maximum number of 64KB buffers that IRLM can use for each of its subcomponent trace types.

#### TRACE TYPE

The trace types are:

- SLM**    Traces interactions with z/OS locking component. Applicable only for data sharing
- XIT**    Traces asynchronous interactions with the z/OS locking component. Applicable only for data sharing
- XCF**    Traces interactions with z/OS cross-system coupling services. Applicable only for data sharing.
- DBM**    Traces interactions with the DBMS identified to this IRLM.
- EXP**    Traces exception conditions.
- INT**    Traces member and group events outside normal locking activity.
- RLE**    Traces interactions between z/OS and IRLM specific to Record List Entries.

#### ACTIVE

Whether the trace is active. (You cannot deactivate the EXP and INT traces.)

#### BUFFERS IN USE

How many 64KB buffers are currently being used by this trace.

#### CTRACE WRITE

Whether the external CTRACE writer is on.

**System action:** Processing continues normally.

**Operator response:** Trace buffers can be set using the modify irlmproc,set,trace=nnn command. You can activate or deactivate traces by using the **TRACE CT** command of z/OS. You cannot turn off the EXP, RLE and INT traces. The XIT, EXP, RLE and INT are automatically activated when you start IRLM All traces are automatically activated when TRACE=YES is specified on the IRLMPROC.



---

## Chapter 60. -MODIFY TRACE (DB2)

The DB2 command MODIFY TRACE changes the IFCIDs (trace events) associated with a particular active trace.

The DB2 command MODIFY TRACE does the following:

- 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.

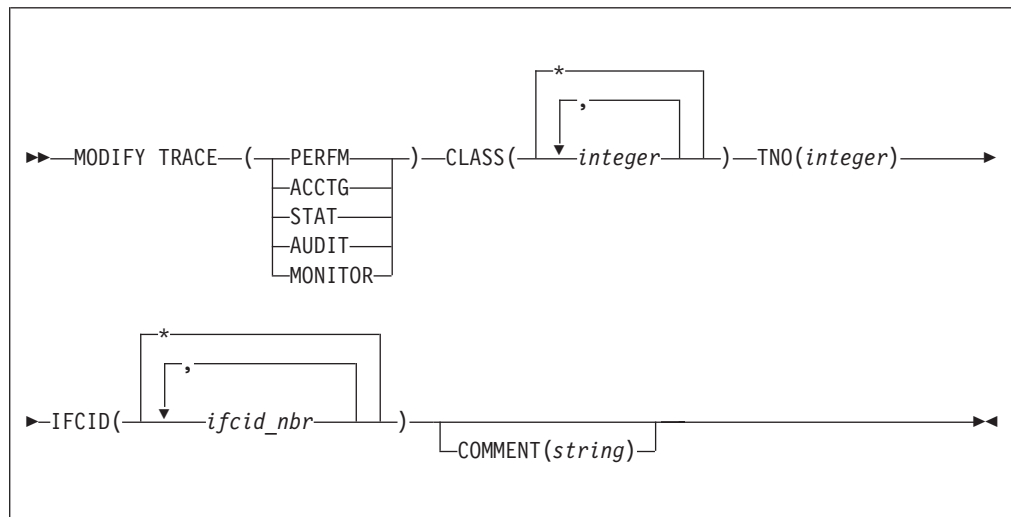
### Authorization

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 23. 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:**

Chapter 99, “-STOP TRACE (DB2),” on page 549

Chapter 87, “-START TRACE (DB2),” on page 497

Chapter 38, “-DISPLAY TRACE (DB2),” on page 293



---

## Chapter 61. 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 when you change the authorizations, create a new index for the package, or use RUNSTATS. When the REBIND PACKAGE(\*) command is issued, trigger packages will not be 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.

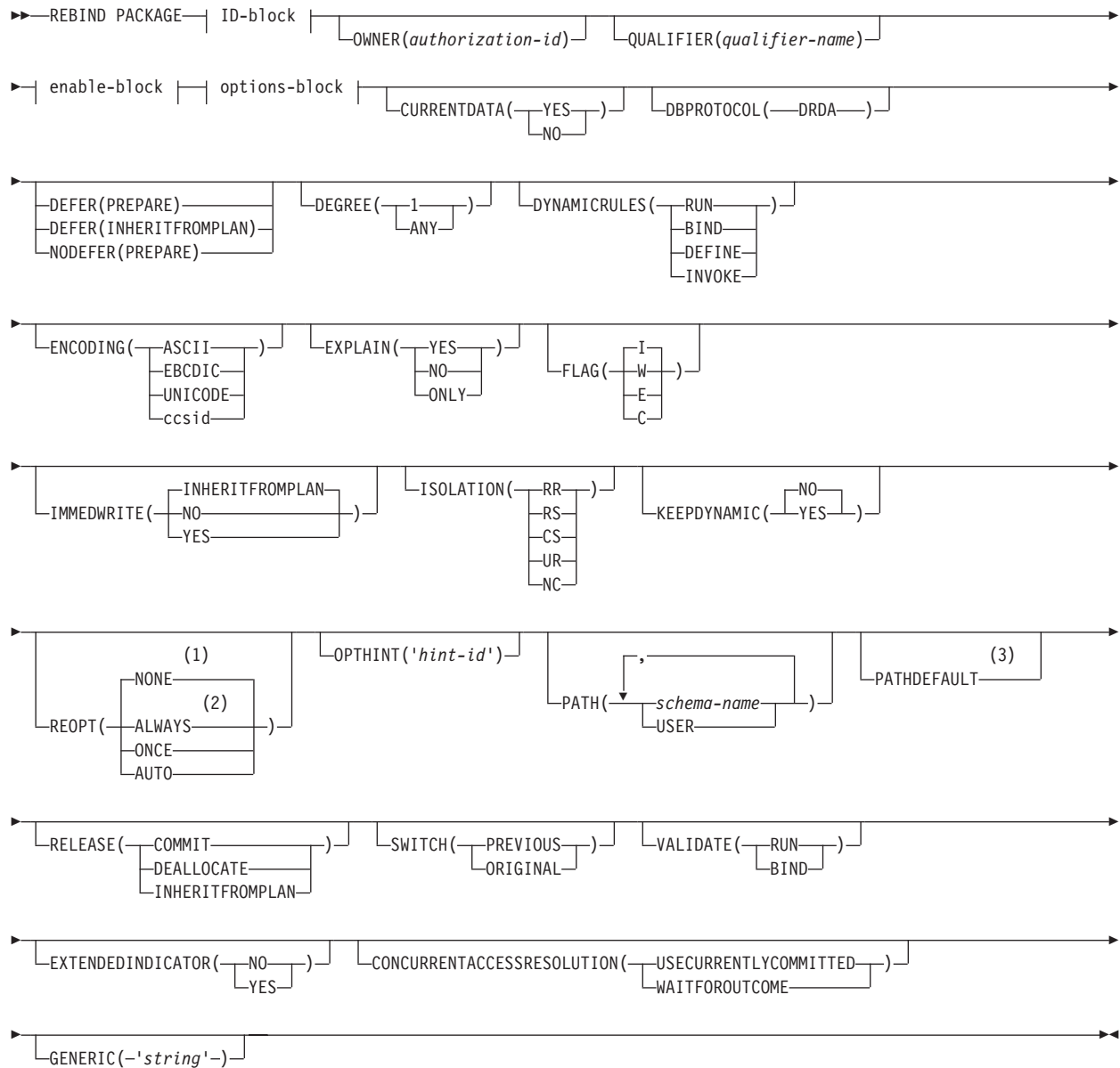
*Table 24. Summary of privileges for REBIND PACKAGE*

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"><li>• Ownership of the package</li><li>• BIND privilege on the package</li><li>• BINDAGENT privilege from the owner of the package</li><li>• PACKADM authority on the collection or on all collections</li><li>• SYSADM or SYSCTRL or System DBADM authority</li></ul>

Table 24. Summary of privileges for REBIND PACKAGE (continued)

Option	Authorization required to run REBIND PACKAGE
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"> <li>• OWNER <i>authorization-id</i> must be one of the primary or secondary authorization IDs of the binder</li> <li>• BINDAGENT privilege from the owner of the package</li> <li>• SYSADM or SYSCTRL or System DBADM authority</li> </ul>
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"> <li>• BIND privilege on the package</li> <li>• PACKADM authority on the collection or on all collections</li> <li>• SYSADM or SYSCTRL or System DBADM authority</li> </ul> <p><b>Specifying the OWNER:</b> 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>

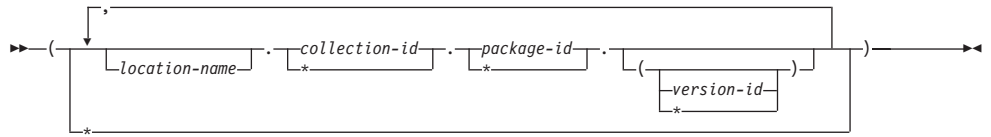
## Syntax



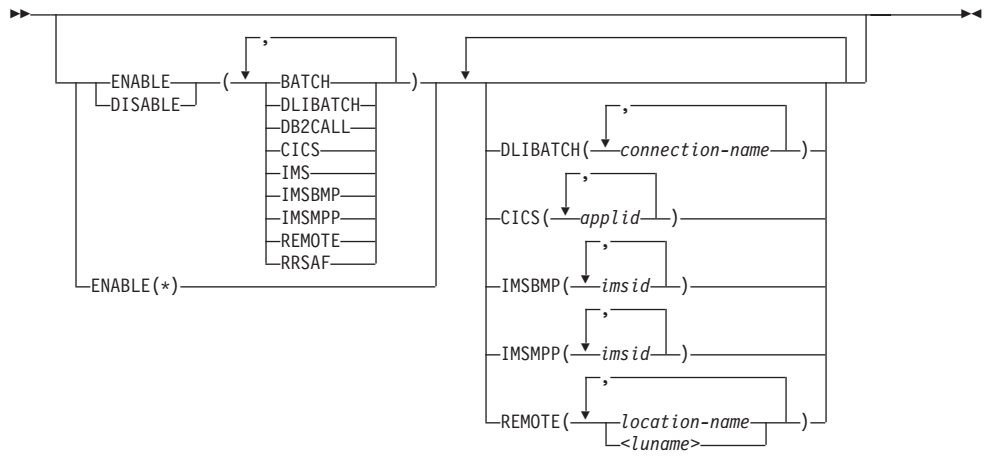
### Notes:

- 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.

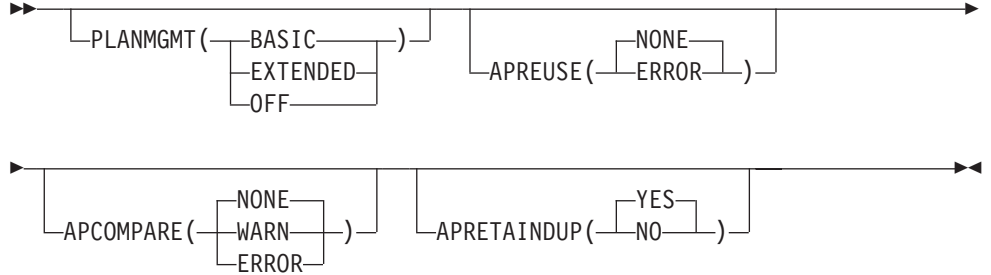
### ID-block



### enable-block



### options-block



## Option descriptions

For descriptions of the options shown in the syntax diagram, see the topic Chapter 19, “BIND and REBIND options,” on page 91.

## 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's 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:

Chapter 62, "REBIND PLAN (DSN)," on page 411

Chapter 16, "BIND PACKAGE (DSN)," on page 73

Chapter 19, "BIND and REBIND options," on page 91





---

## Chapter 62. REBIND PLAN (DSN)

The DSN subcommand REBIND PLAN rebinds an application plan when you make changes that affect the plan, but do not change the SQL statements in the programs.

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.

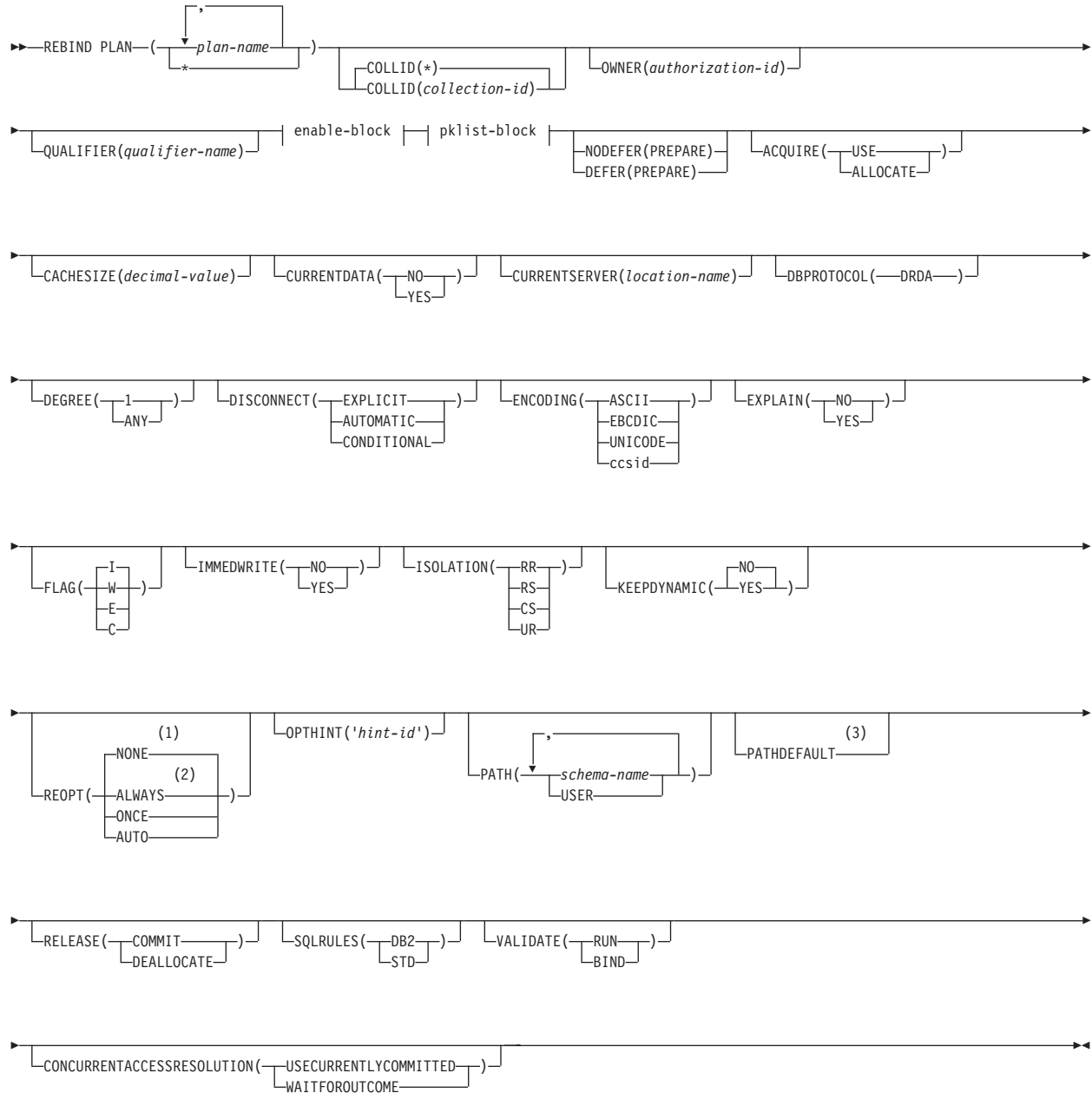
*Table 25. Summary of privileges for REBIND PLAN*

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"><li>• Ownership of the plan</li><li>• BIND privilege on the plan</li><li>• BINDAGENT privilege from the owner of the plan</li><li>• SYSADM or SYSCTRL or System DBADM authority</li></ul>

Table 25. Summary of privileges for REBIND PLAN (continued)

Option	Authorization required to run REBIND PLAN
REBIND PLAN 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"> <li>• OWNER <i>authorization-id</i> must be one of the primary or secondary authorization IDs of the binder</li> <li>• BINDAGENT privilege from the owner of the plan</li> <li>• SYSADM or SYSCTRL or System DBADM authority</li> </ul>
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"> <li>• BIND privilege on the plan</li> <li>• SYSADM or SYSCTRL or System DBADM authority</li> </ul> <p><b>Specifying the OWNER:</b> 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"> <li>• CREATEIN privilege on the COLLID you specified.</li> </ul>
PKLIST, specifying individual packages	<p>Authorization ID of the process must include one of the following authorities:</p> <ul style="list-style-type: none"> <li>• EXECUTE privilege on each package specified in the PKLIST</li> <li>• PACKADM authority on specific collections containing packages or on collection *</li> <li>• SYSADM or DATAACCESS authority</li> </ul>
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"> <li>• EXECUTE privilege on each package in the collection</li> <li>• EXECUTE privilege on <i>collection-id</i> .*</li> <li>• PACKADM authority on <i>collection-id</i> or on *</li> <li>• SYSADM or DATAACCESS authority</li> </ul>

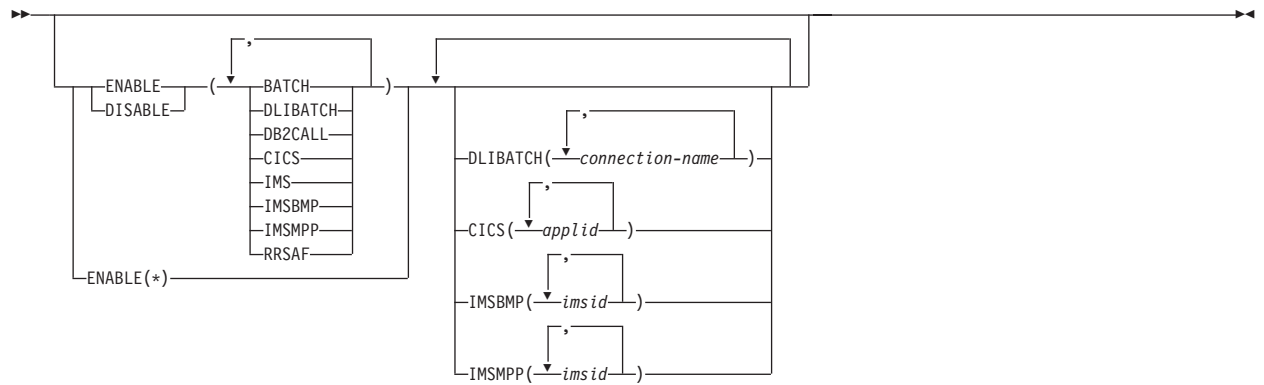
## Syntax



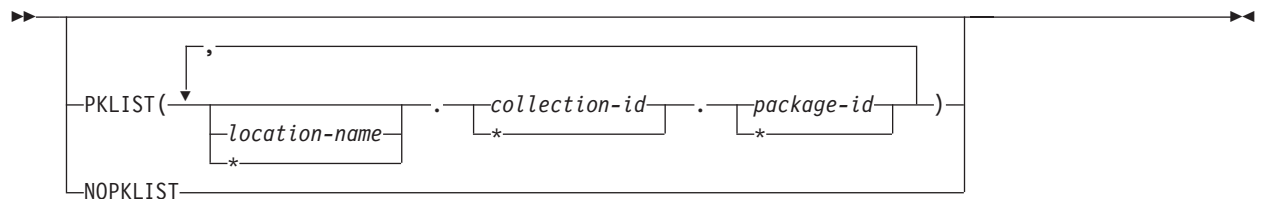
### Notes:

- 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 is mutually exclusive with the PATH keyword. Do not specify both keywords in the same REBIND command.

## enable-block



## pklist-block



## Option descriptions

For descriptions of the options shown in the syntax diagram, see the topic Chapter 19, “BIND and REBIND options,” on page 91.

## Usage note

If you rebind multiple plans, DB2 commits each successful rebind before rebinding the next plan.

## 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:**

Chapter 61, "REBIND PACKAGE (DSN)," on page 405

Chapter 17, "BIND PLAN (DSN)," on page 81

Chapter 19, "BIND and REBIND options," on page 91



---

## Chapter 63. 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

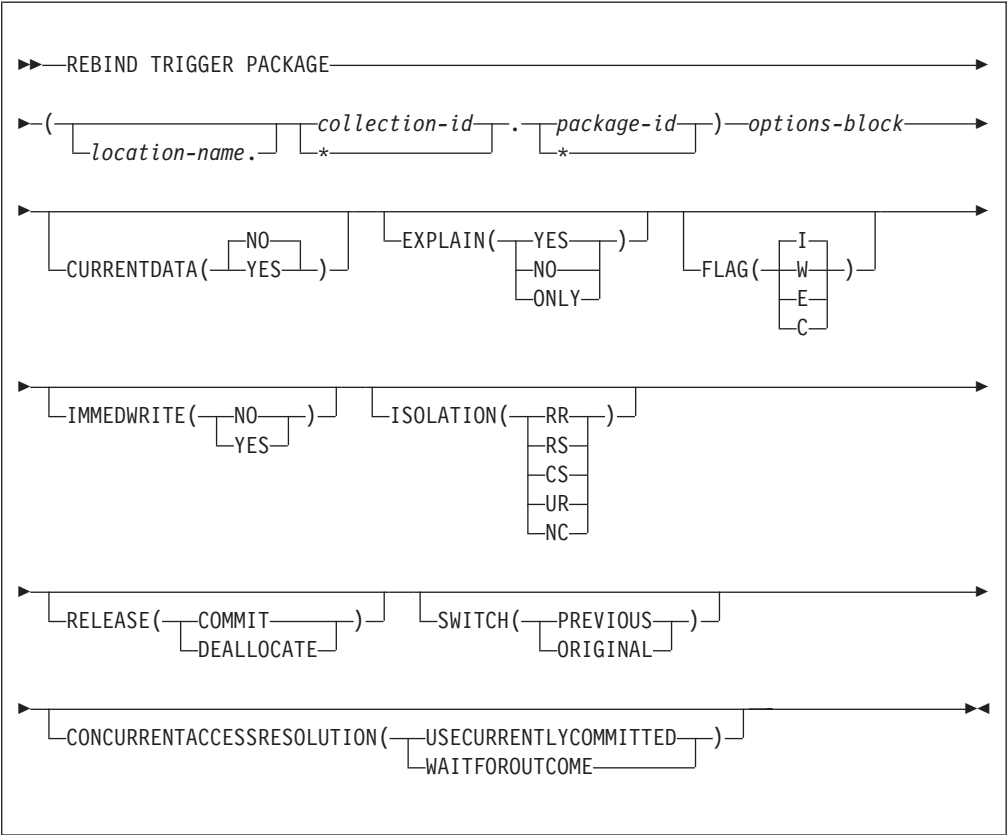
When the trigger package is bound, the privileges of the current authorization ID are used when checking authority to bind statements within the triggered action. On REBIND TRIGGER PACKAGE, you need one of the following privileges or authorities:

- Ownership of the trigger package
- BIND privilege on the trigger package
- BINDAGENT privilege from the owner of the trigger package

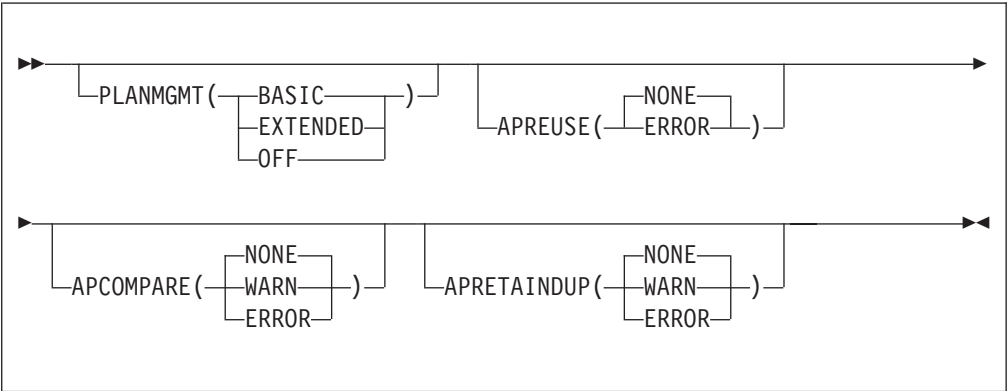
- PACKADM authority on the collection or on all collections
- System DBADM authority
- SYSCTRL authority
- SYSADM authority

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, 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 19, “BIND and REBIND options,” on page 91.

## 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 reference:

Chapter 19, "BIND and REBIND options," on page 91

---

## Chapter 64. -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:

 Recovering the BSDS from a backup copy (DB2 Administration Guide)



---

## Chapter 65. -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

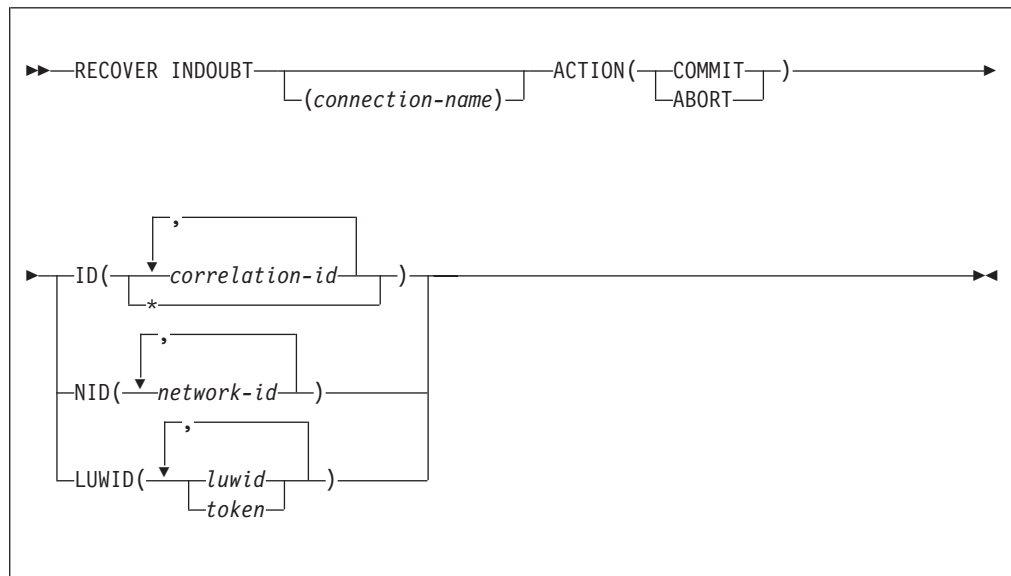
### Authorization

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 cancel 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)

Cancels 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 threads to recover based on their network IDs.

*network-id* is a network ID that is associated with an individual thread. You can use more than one network ID for the same connection name.

For IMS and CICS connections, a network ID is specified as *net-node.number* , which is 3 to 25 characters in length.

- *net-node* is the network-node name of the system that originated the unit of work. *net-node* is one to eight characters in length.
- *number* is a unique number within the system of origin. *number* is 1 to 16 characters in length.

For RRSAF connections, a 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-character number.

The network ID appears on the recovery log of the commit coordinator as a 16-byte unique identification of a unit of work.

- For IMS and CICS, 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.

**LUWID**

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 66. -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

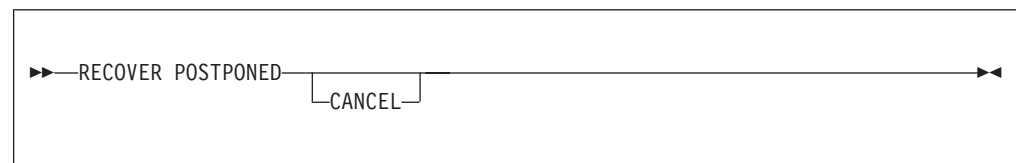
### Authorization

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
- SYSTRL 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 note

**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 DSN1024I 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 - DSN1ARPL BACKOUT PROCESSING HAS COMPLETED
           FOR PAGESET DBKD0103.IPKD013A PART 00000004.
DSNI024I - DSN1ARPL BACKOUT PROCESSING HAS COMPLETED
           FOR PAGESET DBKD0103.TPKD0103 PART 00000004.
DSNI024I - DSN1ARPL BACKOUT PROCESSING HAS COMPLETED
           FOR PAGESET DBKD0103.IXKD013C PART (n/a).
DSNI024I - DSN1ARPL BACKOUT PROCESSING HAS COMPLETED
           FOR PAGESET DBKD0103.IUKD013B PART (n/a).
DSNI024I - DSN1ARPL BACKOUT PROCESSING HAS COMPLETED
           FOR PAGESET DBKD0103.IPKD013A PART 00000002.
DSNI024I - DSN1ARPL BACKOUT PROCESSING HAS COMPLETED
           FOR PAGESET DBKD0103.TPKD0103 PART 00000002.
DSNI024I - DSN1ARPL BACKOUT PROCESSING HAS COMPLETED
           FOR PAGESET DBKD0101.IXKD011C PART (n/a).
DSNI024I - DSN1ARPL BACKOUT PROCESSING HAS COMPLETED
           FOR PAGESET DBKD0101.IXKD011B PART (n/a).
DSNI024I - DSN1ARPL BACKOUT PROCESSING HAS COMPLETED
           FOR PAGESET DBKD0101.IUKD011A PART (n/a).
DSNI024I - DSN1ARPL 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
```

---

## Chapter 67. -REFRESH DB2,EARLY (DB2)

The DB2 command -REFRESH DB2,EARLY reloads the EARLY code modules that were loaded at IPL time, and rebuilds the EARLY control block.

Executing the -REFRESH DB2,EARLY command is an alternative to IPLing z/OS for activating maintenance to EARLY code, after you apply that maintenance to the *prefix.SDSNLINK* data set.

When -REFRESH DB2,EARLY is run, previous copies of the EARLY modules are deleted from the system the next time that DB2 is started. This command is only valid when DB2 is inactive.

**Important:** If *prefix.SDSNLINK* is LLA-managed, you need to perform an LLA refresh after you apply EARLY code maintenance, and before you issue the -REFRESH DB2,EARLY command. You can perform an LLA refresh by issuing the z/OS command MODIFY LLA,REFRESH.

**Abbreviation:** -REF DB2

### Environment

This command can only be issued 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 EARLY code can only be properly updated when DB2 is not active

**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.

### Syntax



```
➡—REFRESH DB2,EARLY—⬅
```

### Option descriptions

The following option is required.

## EARLY

Indicates that the EARLY block will be rebuilt, and that all of the modules loaded at IPL time are reloaded.

## Usage notes

*Scope of the -REFRESH command:* The -REFRESH command has member-only scope, and needs to be executed for every instance where a unique recognition character for the -START command has been defined.

After you issue the command, DB2 displays the maintenance levels of the EARLY code modules that were loaded.

## Examples

### Example: Refreshing the DB2 EARLY code

Suppose that the command prefix for a subsystem is -VA1A. This command reloads the EARLY code modules that were loaded at IPL time, and rebuilds the EARLY control block.

```
-VA1A REFRESH DB2,EARLY
```

After you issue the command, DB2 displays output like this:

```
DSN3100I -VA1A DSN3UR00 - SUBSYSTEM VA1A READY FOR START COMMAND
DSN3117I VA1A MAINTENANCE LEVELS
```

CSECT	DATE	APAR	CSECT	DATE	APAR
DSN3UR00	01/27/05	09.35	DSN3EC0X	01/24/05	09.10
DSN3ECMS	01/13/05	11.16	DSN3RIB	12/13/04	10.46
DSN3RDMP	12/13/04	10.27	DSNAET03	11/30/04	15.42
DSNAET04	11/30/04	15.42	DSNAPRHO	11/30/04	15.42
DSNAPRHX	12/20/04	10.10	DSNVRMTR	11/30/04	19.53
DSNVRSRB	11/30/04	19.53	DSNZPARS	11/30/04	21.48
DSN3AC0X	11/30/04	21.50	DSN3CL0X	11/30/04	21.51
DSN3DEQ0	11/30/04	21.52	DSN3ENQ0	11/30/04	21.52
DSN3EST0	11/30/04	21.53	DSN3RRSX	11/30/04	21.53
DSN3RRXF	11/30/04	21.54	DSN3RS0X	11/30/04	21.54
DSN3RTR0	11/30/04	21.54	DSN3SPRX	11/30/04	21.54

### Related tasks:

 Choosing link list options (DB2 Installation and Migration)

---

## Chapter 68. -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

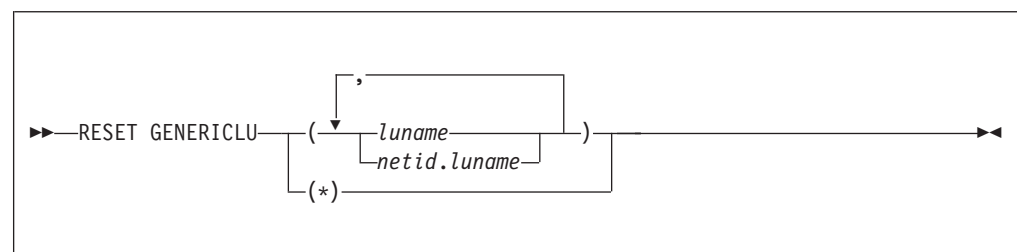
### Authorization

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 69. -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.

This command *must* be used to purge indoubt thread information in the following situations:

- For threads where DB2 has a coordinator responsibility that it cannot fulfill because of participant cold start, sync point protocol errors, or indoubt resolution protocol errors.
- For threads that were indoubt but were resolved with the RECOVER INDOUBT command, and subsequent resynchronization with the coordinator shows heuristic damage.

The RESET column of a display thread report for indoubt threads indicates whether information in the report must be purged with this command.

This command can also be used to purge indoubt thread information for threads where:

- DB2 has a coordinator responsibility even when no errors have been detected that preclude automatic resolution with the participants. The FORCE keyword must be specified to purge this information. Resynchronization with affected participants is not performed.
- DB2 has a participant responsibility even when no errors have been detected that preclude automatic resolution with the coordinator. Resynchronization with the coordinator will not be performed.

**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

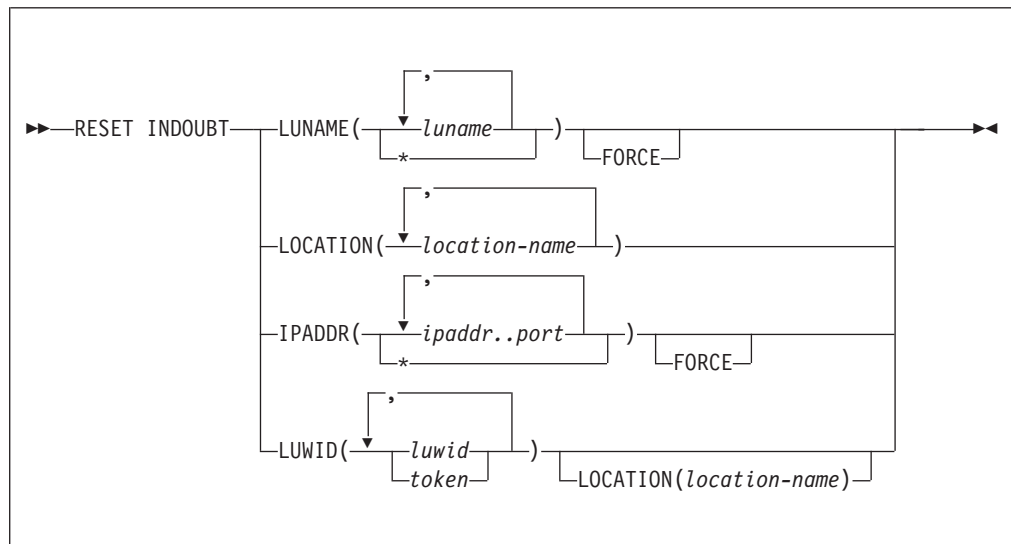
### Authorization

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.

---

## Chapter 70. 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

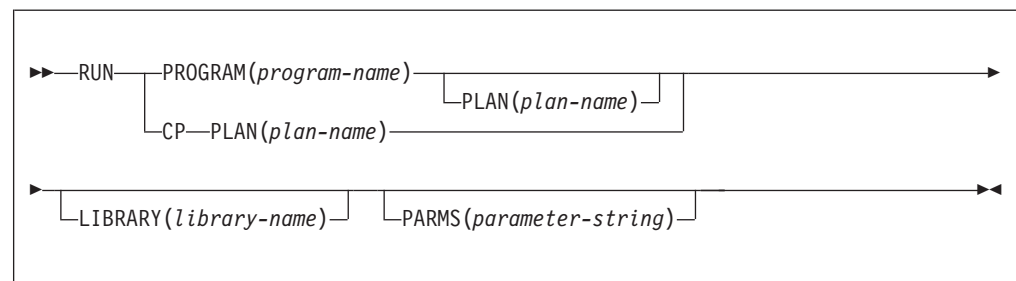
### Authorization

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') PARMS ('/O'TOOLE')
```



---

## Chapter 71. -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

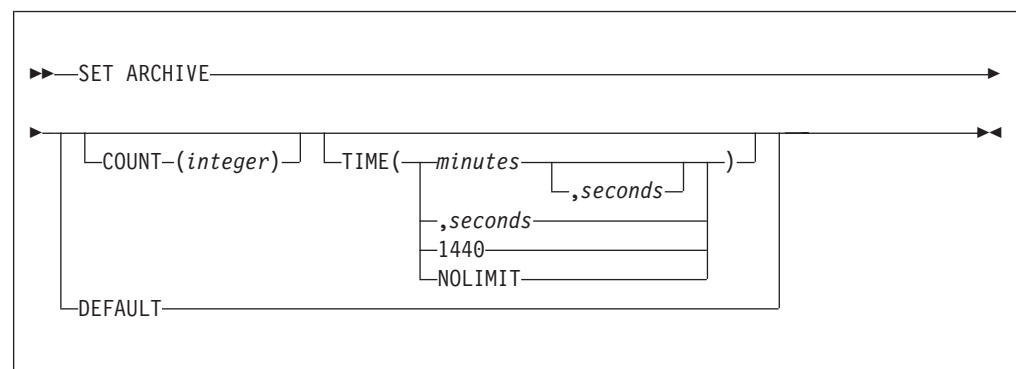
### Authorization

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 72. -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

▶▶—SET LOG

(1)

SINGLE

BOTH

LOGLOAD(*integer*)

CHKTIME(*integer*)

LOGLOAD(*integer*)

SUSPEND

RESUME

NEWLOG—(*—data-set-name—*)—COPY—(*—log-copy—*)

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 deadlock 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.

## 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=DSNA10.SDSNLOAD,DISP=SHR
//NEWLOG EXEC PGM=IKJEFT01,DYNAMNBR=20
//DSNTRACE DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(V91A)
-SET LOG NEWLOG(DSNC910.LOGCOPY1.DS04) COPY(1)
-SET LOG NEWLOG(DSNC910.LOGCOPY1.DS05) COPY(1)
-SET LOG NEWLOG(DSNC910.LOGCOPY1.DS06) COPY(1)
-SET LOG NEWLOG(DSNC910.LOGCOPY1.DS07) COPY(1)
-SET LOG NEWLOG(DSNC910.LOGCOPY2.DS04) COPY(2)
-SET LOG NEWLOG(DSNC910.LOGCOPY2.DS05) COPY(2)
-SET LOG NEWLOG(DSNC910.LOGCOPY2.DS06) COPY(2)
-SET LOG NEWLOG(DSNC910.LOGCOPY2.DS07) COPY(2)
END
/*
//SYSIN DD *
/*
```

The following messages are issued:

```
DSNJ363I ) DSNJW106 COPY1 LOG DATA SET
DSNC910.LOGCOPY1.DS04 ADDED TO THE ACTIVE LOG INVENTORY
DSNJ363I ) DSNJW106 COPY1 LOG DATA SET
DSNC910.LOGCOPY1.DS05 ADDED TO THE ACTIVE LOG INVENTORY
DSNJ363I ) DSNJW106 COPY1 LOG DATA SET
DSNC910.LOGCOPY1.DS06 ADDED TO THE ACTIVE LOG INVENTORY
DSNJ363I ) DSNJW106 COPY1 LOG DATA SET
DSNC910.LOGCOPY1.DS07 ADDED TO THE ACTIVE LOG INVENTORY
DSNJ363I ) DSNJW106 COPY2 LOG DATA SET
DSNC910.LOGCOPY2.DS04 ADDED TO THE ACTIVE LOG INVENTORY
DSNJ363I ) DSNJW106 COPY2 LOG DATA SET
DSNC910.LOGCOPY2.DS05 ADDED TO THE ACTIVE LOG INVENTORY
DSNJ363I ) DSNJW106 COPY2 LOG DATA SET
DSNC910.LOGCOPY2.DS06 ADDED TO THE ACTIVE LOG INVENTORY
DSNJ363I ) DSNJW106 COPY2 LOG DATA SET
DSNC910.LOGCOPY2.DS07 ADDED TO THE ACTIVE LOG INVENTORY
```





---

## Chapter 73. -SET SYSPARM (DB2)

The DB2 command SET SYSPARM lets you change subsystem parameters while DB2 is up.

### 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:

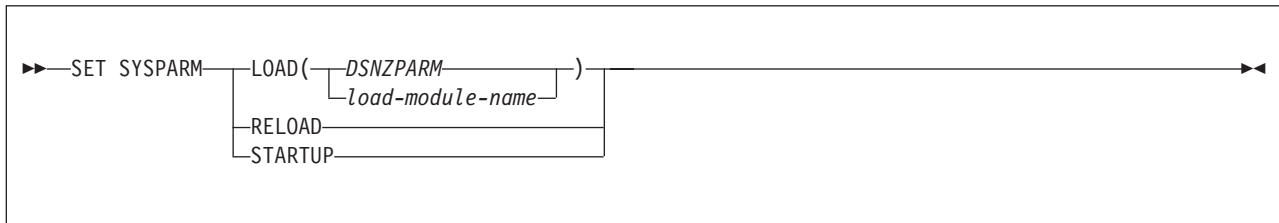
- SYSOPR authority
- SYSCtrl authority
- SYSADM authority
- SECADM authority

To change the following subsystem parameters, you must use a privilege set of the process that includes installation SYSADM authority or SECADM authority:

- SYSADM1
- SYSADM2
- SYSOPR1
- SYSOPR2
- SECADM1
- SECADM2
- SECADM1\_TYPE
- SECADM2\_TYPE
- SEPARATE\_SECURITY
- REVOKE\_DEP\_PRIVILEGES
- AUTHCACH
- BINDNV
- DBACRVW
- EXTSEC
- 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.

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
```

---

## Chapter 74. 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 75. /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 76. -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 execute. 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 using the instrumentation facility interface (IFI).

**Data sharing scope:** Group or local, depending on the SCOPE option.

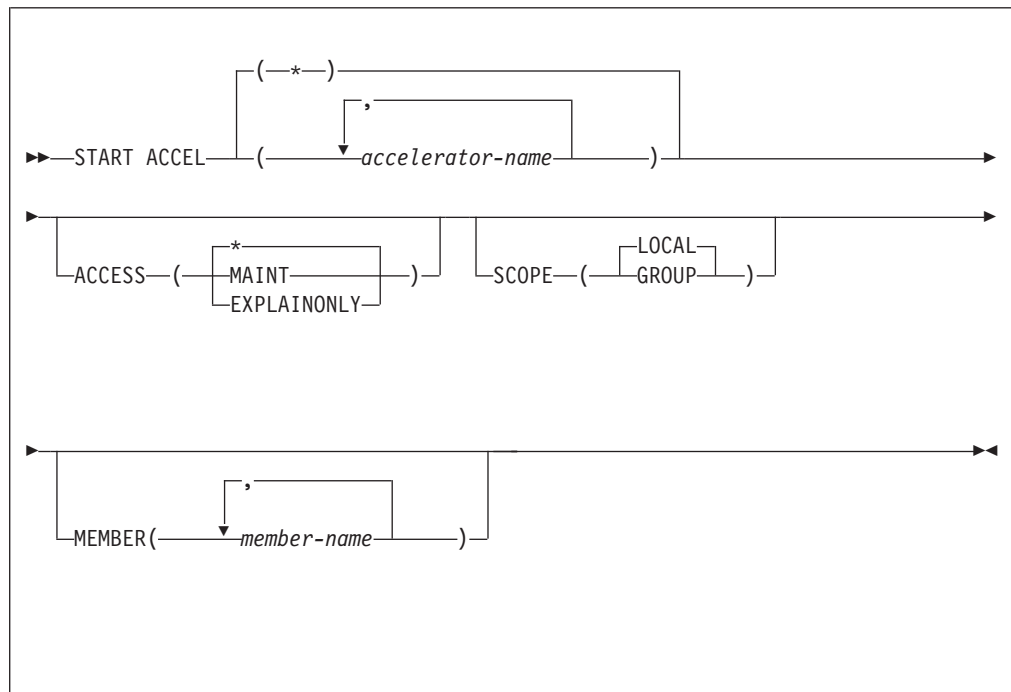
### Authorization

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

## Syntax



### Option descriptions

*(accelerator-name)*

The accelerator server name.

*(\*)*

Indicates that the start command is applied to all accelerator servers.

#### ACCESS

Specifies whether access to accelerator servers is general or restricted. Possible values are:

*(\*)* Makes access general; 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. There are two accepted values.

**(LOCAL)**

Starts the accelerator sserver 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 need to issue START ACCEL when you add a new accelerator server to the SYSACCELERATORS pseudo-catalog table and add IP address information in the SYSACCELIPLIST pseudo-catalog table or when you have modified the information in these tables and issued a -STOP ACCEL command.

If SCOPE(GLOBAL) and MEMBER(member-name) are both specified the command will only be executed on the specified member(s).

You can monitor the status of a started accelerator server using the -DISPLAY ACCEL command.

## Example


Start 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 reference:

 Information about one example of an IBM version 2 accelerator product



---

## Chapter 77. 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

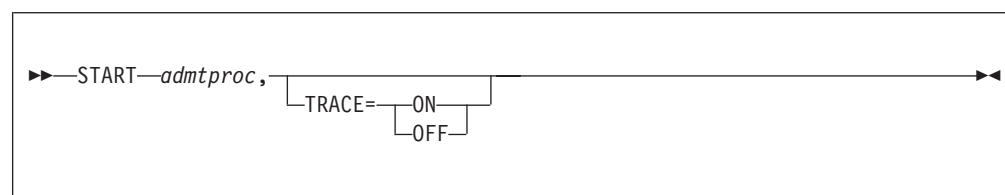
This command can be issued only from a z/OS console.

**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 78. /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

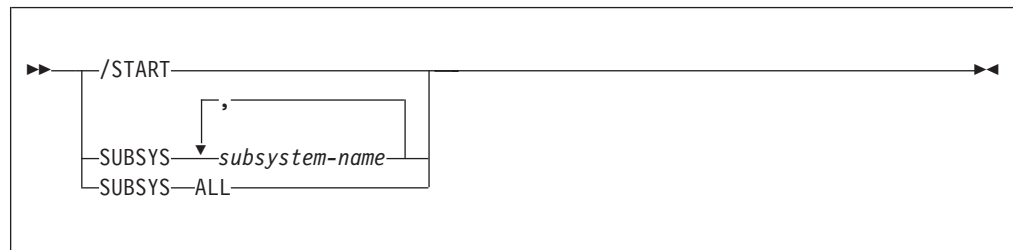
This command can be issued only from an IMS terminal.

**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 79. -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.

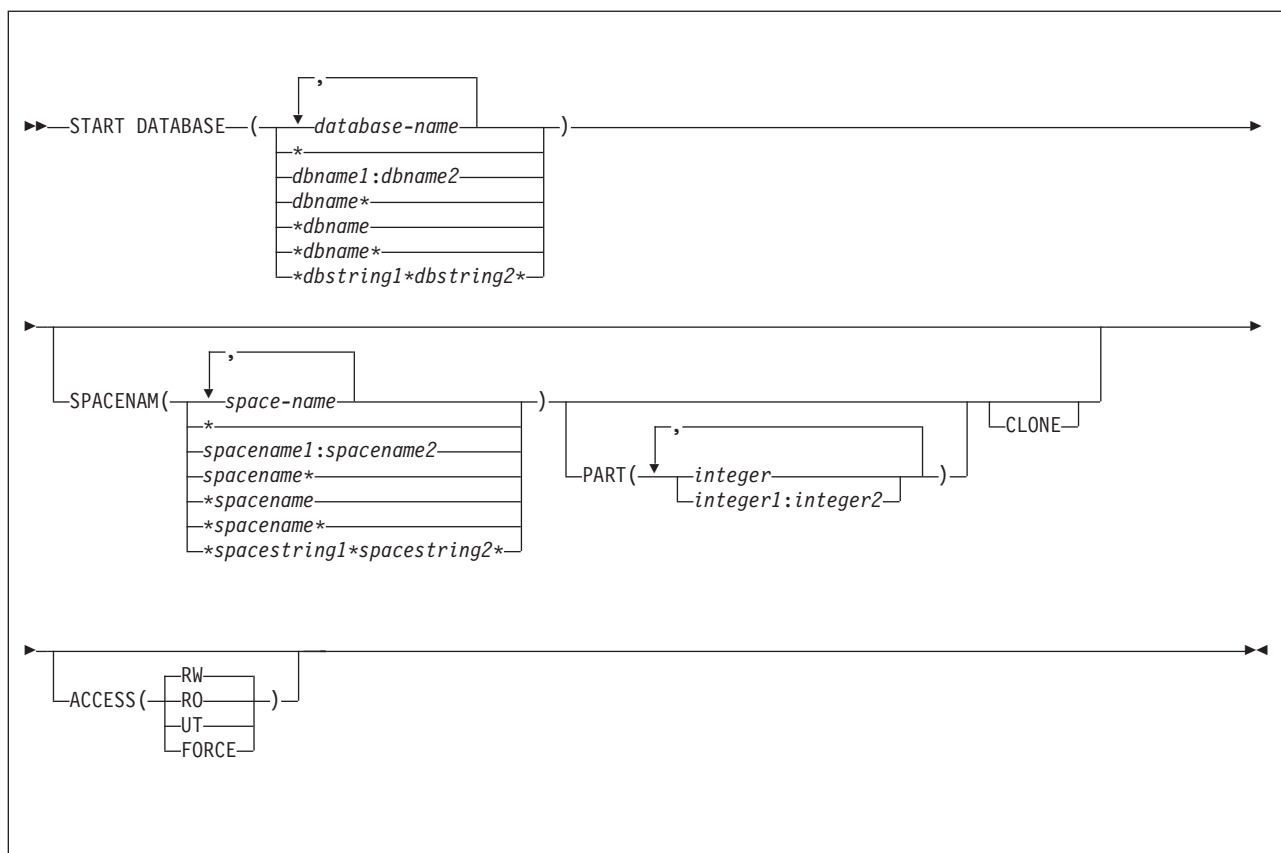
All specified databases with the STARTDB privilege included in the privilege set of the process are started.

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.

### **(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 (RESTD) 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.

**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 DSN1022I 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 DSN1006I 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 DSN1021I 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 80. -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 475.

**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.

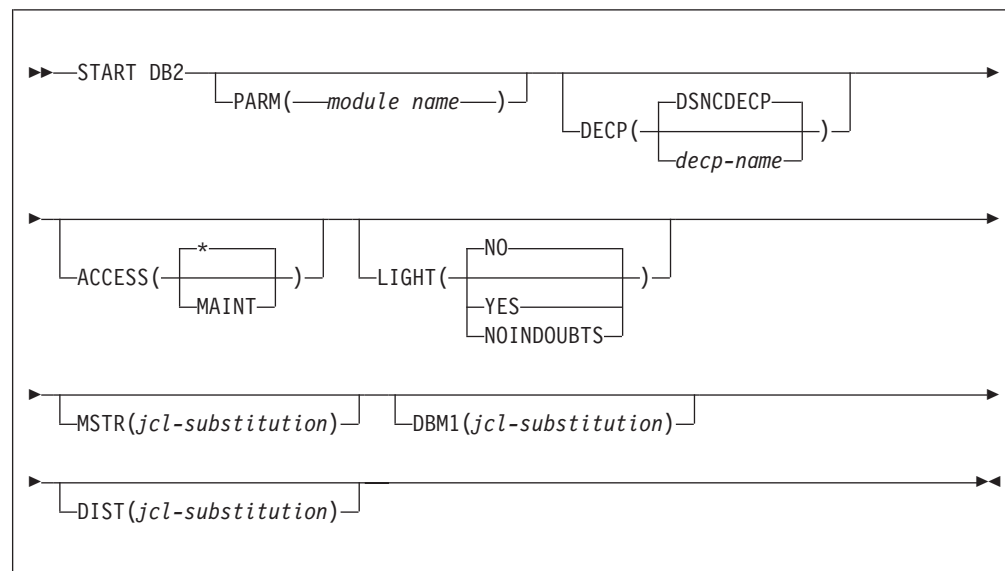
The command is rejected if the DB2 subsystem is already active. The restart recovery status of DB2 resources is determined from the prior DB2 shutdown status.

**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.

**Abbreviation:** NOI

### **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 81. -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 82. -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:

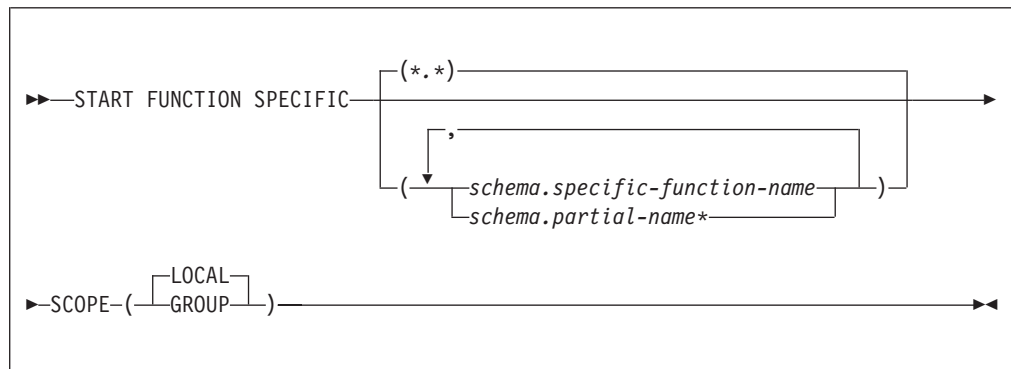
- Ownership of the function
- RECOVER privilege
- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

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 83. 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

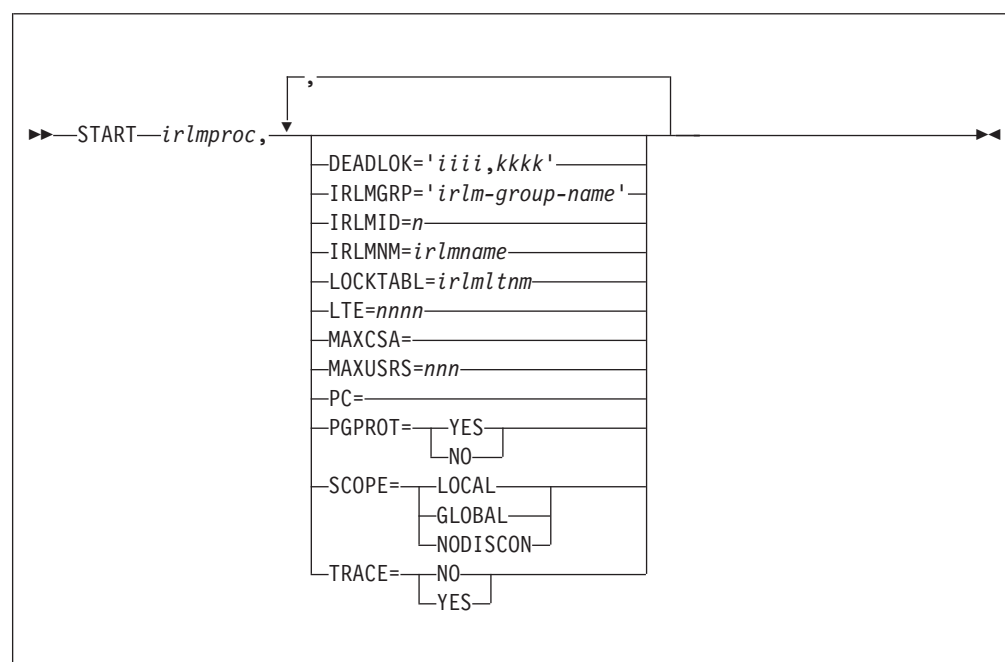
This command can be issued only from a z/OS console.

**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=' iiii, kkkk '**

Specifies the local deadlock-detection interval in seconds ( *iiii* ), and the number of local cycles ( *kkkk* ) that are to occur before a global detection is initiated.

*iiii*

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 26. 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 27. 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

*nnn* 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 84. -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.

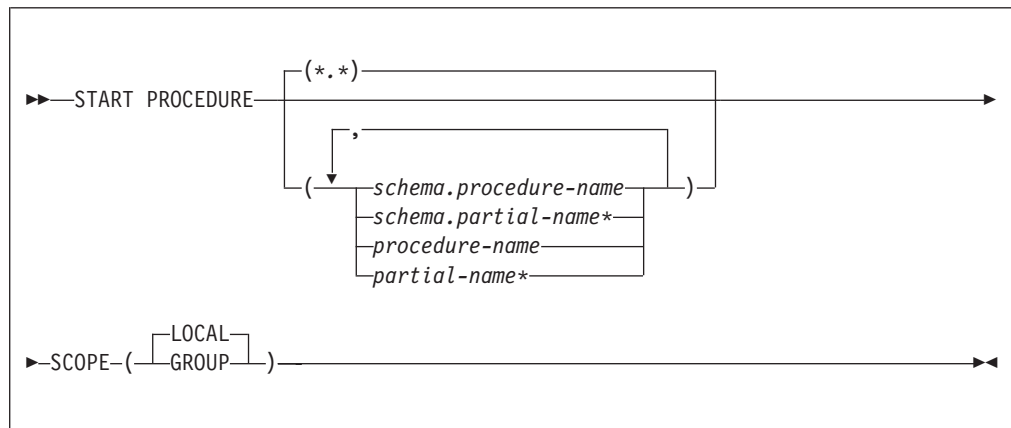
### Authorization

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 85. -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
- SYSCtrl 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 \(DB2 Performance\)](#)

**Related tasks:**

 [Using profiles to monitor and optimize performance \(DB2 Performance\)](#)

 [Optimizing subsystem parameters for SQL statements by using profiles \(DB2 Performance\)](#)

 [Maintaining copies of access paths 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 86. -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. The resource limit specification table or the resource limit middleware table that you identify is used for new threads, and 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

```
»»-START RLIMIT- [ID=id]-<<<
```

### 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 the value that is specified in field RLST NAME SUFFIX on installation panel DSNTIPO.

## 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)


 Limiting resource usage for packages (DB2 Performance)

 Limiting resource usage by client information for middleware servers (DB2 Performance)

### Related reference:

 DSNRLSTxx (DB2 Performance)

 DSNRLMTxx (DB2 Performance)

 Protection panel: DSNTIPP (DB2 Installation and Migration)

## Chapter 87. -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.

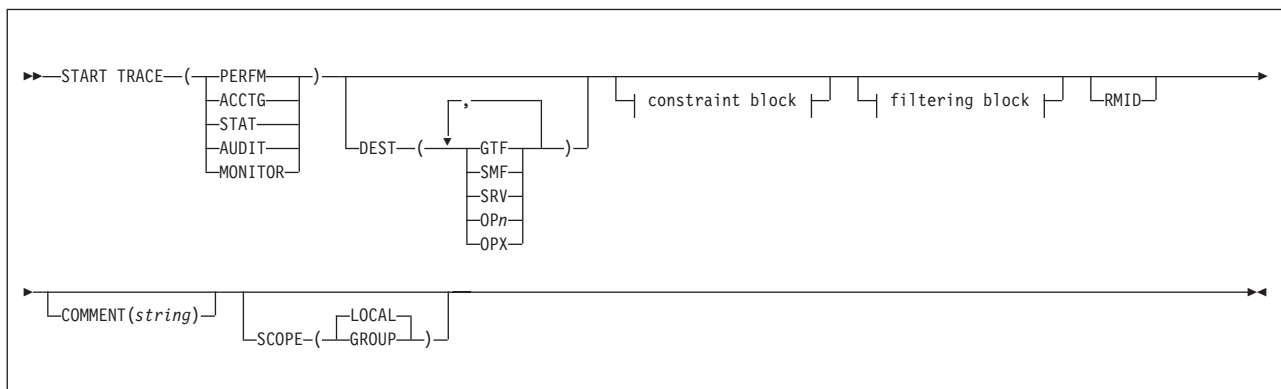
### Authorization

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

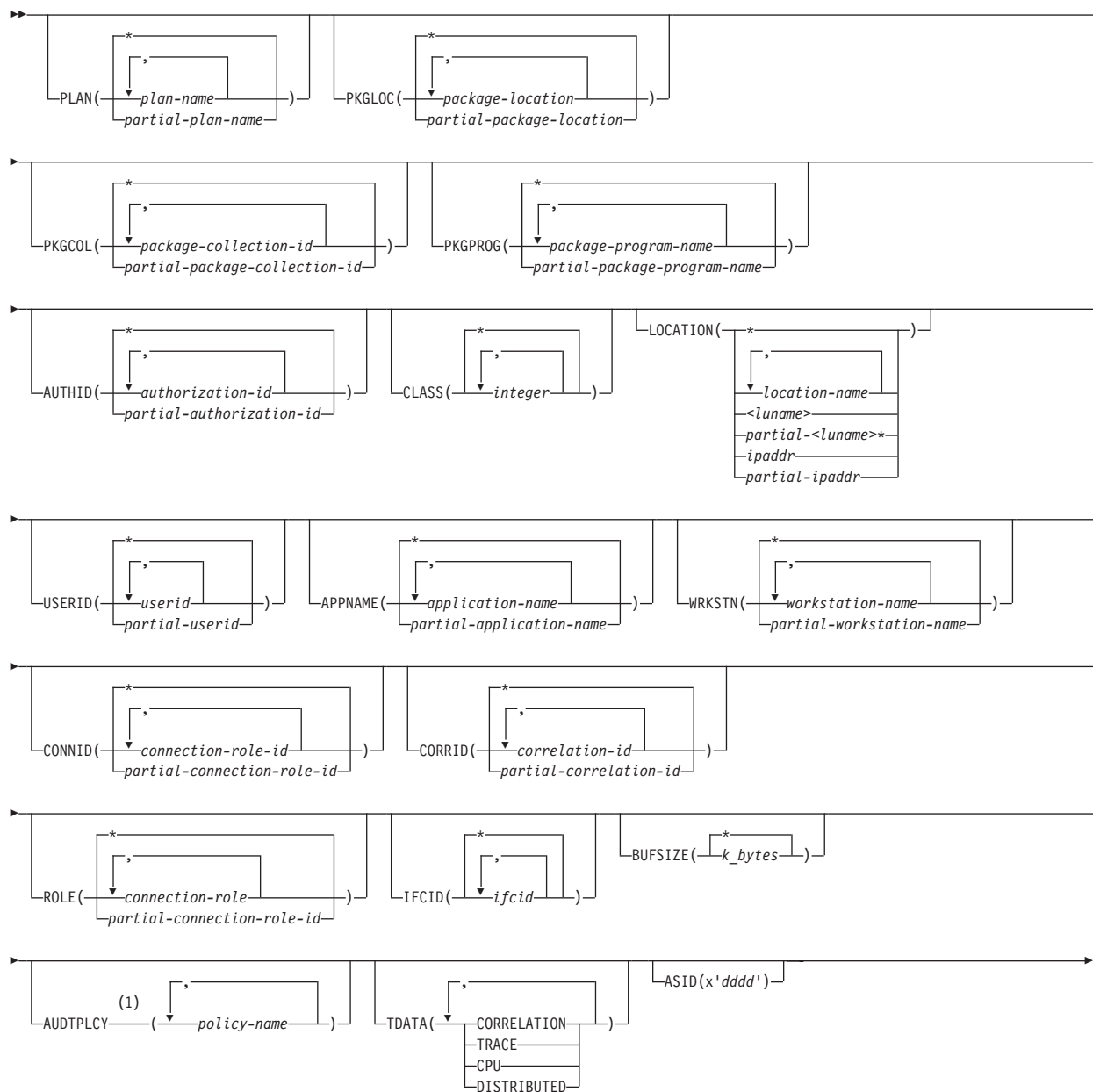
- TRACE privilege
- SQLADM authority
- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority
- SECADM authority

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

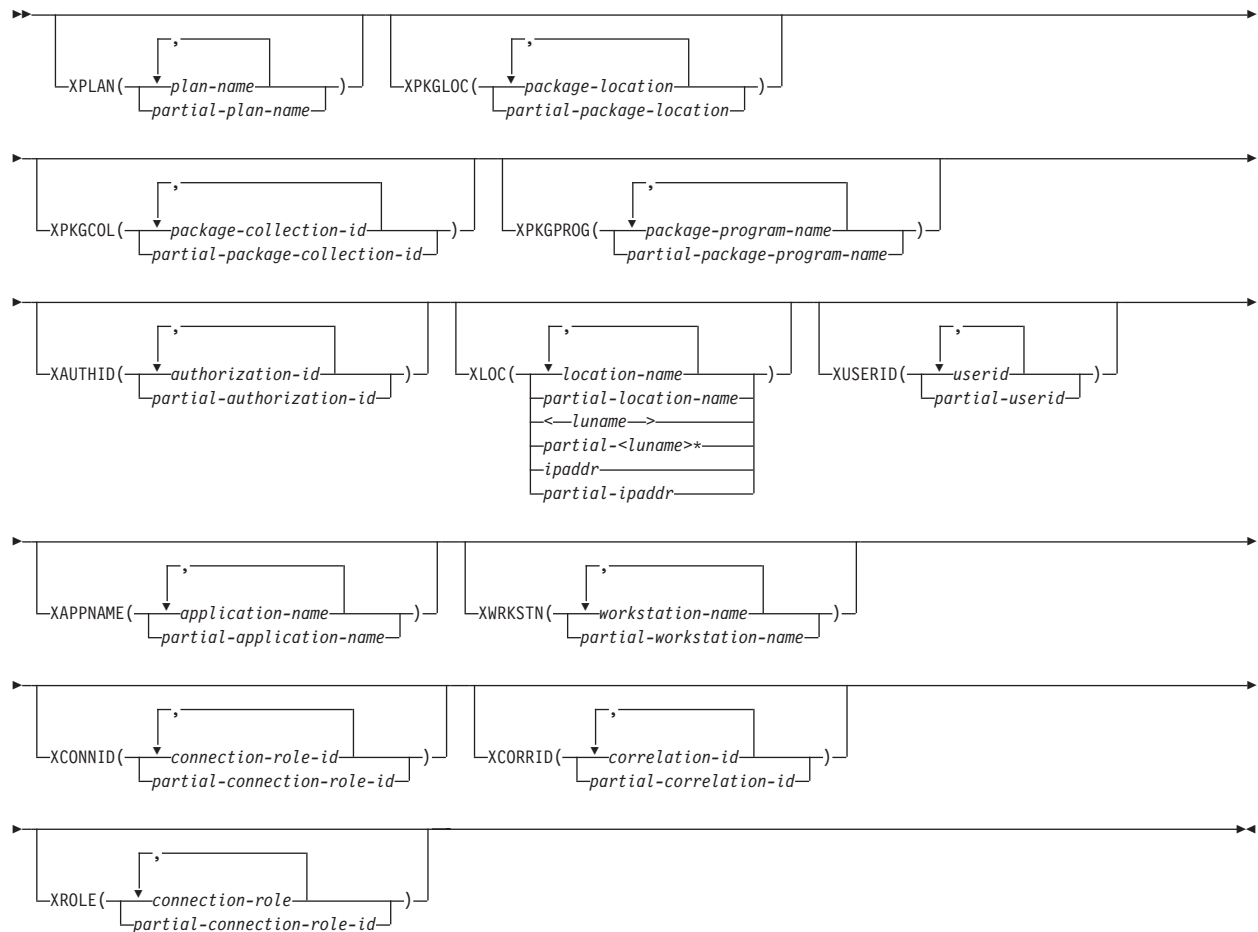


### Notes:

- 1 You cannot specify CLASS or IFCID with AUDTPLCY. AUDTPLCY applies to trace type AUDIT.



## filtering block



## 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* records transaction-level data that is written when the processing for a transaction is completed. Monitor trace is similar to accounting trace. Many classes are identical. However the default destination for the monitor trace is the OPx buffer, which a monitor program can access by an IFI READA call.

**Abbreviation:** MON

### 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.

### ASID(x'dddd')

Specifies the address space for which trace data is collected.

*dddd* 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 OPX 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 28. Allowable destinations for each trace type

Type	GTF	SMF	SRV	OP <i>n</i>	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<sup>®</sup> 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. All other trace types can be started with any constraint option or filter option.

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.

(\*)

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.

**APPNAME or XAPPNAME**

Specifies the application name. Use APPNAME to constrain the trace to the specified applications or XAPPNAME to exclude the specified applications.

**WRKSTN or XWRKSTN**

Specifies the workstation name. Use WRKSTN to constrain the trace to the specified workstations or XWRKSTN to exclude the specified workstations.

**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.

**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 29. 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, 0239
2	Entry or exit from DB2 event signalling.	0200, 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	Entry or exit from DB2 event signalling for package and DBRM accounting.	0200, 0232, 0240
8	Wait time for a package.	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.	0339

Table 29. Classes for DB2 accounting trace (continued)

Class	Description of class	Activated IFCIDs
11 - 29	Reserved.	
30 - 32	Available for local use.	

### Audit trace (AUDIT)

Table 30. 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.	0146
10	Trusted context information.	0269, 0270
11	Audit administrative authorities.	0361
12 - 29	Reserved.	
30 - 32	Available for local use.	

### Statistics trace (STAT)

Table 31. 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

Table 31. Classes for DB2 statistics trace (continued)

Class	Description of class	Activated IFCIDs
7	DRDA location statistics.	0365
8	Data set I/O statistics.	0199
9 - 29	Reserved.	
30 - 32	Available for local use.	

### Performance trace (PERFM)

Table 32. 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
23	Reserved.	



Table 32. Classes for DB2 performance trace (continued)

Class	Description of class	Activated IFCIDs
24	Stored procedure detail.	0380, 0499
25-29	Reserved.	
30 - 32	Available for local use.	

### Monitor trace (MONITOR)

Table 33. 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 signalling.	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 signalling 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 trace must also be activated before the IFCID 0239 records are written: <ul style="list-style-type: none"> <li>• Accounting class 7</li> <li>• Accounting class 8</li> <li>• Monitoring class 7</li> <li>• Monitor class 8</li> </ul>	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:** 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. IFCID 362 record contains information about audit policies that start or fail.

**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. There can be up to 32 traces going at one time. 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 utilize 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 0054 (lock contention). Trace class 30 is available for installation use.

```
-START TRACE (PERFM)
  DEST(GTF)
  LOCATION(USIBMSTODB21)
  CLASS(30)
  IFCID(44)
```

**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 use of DB2 traces \(DB2 Performance\)](#)

**Related reference:**

Chapter 99, “-STOP TRACE (DB2),” on page 549

Chapter 60, “-MODIFY TRACE (DB2),” on page 401

Chapter 38, “-DISPLAY TRACE (DB2),” on page 293





---

## Chapter 88. -STOP ACCEL (DB2)

The DB2 command STOP ACCEL notifies the DB2 subsystem that it should 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 using the instrumentation facility interface (IFI).

**Data sharing scope:** Group or local, depending on the SCOPE option.

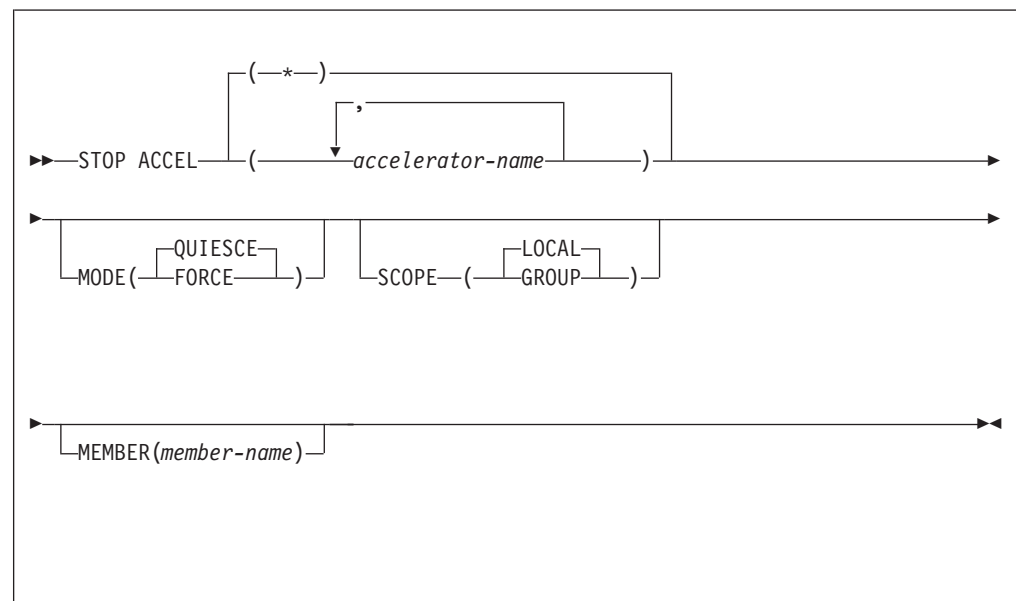
### Authorization

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

### Syntax



### Option descriptions

*(accelerator-name)*

The accelerator server name. This option limits the stop to the specified accelerator server.

(\*)

#### MODE

Indicates whether currently executing active accelerator threads are allowed to complete. There are two accepted values.

##### (QUIESCE)

Allows active accelerator threads to complete normally and terminates only inactive accelerator threads.

##### (FORCE)

Terminates all currently executing accelerator threads. DB2 will revert to baseline processing for the originating threads.

#### SCOPE

Specifies the scope of the command. In a non-data-sharing environment, the option is ignored. There are two accepted values.

##### (LOCAL)

Stops the accelerator server on only the current member.

##### (GROUP)

Stops the accelerator servers on the data sharing group.

#### MEMBER

Restricts the stop to the identified accelerator servers to specific members of the data sharing group. The default is to stop accelerator servers on the local member. In a non-data-sharing environment, the option is ignored.

## Usage notes

If you specify SCOPE(GLOBAL) and MEMBER(member-name), the command will only be executed on the specified members.

## Example

The following command is used to stop all accelerator servers on the current data sharing member and terminate only inactive accelerator threads.

```
-STOP ACCEL(*) MODE(QUIESCE) SCOPE(LOCAL)
```

Sample 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:

 Information about one example of an IBM version 2 accelerator product

---

## Chapter 89. 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 90. /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

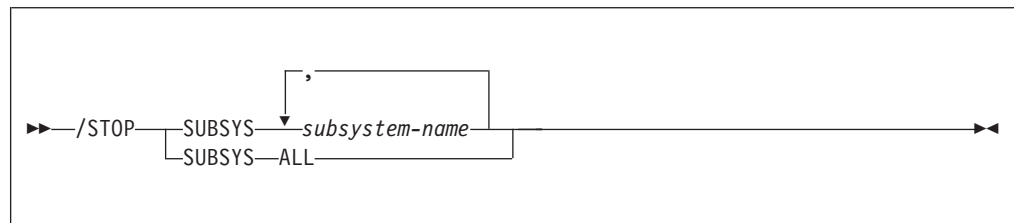
This command can be issued only from an IMS terminal.

**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 91. -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

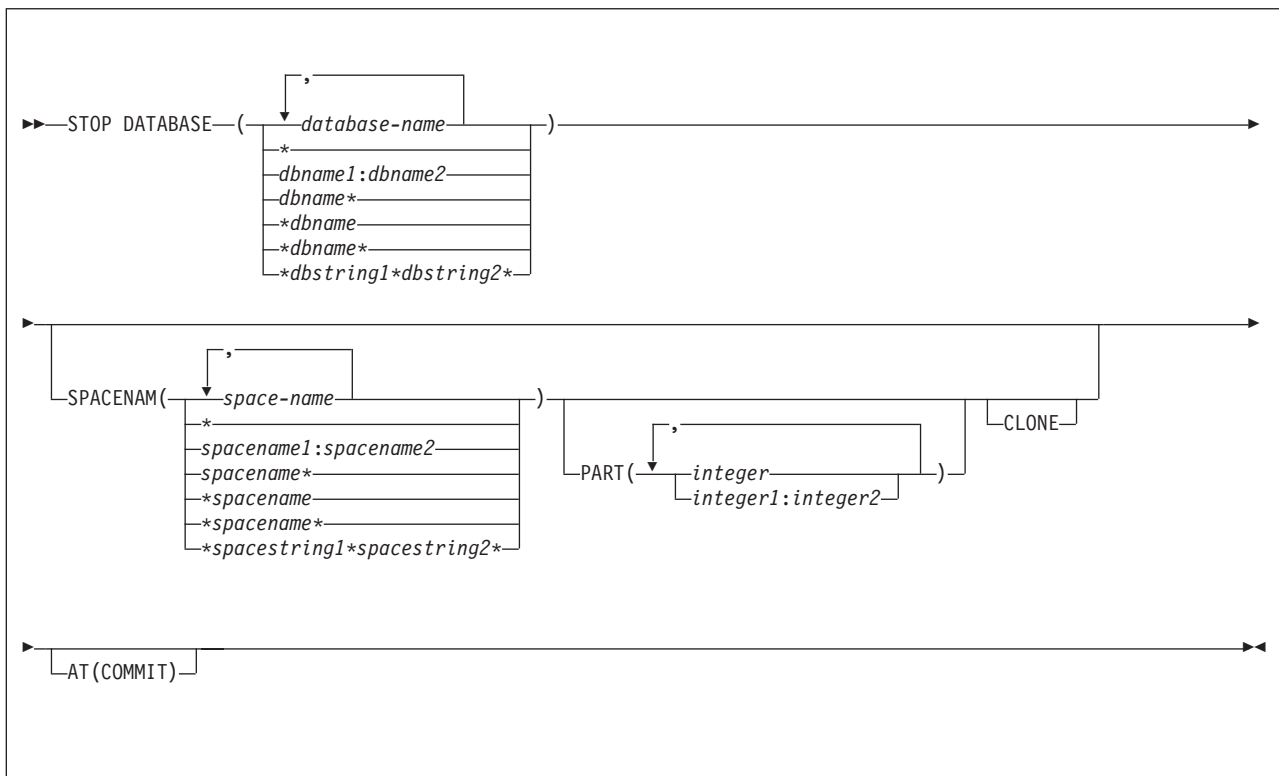
can display information about an implicitly created table space or index space if the command explicitly specifies that table space or index space name.

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.

**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 34. 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 92. -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

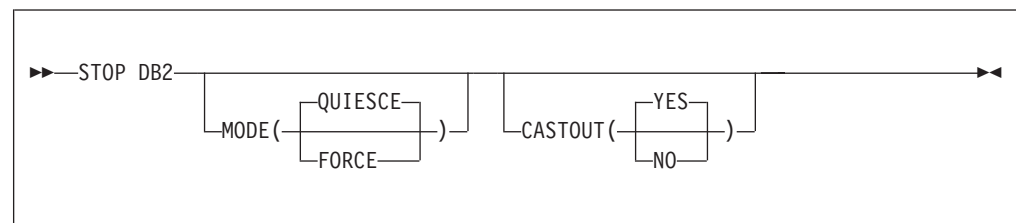
### Authorization

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 Chapter 93, “-STOP DDF (DB2),” on page 531.

#### (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 93. -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

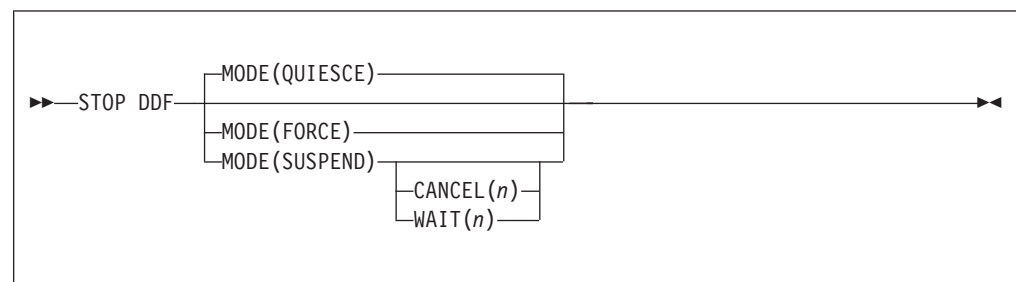
### Authorization

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

- SYSOPR authority
- 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 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 35. 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 94. -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:

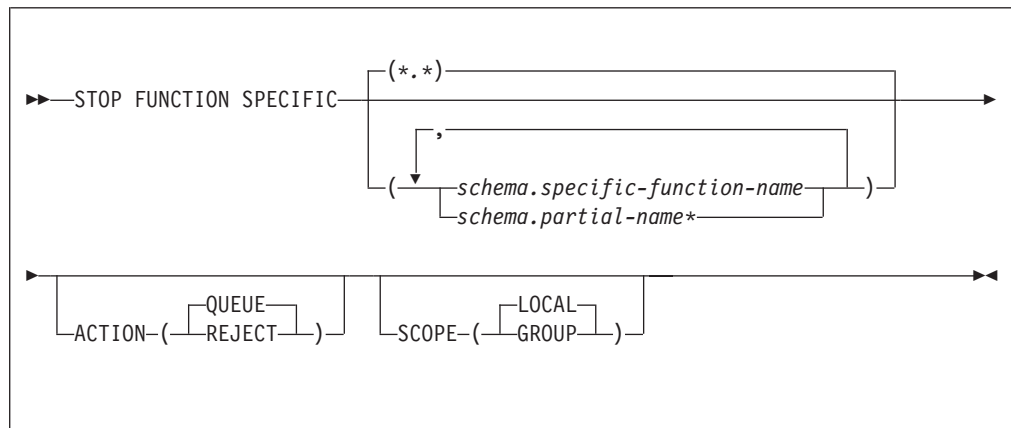
- Ownership of the function
- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

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 95. 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 96. -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:

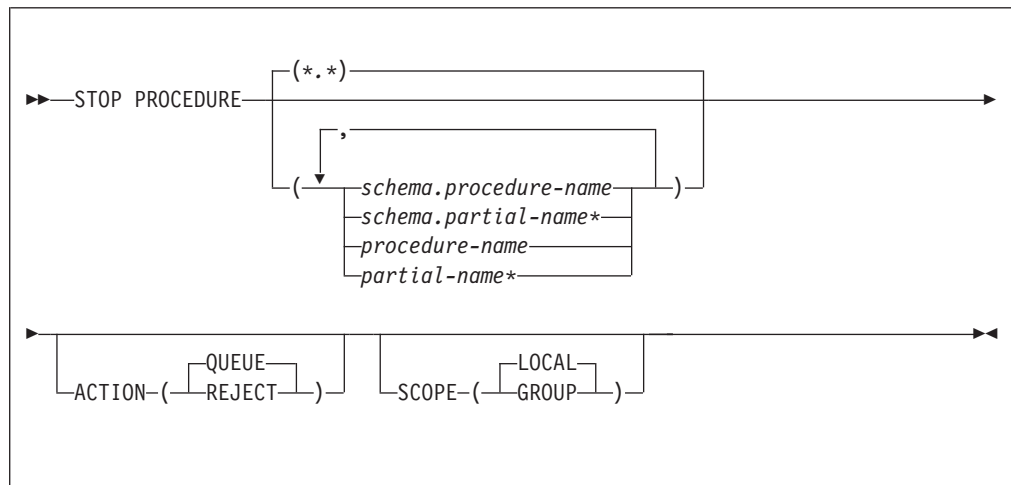
- Ownership of the stored procedure
- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

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 97. STOP PROFILE (DB2)

The STOP PROFILE command 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 \(DB2 Performance\)](#)

#### Related tasks:

 [Creating profiles \(DB2 Performance\)](#)

#### Related reference:

Chapter 85, “-START PROFILE (DB2),” on page 493

 [Profile tables \(DB2 Performance\)](#)





---

## Chapter 98. -STOP RLIMIT (DB2)

The DB2 command STOP RLIMIT stops the resource limit facility. STOP RLIMIT resets all previously set limits to infinity and resets the accumulated time to zero.

All previously limited SQL statements (SELECT, UPDATE, DELETE, and INSERT) executed through an SQL PREPARE or EXECUTE IMMEDIATE statement run with no limit.

**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
```



---

## Chapter 99. -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.

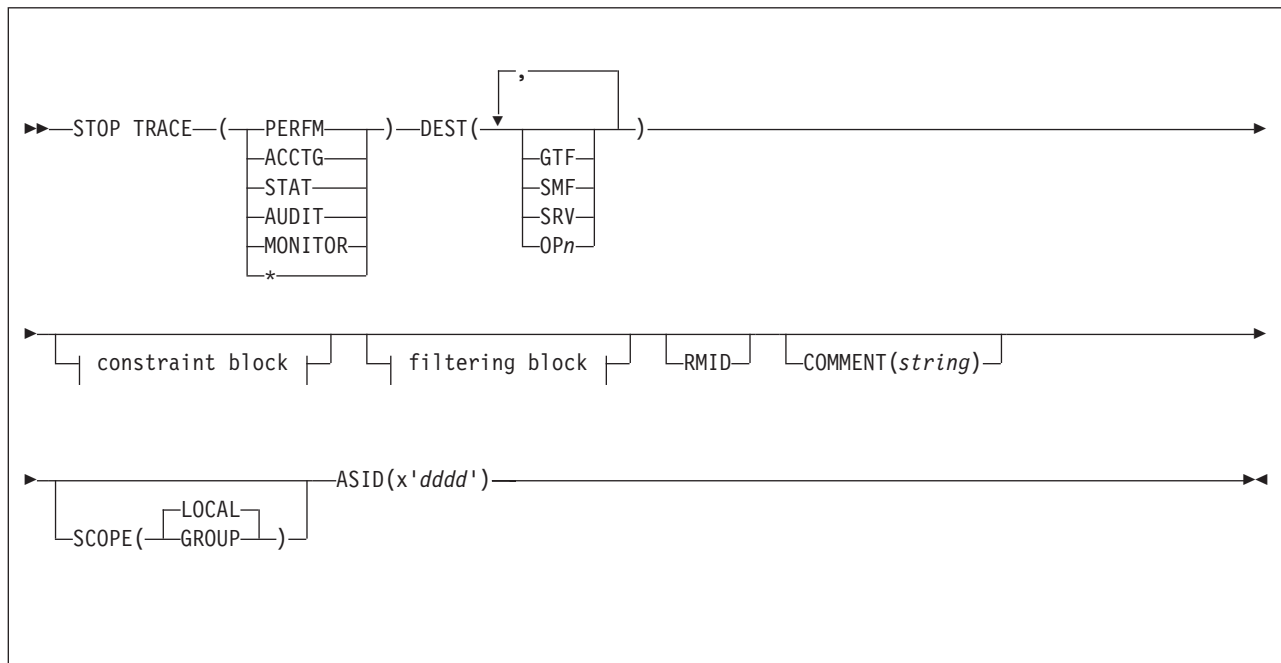
### Authorization

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

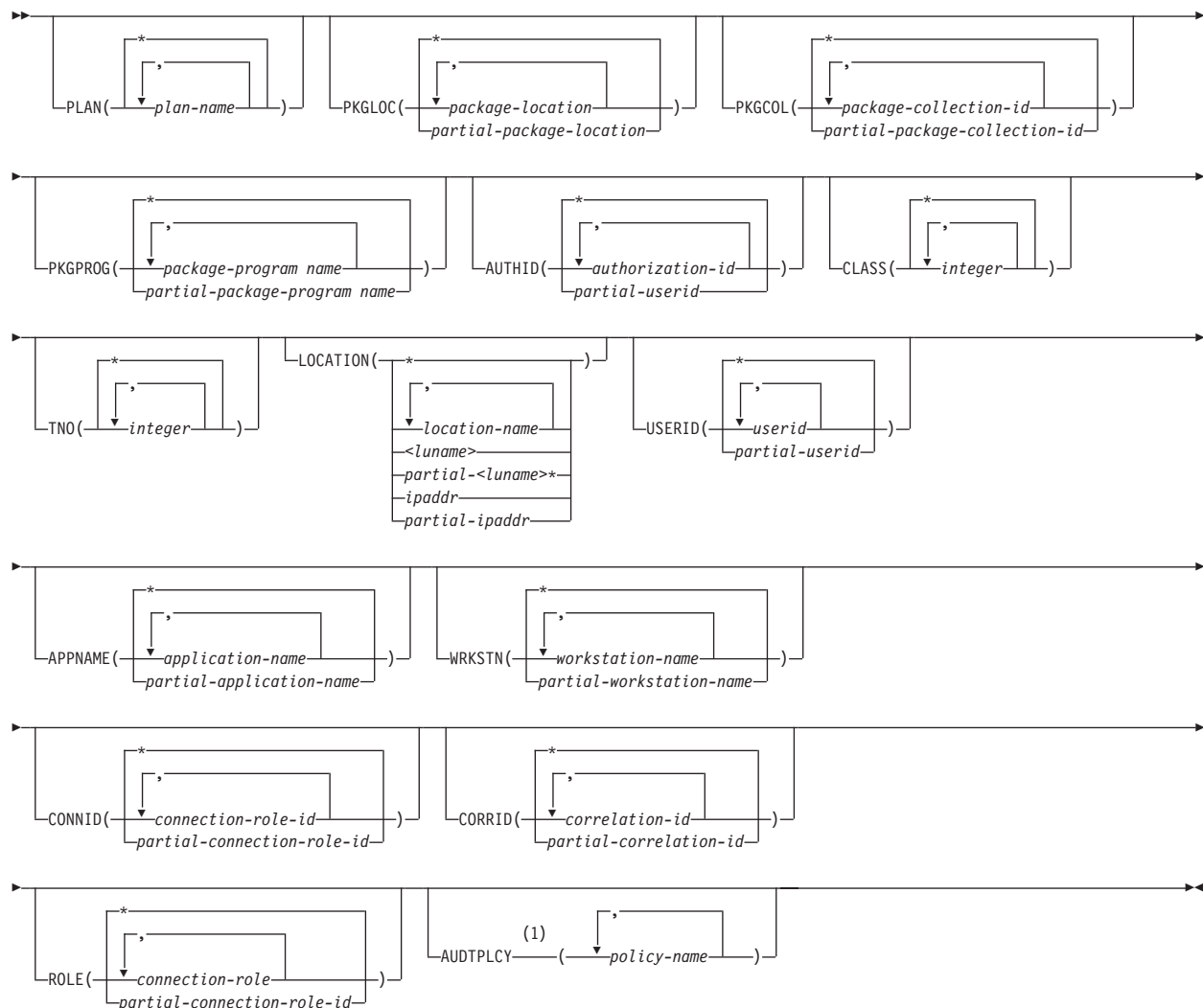
- TRACE privilege
- SQLADM authority
- System DBADM authority
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority
- SECADM authority

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



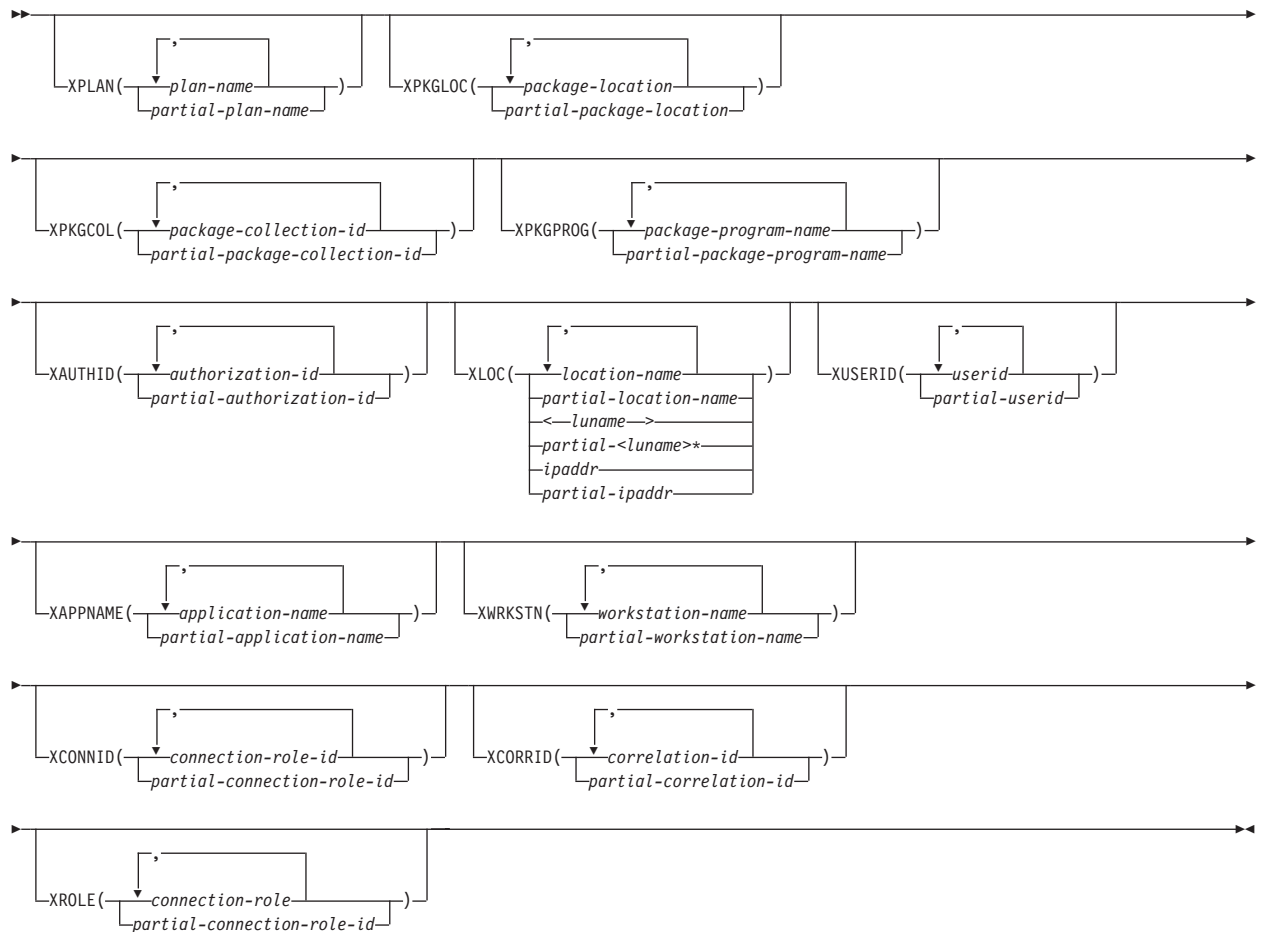
## constraint block



### Notes:

- 1 You cannot specify CLASS or IFCID with AUDTPLCY. AUDTPLCY applies to trace type AUDIT.

## filtering block



## Option descriptions

For additional descriptions of each of the following trace types, see Chapter 87, “-START TRACE (DB2),” on page 497.

### ( 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 556 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'dddd')**

Specifies that the trace for an address space is to be stopped.

*dddd* 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 Chapter 87, “-START TRACE (DB2),” on page 497 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 Chapter 87, “-START TRACE (DB2),” on page 497. 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.

#### **APPNAME or XAPPNAME**

Specifies the application name. Use APPNAME to constrain the trace to the specified applications or XAPPNAME to exclude the specified applications.

#### **WRKSTN or XWRKSTN**

Specifies the workstation name. Use WRKSTN to constrain the trace to the specified workstations or XWRKSTN to exclude the specified workstations.

#### **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 utilize 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 (ADBIOP, 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:

Chapter 38, "-DISPLAY TRACE (DB2)," on page 293

Chapter 60, "-MODIFY TRACE (DB2)," on page 401

Chapter 87, "-START TRACE (DB2)," on page 497

---

## Chapter 100. -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.

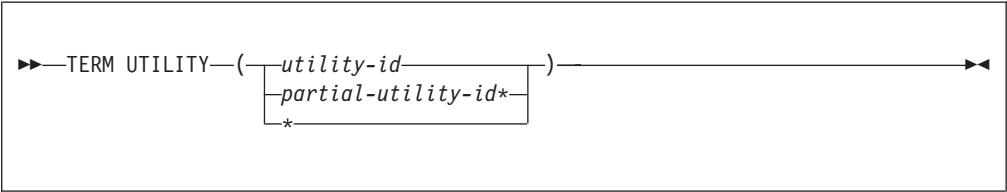
The utilities DIAGNOSE, REPORT, and STOSPACE can be terminated only by the job submitter or by a holder of SYSOPR, SYSCTRL, or SYSADM authority.

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****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****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\*)



---

## Chapter 101. /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.

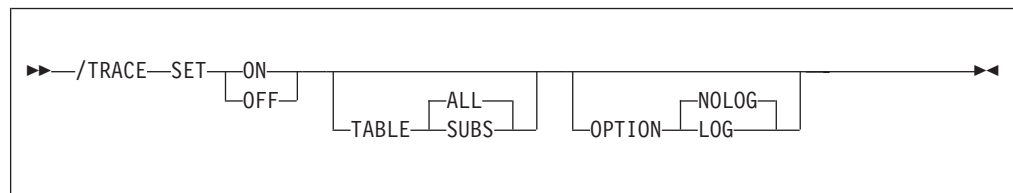
**Data sharing scope:** Member

### 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 102. 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.

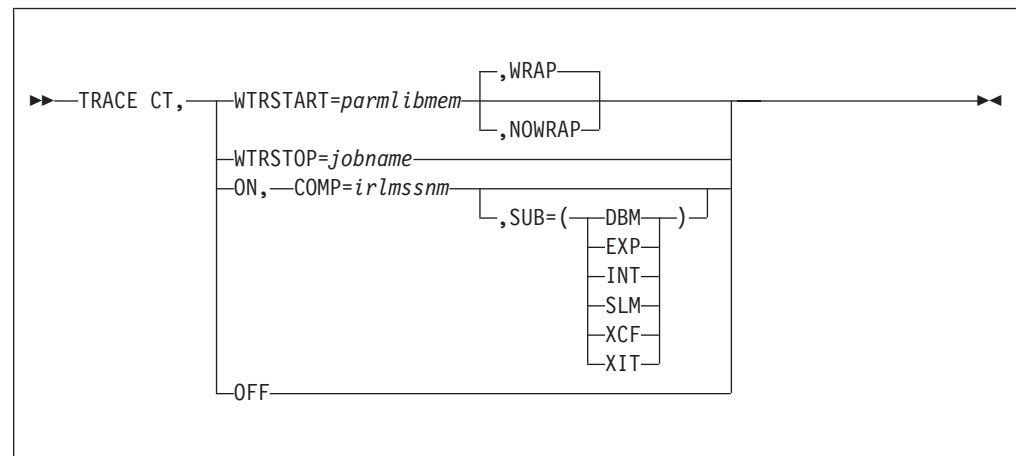
**Data sharing scope:** Member

### 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:**

<b>DBM</b>	Interactions with the identified DBMS
<b>EXP</b>	Any exception condition
<b>INT</b>	Member and group events outside normal locking activity
<b>SLM</b>	Interactions with the z/OS locking component
<b>XCF</b>	All interactions with z/OS cross-system coupling services
<b>XIT</b>	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 4. Appendixes





---

## Information resources for DB2 for z/OS and related products

Many information resources are available to help you use DB2 for z/OS and many related products. A large amount of technical information about IBM products is now available online in information centers or on library websites.

**Disclaimer:** Any web addresses that are included here are accurate at the time this information is being published. However, web addresses sometimes change. If you visit a web address that is listed here but that is no longer valid, you can try to find the current web address for the product information that you are looking for at either of the following sites:

- <http://www.ibm.com/support/publications/us/library/index.shtml>, which lists the IBM information centers that are available for various IBM products
- <http://www.ibm.com/shop/publications/order>, which is the IBM Publications Center, where you can download online PDF books or order printed books for various IBM products

### DB2 for z/OS product information

The primary place to find and use information about DB2 for z/OS is the Information Management Software for z/OS Solutions Information Center (<http://publib.boulder.ibm.com/infocenter/imzic>), which also contains information about IMS, QMF™, and many DB2 and IMS Tools products. This information center is also available as an installable information center that can run on a local system or on an intranet server. You can order the Information Management for z/OS Solutions Information Center DVD (SK5T-7377) for a low cost from the IBM Publications Center (<http://www.ibm.com/shop/publications/order>).

The majority of the DB2 for z/OS information in this information center is also available in the books that are identified in the following table. You can access these books at the DB2 for z/OS library website (<http://www.ibm.com/software/data/db2/zos/library.html>) or at the IBM Publications Center (<http://www.ibm.com/shop/publications/order>).

Table 36. DB2 10 for z/OS book titles

Title	Publication number	Available in information center	Available in PDF	Available in printed format
<i>DB2 10 for z/OS Administration Guide</i>	SC19-2968	X	X	
<i>DB2 10 for z/OS Application Programming &amp; SQL Guide</i>	SC19-2969	X	X	
<i>DB2 10 for z/OS Application Programming Guide and Reference for Java</i>	SC19-2970	X	X	
<i>DB2 10 for z/OS Codes</i>	GC19-2971	X	X	
<i>DB2 10 for z/OS Command Reference</i>	SC19-2972	X	X	
<i>DB2 10 for z/OS Data Sharing: Planning and Administration</i>	SC19-2973	X	X	
<i>DB2 10 for z/OS Diagnosis Guide and Reference</i>	LY37-3220		X	X

Table 36. DB2 10 for z/OS book titles (continued)

Title	Publication number	Available in information center	Available in PDF	Available in printed format
DB2 10 for z/OS Installation and Migration Guide	GC19-2974	X	X	
DB2 10 for z/OS Internationalization Guide	SC19-2975	X	X	
DB2 10 for z/OS Introduction to DB2	SC19-2976	X	X	
DB2 10 for z/OS Licensed Program Specifications	GC19-2977		X	X
DB2 10 for z/OS Managing Performance	SC19-2978	X	X	
DB2 10 for z/OS Managing Security	SC19-3496	X	X	
DB2 10 for z/OS Messages	GC19-2979	X	X	
DB2 10 for z/OS ODBC Guide and Reference	SC19-2980	X	X	
DB2 10 for z/OS Program Directory	GI10-8829		X	X
DB2 10 for z/OS pureXML Guide	SC19-2981	X	X	
DB2 10 for z/OS RACF Access Control Module Guide	SC19-2982	X	X	
DB2 10 for z/OS SQL Reference	SC19-2983	X	X	
DB2 10 for z/OS Utility Guide and Reference	SC19-2984	X	X	
DB2 10 for z/OS What's New?	GC19-2985	X	X	
IRLM Messages and Codes for IMS and DB2 for z/OS	GC19-2666	X	X	

**Note:**

1. DB2 10 for z/OS Diagnosis Guide and Reference is available in PDF format on the DB2 10 for z/OS Licensed Library Collection kit, LK5T-7390. You can order this Licensed Library Collection kit on the IBM Publications Center site (<http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss>). This book is also available in online format in DB2 data set DSN10.SDSNIVPD(DSNDR).

## Information resources for related products

In the following table, related product names are listed in alphabetic order, and the associated web addresses of product information centers or library web pages are indicated.

Table 37. Related product information resource locations

Related product	Information resources
C/C++ for z/OS	Library website: <a href="http://www.ibm.com/software/awdtools/czos/library/">http://www.ibm.com/software/awdtools/czos/library/</a>  This product is now called z/OS XL C/C++.
CICS Transaction Server for z/OS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp">http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp</a>
COBOL	Information center: <a href="http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp">http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp</a>  This product is now called Enterprise COBOL for z/OS.
DB2 Connect™	Information center: <a href="http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp">http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp</a>  This resource is for DB2 Connect 9.

Table 37. Related product information resource locations (continued)

Related product	Information resources
DB2 Database for Linux, UNIX, and Windows	Information center: <a href="http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp">http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp</a> This resource is for DB2 9 for Linux, UNIX, and Windows.
DB2 Query Management Facility™	Information center: <a href="http://publib.boulder.ibm.com/infocenter/imzic">http://publib.boulder.ibm.com/infocenter/imzic</a>
DB2 Server for VSE & VM	Product website: <a href="http://www.ibm.com/software/data/db2/vse-vm/">http://www.ibm.com/software/data/db2/vse-vm/</a>
DB2 Tools	One of the following locations: <ul style="list-style-type: none"> <li>• Information center: <a href="http://publib.boulder.ibm.com/infocenter/imzic">http://publib.boulder.ibm.com/infocenter/imzic</a></li> <li>• Library website: <a href="http://www.ibm.com/software/data/db2imstools/library.html">http://www.ibm.com/software/data/db2imstools/library.html</a></li> </ul> <p>These resources include information about the following products and others:</p> <ul style="list-style-type: none"> <li>• DB2 Administration Tool</li> <li>• DB2 Automation Tool</li> <li>• DB2 Log Analysis Tool</li> <li>• DB2 Object Restore Tool</li> <li>• DB2 Query Management Facility</li> <li>• DB2 SQL Performance Analyzer</li> </ul>
DB2 Universal Database™ for iSeries®	Information center: <a href="http://www.ibm.com/systems/i/infocenter/">http://www.ibm.com/systems/i/infocenter/</a>
Debug Tool for z/OS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp">http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp</a>
Enterprise COBOL for z/OS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp">http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp</a>
Enterprise PL/I for z/OS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp">http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp</a>
InfoSphere® Replication Server for z/OS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.swg.im.iis.db.prod.repl.nav.doc/dochome/iisrcnav_dochome.html">http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.swg.im.iis.db.prod.repl.nav.doc/dochome/iisrcnav_dochome.html</a>  This product was also known as DB2 DataPropagator, DB2 Information Integrator Replication Edition for z/OS, and WebSphere Replication Server for z/OS.
IMS	Information center: <a href="http://publib.boulder.ibm.com/infocenter/imzic">http://publib.boulder.ibm.com/infocenter/imzic</a>
IMS Tools	One of the following locations: <ul style="list-style-type: none"> <li>• Information center: <a href="http://publib.boulder.ibm.com/infocenter/imzic">http://publib.boulder.ibm.com/infocenter/imzic</a></li> <li>• Library website: <a href="http://www.ibm.com/software/data/db2imstools/library.html">http://www.ibm.com/software/data/db2imstools/library.html</a></li> </ul> <p>These resources have information about the following products and others:</p> <ul style="list-style-type: none"> <li>• IMS Batch Terminal Simulator for z/OS</li> <li>• IMS Connect</li> <li>• IMS HALDB Conversion and Maintenance Aid</li> <li>• IMS High Performance Utility products</li> <li>• IMS DataPropagator</li> <li>• IMS Online Reorganization Facility</li> <li>• IMS Performance Analyzer</li> </ul>

Table 37. Related product information resource locations (continued)

Related product	Information resources
Integrated Data Management products	<p>Information center: <a href="http://publib.boulder.ibm.com/infocenter/idm/v2r2/index.jsp">http://publib.boulder.ibm.com/infocenter/idm/v2r2/index.jsp</a></p> <p>This information center has information about the following products and others:</p> <ul style="list-style-type: none"> <li>• IBM Data Studio</li> <li>• InfoSphere Data Architect</li> <li>• InfoSphere Warehouse</li> <li>• Optim<sup>™</sup> Database Administrator</li> <li>• Optim Development Studio</li> <li>• Optim Query Tuner</li> </ul>
PL/I	<p>Information center: <a href="http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp">http://publib.boulder.ibm.com/infocenter/pdthelp/v1r1/index.jsp</a></p> <p>This product is now called Enterprise PL/I for z/OS.</p>
System z <sup>®</sup>	<a href="http://publib.boulder.ibm.com/infocenter/eserver/v1r2/index.jsp">http://publib.boulder.ibm.com/infocenter/eserver/v1r2/index.jsp</a>
Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS	<p>Information center: <a href="http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.omegamon.xe_db2.doc/ko2welcome_pe.htm">http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.omegamon.xe_db2.doc/ko2welcome_pe.htm</a></p> <p>In earlier releases, this product was called DB2 Performance Expert for z/OS.</p>
WebSphere Application Server	Information center: <a href="http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp">http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp</a>
WebSphere Message Broker with Rules and Formatter Extension	<p>Information center: <a href="http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp">http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r0m0/index.jsp</a></p> <p>The product is also known as WebSphere MQ Integrator Broker.</p>
WebSphere MQ	<p>Information center: <a href="http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp">http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp</a></p> <p>The resource includes information about MQSeries<sup>®</sup>.</p>
z/Architecture <sup>®</sup>	Library Center site: <a href="http://www.ibm.com/servers/eserver/zseries/zos/bkserv/">http://www.ibm.com/servers/eserver/zseries/zos/bkserv/</a>

Table 37. Related product information resource locations (continued)

Related product	Information resources
z/OS	<p>Library Center site: <a href="http://www.ibm.com/servers/eserver/zseries/zos/bkserv/">http://www.ibm.com/servers/eserver/zseries/zos/bkserv/</a></p> <p>This resource includes information about the following z/OS elements and components:</p> <ul style="list-style-type: none"> <li>• Character Data Representation Architecture</li> <li>• Device Support Facilities</li> <li>• DFSORT</li> <li>• Fortran</li> <li>• High Level Assembler</li> <li>• NetView®</li> <li>• SMP/E for z/OS</li> <li>• SNA</li> <li>• TCP/IP</li> <li>• TotalStorage Enterprise Storage Server®</li> <li>• VTAM</li> <li>• z/OS C/C++</li> <li>• z/OS Communications Server</li> <li>• z/OS DCE</li> <li>• z/OS DFSMS</li> <li>• z/OS DFSMS Access Method Services</li> <li>• z/OS DFSMSdss</li> <li>• z/OS DFSMSHsm</li> <li>• z/OS DFSMSdftp</li> <li>• z/OS ICSF</li> <li>• z/OS ISPF</li> <li>• z/OS JES3</li> <li>• z/OS Language Environment</li> <li>• z/OS Managed System Infrastructure</li> <li>• z/OS MVS</li> <li>• z/OS MVS JCL</li> <li>• z/OS Parallel Sysplex®</li> <li>• z/OS RMF</li> <li>• z/OS Security Server</li> <li>• z/OS UNIX System Services</li> </ul>
z/OS XL C/C++	<p><a href="http://www.ibm.com/software/awdtools/czos/library/">http://www.ibm.com/software/awdtools/czos/library/</a></p>

The following information resources from IBM are not necessarily specific to a single product:

- The DB2 for z/OS Information Roadmap; available at: <http://www.ibm.com/software/data/db2/zos/roadmap.html>
- DB2 Redbooks® and Redbooks about related products; available at: <http://www.ibm.com/redbooks>
- IBM Educational resources:
  - Information about IBM educational offerings is available on the web at: <http://www.ibm.com/software/sw-training/>

- A collection of glossaries of IBM terms in multiple languages is available on the IBM Terminology website at: <http://www.ibm.com/software/globalization/terminology/index.jsp>
- National Language Support information; available at the IBM Publications Center at: <http://www.elink.ibm.link.ibm.com/public/applications/publications/cgi-bin/pbi.cgi>
- *SQL Reference for Cross-Platform Development*; available at the following developerWorks® site: <http://www.ibm.com/developerworks/db2/library/techarticle/0206sqlref/0206sqlref.html>

The following information resources are not published by IBM but can be useful to users of DB2 for z/OS and related products:

- Database design topics:
  - *DB2 for z/OS and OS/390® Development for Performance Volume I*, by Gabrielle Wiorkowski, Gabrielle & Associates, ISBN 0-96684-605-2
  - *DB2 for z/OS and OS/390 Development for Performance Volume II*, by Gabrielle Wiorkowski, Gabrielle & Associates, ISBN 0-96684-606-0
  - *Handbook of Relational Database Design*, by C. Fleming and B. Von Halle, Addison Wesley, ISBN 0-20111-434-8
- Distributed Relational Database Architecture™ (DRDA) specifications; <http://www.opengroup.org>
- Domain Name System: *DNS and BIND*, Third Edition, Paul Albitz and Cricket Liu, O'Reilly, ISBN 0-59600-158-4
- Microsoft Open Database Connectivity (ODBC) information; <http://msdn.microsoft.com/library/>
- Unicode information; <http://www.unicode.org>

---

## How to obtain DB2 information

You can access the official information about the DB2 product in a number of ways.

- “DB2 on the web”
- “DB2 product information”
- “DB2 education” on page 578
- “How to order the DB2 library” on page 578

### DB2 on the web

Stay current with the latest information about DB2 by visiting the DB2 home page on the web:

<http://www.ibm.com/software/db2zos>

On the DB2 home page, you can find links to a wide variety of information resources about DB2. You can read news items that keep you informed about the latest enhancements to the product. Product announcements, press releases, fact sheets, and technical articles help you plan and implement your database management strategy.

### DB2 product information

The official DB2 for z/OS information is available in various formats and delivery methods. IBM provides mid-version updates to the information in the information center and in softcopy updates that are available on the web and on CD-ROM.

#### Information Management Software for z/OS Solutions Information Center

DB2 product information is viewable in the information center, which is the primary delivery vehicle for information about DB2 for z/OS, IMS, QMF, and related tools. This information center enables you to search across related product information in multiple languages for data management solutions for the z/OS environment and print individual topics or sets of related topics. You can also access, download, and print PDFs of the publications that are associated with the information center topics. Product technical information is provided in a format that offers more options and tools for accessing, integrating, and customizing information resources. The information center is based on Eclipse open source technology.

The Information Management Software for z/OS Solutions Information Center is viewable at the following website:

<http://publib.boulder.ibm.com/infocenter/imzic>

#### CD-ROMs and DVD

Books for DB2 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.

The books for DB2 for z/OS are also available on the following DVD collection kit, which contains online books for many IBM products:

- IBM z/OS Software Products DVD Collection, SK3T-4271, in English

#### **PDF format**

Many of the DB2 books are available in PDF (Portable Document Format) for viewing or printing from CD-ROM or the DB2 home page on the web or from the information center. Download the PDF books to your intranet for distribution throughout your enterprise.

#### **DB2 education**

IBM Education and Training offers a wide variety of classroom courses to help you quickly and efficiently gain DB2 expertise. IBM schedules classes in cities all over the world. You can find class information, by country, at the IBM Learning Services website:

<http://www.ibm.com/services/learning>

IBM also offers classes at your location, at a time that suits your needs. IBM can customize courses to meet your exact requirements. For more information, including the current local schedule, contact your IBM representative.

#### **How to order the DB2 library**

To order books, visit the IBM Publication Center on the web:

<http://www.ibm.com/shop/publications/order>

From the IBM Publication Center, you can go to the Publication Notification System (PNS). PNS users receive electronic notifications of updated publications in their profiles. You have the option of ordering the updates by using the publications direct ordering application or any other IBM publication ordering channel. The PNS application does not send automatic shipments of publications. You will receive updated publications and a bill for them if you respond to the electronic notification.

You can also order DB2 publications and CD-ROMs from your IBM representative or the IBM branch office that serves your locality. If your location is within the United States or Canada, you can place your order by calling one of the toll-free numbers:

- In the U.S., call 1-800-879-2755.
- In Canada, call 1-800-426-4968.

To order additional copies of licensed publications, specify the SOFTWARE option. To order additional publications or CD-ROMs, specify the PUBLICATIONS option. Be prepared to give your customer number, the product number, and either the feature codes or order numbers that you want.



---

## How to use the DB2 library

Titles of books in the library begin with DB2 10 for z/OS. However, references from one book in the library to another are shortened and do not include the product name, version, and release. Instead, they point directly to the section that holds the information. The primary place to find and use information about DB2 for z/OS is the Information Management Software for z/OS Solutions Information Center (<http://publib.boulder.ibm.com/infocenter/imzic>).

If you are new to DB2 for z/OS, *Introduction to DB2 for z/OS* provides a comprehensive introduction to DB2 10 for z/OS. Topics included in this book explain the basic concepts that are associated with relational database management systems in general, and with DB2 for z/OS in particular.

The most rewarding task associated with a database management system is asking questions of it and getting answers, the task called *end use*. Other tasks are also necessary—defining the parameters of the system, putting the data in place, and so on. The tasks that are associated with DB2 are grouped into the following major categories.

### Installation

If you are involved with installing DB2, you will need to use a variety of resources, such as:

- *DB2 Program Directory*
- *DB2 Installation and Migration Guide*
- *DB2 Administration Guide*
- *DB2 Application Programming Guide and Reference for Java*
- *DB2 Codes*
- *DB2 Internationalization Guide*
- *DB2 Messages*
- *DB2 Managing Performance*
- *DB2 Managing Security*
- *DB2 RACF Access Control Module Guide*
- *DB2 Utility Guide and Reference*

If you will be using data sharing capabilities you also need *DB2 Data Sharing: Planning and Administration*, which describes installation considerations for data sharing.

If you will be installing and configuring DB2 ODBC, you will need *DB2 ODBC Guide and Reference*.

If you are installing IBM Spatial Support for DB2 for z/OS, you will need *IBM Spatial Support for DB2 for z/OS User's Guide and Reference*.

If you are enabling the text search feature for DB2 for z/OS, you will need *IBM Text Search for DB2 for z/OS Installation, Administration, and Reference*.

## End use

End users issue SQL statements to retrieve data. They can also insert, update, or delete data, with SQL statements. They might need an introduction to SQL, detailed instructions for using SPUI, and an alphabetized reference to the types of SQL statements. This information is found in *DB2 Application Programming and SQL Guide*, and *DB2 SQL Reference*.

End users can also issue SQL statements through the DB2 Query Management Facility (QMF) or some other program, and the library for that licensed program might provide all the instruction or reference material they need.

## Application programming

Some users access DB2 without knowing it, using programs that contain SQL statements. DB2 application programmers write those programs. Because they write SQL statements, they need the same resources that end users do.

Application programmers also need instructions for many other topics:

- How to transfer data between DB2 and a host program—written in Java™, C, or COBOL, for example
- How to prepare to compile a program that embeds SQL statements
- How to process data from two systems simultaneously, for example, DB2 and IMS or DB2 and CICS
- How to write distributed applications across operating systems
- How to write applications that use Open Database Connectivity (ODBC) to access DB2 servers
- How to write applications that use JDBC and SQLJ with the Java programming language to access DB2 servers
- How to write applications to store XML data on DB2 servers and retrieve XML data from DB2 servers.

The material needed for writing a host program containing SQL is in *DB2 Application Programming and SQL Guide*.

The material needed for writing applications that use JDBC and SQLJ to access DB2 servers is in *DB2 Application Programming Guide and Reference for Java*. The material needed for writing applications that use DB2 CLI or ODBC to access DB2 servers is in *DB2 ODBC Guide and Reference*. The material needed for working with XML data in DB2 is in *DB2 pureXML Guide*. For handling errors, see *DB2 Messages and DB2 Codes*.

## System and database administration

*Administration* covers almost everything else. *DB2 Administration Guide* divides some of those tasks among the following sections:

- Designing a database: Discusses the decisions that must be made when designing a database and tells how to implement the design by creating and altering DB2 objects, loading data, and adjusting to changes.
- Operation and recovery: Describes the steps in normal day-to-day operation and discusses the steps one should take to prepare for recovery in the event of some failure.

*DB2 Managing Performance* explains how to monitor the performance of the DB2 system and its parts. It also lists things that can be done to make some parts run faster.

*DB2 Managing Security* describes ways of controlling access to the DB2 system and to data within DB2, ways to audit aspects of DB2 usage, and ways to answer other security and auditing concerns.

If you will be using the RACF access control module for DB2 authorization checking, you will need *DB2 RACF Access Control Module Guide*.

If you are involved with DB2 only to design the database, or plan operational procedures, you need *DB2 Administration Guide*. If you also want to carry out your own plans by creating DB2 objects, granting privileges, running utility jobs, and so on, you also need:

- *DB2 SQL Reference*, which describes the SQL statements you use to create, alter, and drop objects and grant and revoke privileges
- *DB2 Utility Guide and Reference*, which explains how to run utilities
- *DB2 Command Reference*, which explains how to run commands

If you will be using data sharing, you need *DB2 Data Sharing: Planning and Administration*, which describes how to plan for and implement data sharing.

Additional information about system and database administration can be found in *DB2 Messages* and *DB2 Codes*, which list messages and codes issued by DB2, with explanations and suggested responses.

## **Diagnosis**

Diagnostics detect and describe errors in the DB2 program. They might also recommend or apply a remedy. The documentation for this task is in *DB2 Diagnosis Guide and Reference*, *DB2 Messages*, and *DB2 Codes*.



---

## Notices

This information was developed for products and services offered in the U.S.A.

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  
Armonk, NY 10504-1785  
U.S.A.

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.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM 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 Corporation  
J46A/G4  
555 Bailey Avenue  
San Jose, CA 95141-1003  
U.S.A.

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 the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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)<sup>®</sup> 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 <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.





---

## Glossary

**abend** See abnormal end of task.

**abend reason code**

A 4-byte hexadecimal code that uniquely identifies a problem with DB2.

**abnormal end of task (abend)**

Termination of a task, job, or subsystem because of an error condition that recovery facilities cannot resolve during execution.

**access method services**

The facility that is used to define, alter, delete, print, and reproduce VSAM key-sequenced data sets.

**access path**

The path that is used to locate data that is specified in SQL statements. An access path can be indexed or sequential.

**access path stability**

A characteristic of an access path that defines reliability for dynamic or static queries. Access paths are not regenerated unless there is a schema change or manual intervention.

**active log**

The portion of the DB2 log to which log records are written as they are generated. The active log always contains the most recent log records. See also archive log.

**address space**

A range of virtual storage pages that is identified by a number (ASID) and a collection of segment and page tables that map the virtual pages to real pages of the computer's memory.

**address space connection**

The result of connecting an allied address space to DB2. See also allied address space and task control block.

**address space identifier (ASID)**

A unique system-assigned identifier for an address space.

**AFTER trigger**

A trigger that is specified to be activated after a defined trigger event (an insert, update, or delete operation on the table that is specified in a trigger definition).

Contrast with BEFORE trigger and INSTEAD OF trigger.

**agent** In DB2, the structure that associates all processes that are involved in a DB2 unit of work. See also allied agent and system agent.

**aggregate function**

An operation that derives its result by using values from one or more rows. Contrast with scalar function.

**alias**

An alternative name that can be used in SQL statements to refer to a table or view in the same or a remote DB2 subsystem. An alias can be qualified with a schema qualifier and can thereby be referenced by other users. Contrast with synonym.

**allied address space**

An area of storage that is external to DB2 and that is connected to DB2. An allied address space can request DB2 services. See also address space.

**allied agent**

An agent that represents work requests that originate in allied address spaces. See also system agent.

**allied thread**

A thread that originates at the local DB2 subsystem and that can access data at a remote DB2 subsystem.

**allocated cursor**

A cursor that is defined for a stored procedure result set by using the SQL ALLOCATE CURSOR statement.

**ambiguous cursor**

A database cursor for which DB2 cannot determine whether it is used for update or read-only purposes.

**APAR** See authorized program analysis report.

**APF** See authorized program facility.

**API** See application programming interface.

**APPL** A VTAM network definition statement that is used to define DB2 to VTAM as an application program that uses SNA LU 6.2 protocols.

**application**

A program or set of programs that performs a task; for example, a payroll application.

**application period**

A pair of columns with application-maintained values that indicates the period of time when a row is valid. See also application-period temporal table.

**application-period temporal table**

A table that includes an application period. See also application period and bitemporal table.

**application plan**

The control structure that is produced during the bind process. DB2 uses the application plan to process SQL statements that it encounters during statement execution.

**application process**

The unit to which resources and locks are allocated. An application process involves the execution of one or more programs.

**application programming interface (API)**

A functional interface that is supplied by the operating system or by a separately ordered licensed program that allows an application program that is written in a high-level language to use specific data or functions of the operating system or licensed program.

**application requester**

The component on a remote system that generates DRDA requests for data on behalf of an application.

**application server**

The target of a request from a remote application. In the DB2 environment, the application server function is provided by the distributed data facility and is used to access DB2 data from remote applications.

**archive log**

The portion of the DB2 log that contains log records that have been copied from the active log. See also active log.

**ASCII** An encoding scheme that is used to represent strings in many environments, typically on personal computers and workstations. Contrast with EBCDIC and Unicode.

**ASID** See address space identifier.

**attachment facility**

An interface between DB2 and TSO, IMS, CICS, or batch address spaces. An attachment facility allows application programs to access DB2.

**attribute**

A characteristic of an entity. For example, in database design, the phone number of an employee is an attribute of that employee.

**authorization ID**

A string that can be verified for connection to DB2 and to which a set of privileges is allowed. An authorization ID can represent an individual or an organizational group.

**authorized program analysis report (APAR)**

A report of a problem that is caused by a suspected defect in a current release of an IBM supplied program.

**authorized program facility (APF)**

A facility that allows an installation to identify system or user programs that can use sensitive system functions.

**automatic bind**

(More correctly *automatic rebind*.) A process by which SQL statements are bound automatically (without a user issuing a BIND command) when an application process begins execution and the bound application plan or package it requires is not valid.

**automatic query rewrite**

A process that examines an SQL statement that refers to one or more base tables or materialized query tables, and, if appropriate, rewrites the query so that it performs better.

**auxiliary index**

An index on an auxiliary table in which each index entry refers to a LOB or XML document.

**auxiliary table**

A table that contains columns outside the actual table in which they are defined. Auxiliary tables can contain either LOB or XML data.

**backout**

The process of undoing uncommitted changes that an application process made.

A backout is often performed in the event of a failure on the part of an application process, or as a result of a deadlock situation.

#### **backward log recovery**

The final phase of restart processing during which DB2 scans the log in a backward direction to apply UNDO log records for all aborted changes.

#### **base table**

A table that is created by the SQL CREATE TABLE statement and that holds persistent data. Contrast with clone table, materialized query table, result table, temporary table, and transition table.

#### **base table space**

A table space that contains base tables.

#### **basic row format**

A row format in which values for columns are stored in the row in the order in which the columns are defined by the CREATE TABLE statement. Contrast with reordered row format.

#### **basic sequential access method (BSAM)**

An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential-access or a direct-access device.

#### **BEFORE trigger**

A trigger that is specified to be activated before a defined trigger event (an insert, an update, or a delete operation on the table that is specified in a trigger definition). Contrast with AFTER trigger and INSTEAD OF trigger.

#### **begin column**

In a system period or an application period, the column that indicates the beginning of the period. See also period.

#### **binary large object (BLOB)**

A binary string data type that contains a sequence of bytes that can range in size from 0 bytes to 2 GB, less 1 byte. This string does not have an associated code page and character set. BLOBs can contain, for example, image, audio, or video data. In general, BLOB values are used whenever a binary string might exceed the limits of the VARBINARY type.

#### **binary string**

A sequence of bytes that is not associated with a CCSID. Binary string data type can be further classified as BINARY, VARBINARY, or BLOB.

#### **binary XML format**

A representation of XML data that uses binary values, an approach that facilitates more efficient storage and exchange.

#### **bind**

A process by which a usable control structure with SQL statements is generated; the structure is often called an access plan, an application plan, or a package. During this bind process, access paths to the data are selected, and some authorization checking is performed. See also automatic bind.

#### **bit data**

- Data with character type CHAR or VARCHAR that is defined with the FOR BIT DATA clause. Note that using BINARY or VARBINARY rather than FOR BIT DATA is highly recommended.
- Data with character type CHAR or VARCHAR that is defined with the FOR BIT DATA clause.
- A form of character data. Binary data is generally more highly recommended than character-for-bit data.

#### **bitemporal table**

A table that is both a system-period temporal table and an application-period temporal table. See also application-period temporal table and system-period temporal table.

**BLOB** See binary large object.

#### **block fetch**

A capability in which DB2 can retrieve, or fetch, a large set of rows together. Using block fetch can significantly reduce the number of messages that are being sent across the network. Block fetch applies only to non-rowset cursors that do not update data.

#### **bootstrap data set (BSDS)**

A VSAM data set that contains name and status information for DB2 and RBA range specifications for all active and archive log data sets. The BSDS also contains passwords for the DB2 directory

and catalog, and lists of conditional restart and checkpoint records.

**BSAM**

See basic sequential access method.

**BSDS** See bootstrap data set.

**buffer pool**

An area of memory into which data pages are read, modified, and held during processing.

**built-in data type**

A data type that IBM supplies. Among the built-in data types for DB2 for z/OS are string, numeric, XML, ROWID, and datetime. Contrast with distinct type.

**built-in function**

A function that is generated by DB2 and that is in the SYSIBM schema. Contrast with user-defined function. See also function, cast function, external function, sourced function, and SQL function.

**business dimension**

A category of data, such as products or time periods, that an organization might want to analyze.

**cache structure**

A coupling facility structure that stores data that can be available to all members of a Sysplex. A DB2 data sharing group uses cache structures as group buffer pools.

**CAF** See call attachment facility.

**call attachment facility (CAF)**

A DB2 attachment facility for application programs that run in TSO or z/OS batch. The CAF is an alternative to the DSN command processor and provides greater control over the execution environment. Contrast with Resource Recovery Services attachment facility.

**call-level interface (CLI)**

A callable application programming interface (API) for database access, which is an alternative to using embedded SQL.

**cascade delete**

A process by which DB2 enforces referential constraints by deleting all descendent rows of a deleted parent row.

**CASE expression**

An expression that is selected based on the evaluation of one or more conditions.

**cast function**

A function that is used to convert instances of a (source) data type into instances of a different (target) data type.

**castout**

The DB2 process of writing changed pages from a group buffer pool to disk.

**castout owner**

The DB2 member that is responsible for casting out a particular page set or partition.

**catalog**

In DB2, a collection of tables that contains descriptions of objects such as tables, views, and indexes.

**catalog table**

Any table in the DB2 catalog.

**CCSID**

See coded character set identifier.

**CDB**

See communications database.

**CDRA**

See Character Data Representation Architecture.

**CEC**

See central processor complex.

**central electronic complex (CEC)**

See central processor complex.

**central processor complex (CPC)**

A physical collection of hardware that consists of main storage, one or more central processors, timers, and channels.

**central processor (CP)**

The part of the computer that contains the sequencing and processing facilities for instruction execution, initial program load, and other machine operations.

**CFRM** See coupling facility resource management.

**CFRM policy**

The allocation rules for a coupling facility structure that are declared by a z/OS administrator.

**character conversion**

The process of changing characters from one encoding scheme to another.

**Character Data Representation Architecture (CDRA)**

An architecture that is used to achieve

consistent representation, processing, and interchange of string data.

**character large object (CLOB)**

A character string data type that contains a sequence of bytes that represent characters (single-byte, multibyte, or both) that can range in size from 0 bytes to 2 GB, less 1 byte. In general, CLOB values are used whenever a character string might exceed the limits of the VARCHAR type.

**character set**

A defined set of characters.

**character string**

A sequence of bytes that represent bit data, single-byte characters, or a mixture of single-byte and multibyte characters. Character data can be further classified as CHARACTER, VARCHAR, or CLOB.

**check constraint**

A user-defined constraint that specifies the values that specific columns of a base table can contain.

**check integrity**

The condition that exists when each row in a table conforms to the check constraints that are defined on that table.

**check pending**

A state of a table space or partition that prevents its use by some utilities and by some SQL statements because of rows that violate referential constraints, check constraints, or both.

**checkpoint**

A point at which DB2 records status information on the DB2 log; the recovery process uses this information if DB2 abnormally terminates.

**child lock**

For explicit hierarchical locking, a lock that is held on either a table, page, row, or a large object (LOB). Each child lock has a parent lock. See also parent lock.

**CI** See control interval.

**CICS** Represents (in this information): CICS Transaction Server for z/OS: Customer Information Control System Transaction Server for z/OS.

**CICS attachment facility**

A facility that provides a multithread

connection to DB2 to allow applications that run in the CICS environment to execute DB2 statements.

**claim** A notification to DB2 that an object is being accessed. Claims prevent drains from occurring until the claim is released, which usually occurs at a commit point. Contrast with drain.

**claim class**

A specific type of object access that can be one of the following isolation levels:

- Cursor stability (CS)
- Repeatable read (RR)
- Write

**class of service**

A VTAM term for a list of routes through a network, arranged in an order of preference for their use.

**clause** In SQL, a distinct part of a statement, such as a SELECT clause or a WHERE clause.

**CLI** See call-level interface.

**client** See requester.

**CLOB** See character large object.

**clone object**

An object that is associated with a clone table, including the clone table itself and check constraints, indexes, and BEFORE triggers on the clone table.

**clone table**

A table that is structurally identical to a base table. The base and clone table each have separate underlying VSAM data sets, which are identified by their data set instance numbers. Contrast with base table.

**closed application**

An application that requires exclusive use of certain statements on certain DB2 objects, so that the objects are managed solely through the external interface of that application.

**clustering index**

An index that determines how rows are physically ordered (*clustered*) in a table space. If a clustering index on a partitioned table is not a partitioning index, the rows are ordered in cluster sequence within each data partition instead of spanning partitions.



| **CM** See conversion mode.

| **CM\*** See conversion mode\*.

**C++ member**

A data object or function in a structure, union, or class.

**C++ member function**

An operator or function that is declared as a member of a class. A member function has access to the private and protected data members and to the member functions of objects in its class. Member functions are also called methods.

**C++ object**

A region of storage. An object is created when a variable is defined or a new function is invoked.

An instance of a class.

**coded character set**

A set of unambiguous rules that establish a character set and the one-to-one relationships between the characters of the set and their coded representations.

**coded character set identifier (CCSID)**

A 16-bit number that uniquely identifies a coded representation of graphic characters. It designates an encoding scheme identifier and one or more pairs that consist of a character set identifier and an associated code page identifier.

**code page**

A set of assignments of characters to code points. Within a code page, each code point has only one specific meaning. In EBCDIC, for example, the character *A* is assigned code point X'C1', and character *B* is assigned code point X'C2'.

**code point**

In CDRA, a unique bit pattern that represents a character in a code page.

**code unit**

The fundamental binary width in a computer architecture that is used for representing character data, such as 7 bits, 8 bits, 16 bits, or 32 bits. Depending on the character encoding form that is used, each code point in a coded character set can be represented by one or more code units.

**coexistence**

During migration, the period of time in which two releases exist in the same data sharing group.

**cold start**

A process by which DB2 restarts without processing any log records. Contrast with warm start.

**collection**

A group of packages that have the same qualifier.

**column**

The vertical component of a table. A column has a name and a particular data type (for example, character, decimal, or integer).

**column function**

See aggregate function.

**"come from" checking**

An LU 6.2 security option that defines a list of authorization IDs that are allowed to connect to DB2 from a partner LU.

**command**

A DB2 operator command or a DSN subcommand. A command is distinct from an SQL statement.

**command prefix**

A 1- to 8-character command identifier. The command prefix distinguishes the command as belonging to an application or subsystem rather than to z/OS.

**command recognition character (CRC)**

A character that permits a z/OS console operator or an IMS subsystem user to route DB2 commands to specific DB2 subsystems.

**command scope**

The scope of command operation in a data sharing group.

**commit**

The operation that ends a unit of work by releasing locks so that the database changes that are made by that unit of work can be perceived by other processes. Contrast with rollback.

**commit point**

A point in time when data is considered consistent.

**common service area (CSA)**

| In z/OS, a part of the common area that

| contains data areas that are addressable  
| by all address spaces. Most DB2 use is in  
| the extended CSA, which is above the  
| 16-MB line.

**communications database (CDB)**

A set of tables in the DB2 catalog that are used to establish conversations with remote database management systems.

**comparison operator**

A token (such as =, >, or <) that is used to specify a relationship between two values.

**compatibility mode**

See conversion mode.

**compatibility mode\* (CM\*)**

See conversion mode\*.

**composite key**

| An ordered set of key columns or  
| expressions of the same table.

**compression dictionary**

The dictionary that controls the process of compression and decompression. This dictionary is created from the data in the table space or table space partition.

**concurrency**

The shared use of resources by more than one application process at the same time.

**conditional restart**

A DB2 restart that is directed by a user-defined conditional restart control record (CRCR).

**connection**

In SNA, the existence of a communication path between two partner LUs that allows information to be exchanged (for example, two DB2 subsystems that are connected and communicating by way of a conversation).

**connection context**

In SQLJ, a Java object that represents a connection to a data source.

**connection declaration clause**

In SQLJ, a statement that declares a connection to a data source.

**connection handle**

The data object containing information that is associated with a connection that DB2 ODBC manages. This includes general status information, transaction status, and diagnostic information.

**connection ID**

An identifier that is supplied by the attachment facility and that is associated with a specific address space connection.

**consistency token**

A timestamp that is used to generate the version identifier for an application. See also version.

**constant**

A language element that specifies an unchanging value. Constants are classified as string constants or numeric constants. Contrast with variable.

**constraint**

A rule that limits the values that can be inserted, deleted, or updated in a table. See referential constraint, check constraint, and unique constraint.

**context**

An application's logical connection to the data source and associated DB2 ODBC connection information that allows the application to direct its operations to a data source. A DB2 ODBC context represents a DB2 thread.

**contracting conversion**

A process that occurs when the length of a converted string is smaller than that of the source string. For example, this process occurs when an EBCDIC mixed-data string that contains DBCS characters is converted to ASCII mixed data; the converted string is shorter because the shift codes are removed.

**control interval (CI)**

- A unit of information that VSAM transfers between virtual and auxiliary storage.
- In a key-sequenced data set or file, the set of records that an entry in the sequence-set index record points to.

**conversation**

Communication, which is based on LU 6.2 or Advanced Program-to-Program Communication (APPC), between an application and a remote transaction program over an SNA logical unit-to-logical unit (LU-LU) session that allows communication while processing a transaction.

<b>conversion mode* (CM*)</b>	A stage of the version-to-version migration process that applies to a DB2 subsystem or data sharing group that was in enabling-new-function mode (ENFM), enabling-new-function mode* (ENFM*), or new-function mode (NFM) at one time. Fallback to a prior version is not supported. When in conversion mode*, a DB2 data sharing group cannot coexist with members that are still at the prior version level. Contrast with conversion mode, enabling-new-function mode, enabling-new-function mode*, and new-function mode.	<b>copy version</b>	A point-in-time FlashCopy copy that is managed by HSM. Each copy pool has a version parameter that specifies the number of copy versions to be maintained on disk.
Previously known as compatibility mode* (CM*).		<b>correlated columns</b>	A relationship between the value of one column and the value of another column.
<b>conversion mode (CM)</b>	The first stage of the version-to-version migration process. In a DB2 data sharing group, members in conversion mode can coexist with members that are still at the prior version level. Fallback to the prior version is also supported. When in conversion mode, the DB2 subsystem cannot use most new functions of the new version. Contrast with conversion mode*, enabling-new-function mode, enabling-new-function mode*, and new-function mode.	<b>correlated subquery</b>	A subquery (part of a WHERE or HAVING clause) that is applied to a row or group of rows of a table or view that is named in an outer subselect statement.
Previously known as compatibility mode (CM).		<b>correlation ID</b>	An identifier that is associated with a specific thread. In TSO, it is either an authorization ID or the job name.
<b>coordinator</b>	The system component that coordinates the commit or rollback of a unit of work that includes work that is done on one or more other systems.	<b>correlation name</b>	An identifier that is specified and used within a single SQL statement as the exposed name for objects such as a table, view, table function reference, nested table expression, or result of a data change statement. Correlation names are useful in an SQL statement to allow two distinct references to the same base table and to allow an alternative name to be used to represent an object.
<b>coprocessor</b>	See SQL statement coprocessor.	<b>cost category</b>	A category into which DB2 places cost estimates for SQL statements at the time the statement is bound. The cost category is externalized in the COST_CATEGORY column of the DSN_STATEMENT_TABLE when a statement is explained.
<b>copy pool</b>	A collection of names of storage groups that are processed collectively for fast replication operations.	<b>coupling facility</b>	A special PR/SM logical partition (LPAR) that runs the coupling facility control program and provides high-speed caching, list processing, and locking functions in a Parallel Sysplex.
<b>copy target</b>	A named set of SMS storage groups that are to be used as containers for copy pool volume copies. A copy target is an SMS construct that lets you define which storage groups are to be used as containers for volumes that are copied by using FlashCopy functions.	<b>coupling facility resource management (CFRM)</b>	A component of z/OS that provides the services to manage coupling facility resources in a Parallel Sysplex. This management includes the enforcement of CFRM policies to ensure that the coupling facility and structure requirements are satisfied.
		<b>CP</b>	See central processor.



**CPC** See central processor complex.

**CRC** See command recognition character.

**created temporary table**

A persistent table that holds temporary data and is defined with the SQL statement CREATE GLOBAL TEMPORARY TABLE. Information about created temporary tables is stored in the DB2 catalog and can be shared across application processes. Contrast with declared temporary table. See also temporary table.

**cross-system coupling facility (XCF)**

A component of z/OS that provides functions to support cooperation between authorized programs that run within a Sysplex.

**cross-system extended services (XES)**

A set of z/OS services that allow multiple instances of an application or subsystem, running on different systems in a Sysplex environment, to implement high-performance, high-availability data sharing by using a coupling facility.

**CS** See cursor stability.

**CSA** See common service area.

**CT** See cursor table.

**current data**

Data within a host structure that is current with (identical to) the data within the base table.

**current status rebuild**

The second phase of restart processing during which the status of the subsystem is reconstructed from information on the log.

**cursor** A control structure that an application program uses to point to a single row or multiple rows within some ordered set of rows of a result table. A cursor can be used to retrieve, update, or delete rows from a result table.

**cursor sensitivity**

The degree to which database updates are visible to the subsequent FETCH statements in a cursor.

**cursor stability (CS)**

The isolation level that provides maximum concurrency without the ability to read uncommitted data. With cursor

stability, a unit of work holds locks only on its uncommitted changes and on the current row of each of its cursors. See also read stability, repeatable read, and uncommitted read.

**cursor table (CT)**

The internal representation of a cursor.

**cycle** A set of tables that can be ordered so that each table is a descendent of the one before it, and the first table is a descendent of the last table. A self-referencing table is a cycle with a single member. See also referential cycle.

**database**

A collection of tables, or a collection of table spaces and index spaces.

**database access thread (DBAT)**

A thread that accesses data at the local subsystem on behalf of a remote subsystem.

**database administrator (DBA)**

An individual who is responsible for designing, developing, operating, safeguarding, maintaining, and using a database.

**database alias**

The name of the target server if it is different from the location name. The database alias is used to provide the name of the database server as it is known to the network.

**database descriptor (DBD)**

An internal representation of a DB2 database definition, which reflects the data definition that is in the DB2 catalog. The objects that are defined in a database descriptor are table spaces, tables, indexes, index spaces, relationships, check constraints, and triggers. A DBD also contains information about accessing tables in the database.

**database exception status**

In a data sharing environment, an indication that something is wrong with a database.

**database identifier (DBID)**

An internal identifier of the database.

**database management system (DBMS)**

A software system that controls the

creation, organization, and modification of a database and the access to the data that is stored within it.

**database request module (DBRM)**

A data set member that is created by the DB2 precompiler and that contains information about SQL statements. DBRMs are used in the bind process.

**database server**

The target of a request from a local application or a remote intermediate database server.

**data currency**

The state in which the data that is retrieved into a host variable in a program is a copy of the data in the base table.

| **data-dependent pagination**

| The process used when applications need  
| to access part of a DB2 result set that is  
| based on a logical key value.

**data dictionary**

A repository of information about an organization's application programs, databases, logical data models, users, and authorizations.

**data partition**

A VSAM data set that is contained within a partitioned table space.

**data-partitioned secondary index (DPSI)**

A secondary index that is partitioned according to the underlying data. Contrast with nonpartitioned secondary index.

| **data set instance number**

| A number that indicates the data set that  
| contains the data for an object.

**data sharing**

A function of DB2 for z/OS that enables applications on different DB2 subsystems to read from and write to the same data concurrently.

**data sharing group**

A collection of one or more DB2 subsystems that directly access and change the same data while maintaining data integrity.

**data sharing member**

A DB2 subsystem that is assigned by XCF services to a data sharing group.

**data source**

A local or remote relational or non-relational data manager that is capable of supporting data access via an ODBC driver that supports the ODBC APIs. In the case of DB2 for z/OS, the data sources are always relational database managers.

**data type**

An attribute of columns, constants, variables, parameters, special registers, and the results of functions and expressions.

**data warehouse**

A system that provides critical business information to an organization. The data warehouse system cleanses the data for accuracy and currency, and then presents the data to decision makers so that they can interpret and use it effectively and efficiently.

**DBA** See database administrator.

**DBAT** See database access thread.

**DB2 catalog**

A collection of tables that are maintained by DB2 and contain descriptions of DB2 objects, such as tables, views, and indexes.

**DBCLOB**

See double-byte character large object.

**DB2 command**

An instruction to the DB2 subsystem that a user enters to start or stop DB2, to display information on current users, to start or stop databases, to display information on the status of databases, and so on.

**DBCS** See double-byte character set.

**DBD** See database descriptor.

**DB2I** See DB2 Interactive.

**DBID** See database identifier.

**DB2 Interactive (DB2I)**

An interactive service within DB2 that facilitates the execution of SQL statements, DB2 (operator) commands, and programmer commands, and the invocation of utilities.

**DBMS**

See database management system.

**DBRM**

See database request module.

**DB2 thread**

The database manager structure that describes an application's connection, traces its progress, processes resource functions, and delimits its accessibility to the database manager resources and services. Most DB2 for z/OS functions execute under a thread structure.

**DCLGEN**

See declarations generator.

**DDF** See distributed data facility.

**deadlock**

Unresolved contention for the use of a resource, such as a table or an index.

**declarations generator (DCLGEN)**

A subcomponent of DB2 that generates SQL table declarations and COBOL, C, or PL/I data structure declarations that conform to the table. The declarations are generated from DB2 system catalog information.

**declared temporary table**

A non-persistent table that holds temporary data and is defined with the SQL statement DECLARE GLOBAL TEMPORARY TABLE. Information about declared temporary tables is not stored in the DB2 catalog and can be used only by the application process that issued the DECLARE statement. Contrast with created temporary table. See also temporary table.

**default value**

A predetermined value, attribute, or option that is assumed when no other value is specified. A default value can be defined for column data in DB2 tables by specifying the DEFAULT keyword in an SQL statement that changes data (such as INSERT, UPDATE, and MERGE).

**deferred embedded SQL**

SQL statements that are neither fully static nor fully dynamic. These statements are embedded within an application and are prepared during the execution of the application.

**deferred write**

The process of asynchronously writing changed data pages to disk.

**degree of parallelism**

The number of concurrently executed operations that are initiated to process a query.

**delete hole**

The location on which a cursor is positioned when a row in a result table is refetched and the row no longer exists on the base table. See also update hole.

**delete rule**

The rule that tells DB2 what to do to a dependent row when a parent row is deleted. Delete rules include CASCADE, RESTRICT, SET NULL, or NO ACTION.

**delete trigger**

A trigger that is defined with the triggering delete SQL operation.

**delimited identifier**

A sequence of one or more characters enclosed by escape characters, such as quotation marks ("").

**delimiter token**

A string constant, a delimited identifier, an operator symbol, or any of the special characters that are shown in DB2 syntax diagrams.

**denormalization**

The intentional duplication of columns in multiple tables to increase data redundancy. Denormalization is sometimes necessary to minimize performance problems. Contrast with normalization.

**dependent**

An object (row, table, or table space) that has at least one parent. The object is also said to be a dependent (row, table, or table space) of its parent. See also parent row, parent table, and parent table space.

**dependent row**

A row that contains a foreign key that matches the value of a primary key in the parent row.

**dependent table**

A table that is a dependent in at least one referential constraint.

**descendent**

An object that is a dependent of an object or is the dependent of a descendent of an object.

**descendent row**

A row that is dependent on another row, or a row that is a descendent of a dependent row.

**descendent table**

A table that is a dependent of another table, or a table that is a descendent of a dependent table.

**deterministic function**

A user-defined function whose result is dependent on the values of the input arguments. That is, successive invocations with the same input values produce the same answer. Sometimes referred to as a *not-variant* function. Contrast with nondeterministic function (sometimes called a *variant function*).

**dimension**

A data category such as time, products, or markets. The elements of a dimension are referred to as members. See also dimension table.

**dimension table**

The representation of a dimension in a star schema. Each row in a dimension table represents all of the attributes for a particular member of the dimension. See also dimension, star schema, and star join.

**directory**

The DB2 system database that contains internal objects such as database descriptors and skeleton cursor tables.

**disk** A direct-access storage device that records data magnetically.

**distinct type**

A user-defined data type that shares a common representation with a built-in data type.

**distributed data**

Data that resides on a DBMS other than the local system.

**distributed data facility (DDF)**

A set of DB2 components through which DB2 communicates with another relational database management system.

**Distributed Relational Database Architecture (DRDA)**

A connection protocol for distributed relational database processing that is used by IBM relational database products. DRDA includes protocols for

communication between an application and a remote relational database management system, and for communication between relational database management systems. See also DRDA access.

**DNS** See domain name server.

**DOCID**

See document ID.

**document ID**

A value that uniquely identifies a row that contains an XML column. This value is stored with the row and never changes.

**domain**

The set of valid values for an attribute.

**domain name**

The name by which TCP/IP applications refer to a TCP/IP host within a TCP/IP network.

**domain name server (DNS)**

A special TCP/IP network server that manages a distributed directory that is used to map TCP/IP host names to IP addresses.

**double-byte character large object (DBCLOB)**

A graphic string data type in which a sequence of bytes represent double-byte characters that range in size from 0 bytes to 2 GB, less 1 byte. In general, DBCLOB values are used whenever a double-byte character string might exceed the limits of the VARCHAR type.

**double-byte character set (DBCS)**

A set of characters, which are used by national languages such as Japanese and Chinese, that have more symbols than can be represented by a single byte. Each character is 2 bytes in length. Contrast with single-byte character set and multibyte character set.

**double-precision floating point number**

A 64-bit approximate representation of a real number.

**DPSI** See data-partitioned secondary index.

**drain** The act of acquiring a locked resource by quiescing access to that object. Contrast with claim.

**drain lock**

A lock on a claim class that prevents a claim from occurring.

## **DRDA**

See Distributed Relational Database Architecture.

## **DRDA access**

An open method of accessing distributed data that you can use to connect to another database server to execute packages that were previously bound at the server location.

## **DSN**

- The default DB2 subsystem name.
- The name of the TSO command processor of DB2.
- The first three characters of DB2 module and macro names.

## **dynamic cursor**

A named control structure that an application program uses to change the size of the result table and the order of its rows after the cursor is opened. Contrast with static cursor.

## **dynamic dump**

A dump that is issued during the execution of a program, usually under the control of that program.

## **dynamic SQL**

SQL statements that are prepared and executed at run time. In dynamic SQL, the SQL statement is contained as a character string in a host variable or as a constant, and it is not precompiled.

## **EA-enabled table space**

A table space or index space that is enabled for extended addressability and that contains individual partitions (or pieces, for LOB table spaces) that are greater than 4 GB.

## **EB**

See exabyte.

## **EBCDIC**

Extended binary coded decimal interchange code. An encoding scheme that is used to represent character data in the z/OS, VM, VSE, and iSeries environments. Contrast with ASCII and Unicode.

## **embedded SQL**

SQL statements that are coded within an application program. See static SQL.

## **enabling-new-function mode\* (ENFM\*)**

A transitional stage of the

version-to-version migration process that applies to a DB2 subsystem or data sharing group that was in new-function mode (NFM) at one time. When in enabling-new-function mode\*, a DB2 subsystem or data sharing group is preparing to use the new functions of the new version but cannot yet use them. A data sharing group that is in enabling-new-function mode\* cannot coexist with members that are still at the prior version level. Fallback to a prior version is not supported. Contrast with conversion mode, conversion mode\*, enabling-new-function mode, and new-function mode.

## **enabling-new-function mode (ENFM)**

A transitional stage of the version-to-version migration process during which the DB2 subsystem or data sharing group is preparing to use the new functions of the new version. When in enabling-new-function mode, a DB2 data sharing group cannot coexist with members that are still at the prior version level. Fallback to a prior version is not supported, and most new functions of the new version are not available for use in enabling-new-function mode. Contrast with conversion mode, conversion mode\*, enabling-new-function mode\*, and new-function mode.

## **enclave**

In Language Environment, an independent collection of routines, one of which is designated as the main routine. An enclave is similar to a program or run unit. See also WLM enclave.

## **encoding scheme**

A set of rules to represent character data (ASCII, EBCDIC, or Unicode).

## **end column**

In a system period or an application period, the column that indicates the end of the period. See also period.

**ENFM** See enabling-new-function mode.

## **ENFM\***

See enabling-new-function mode\*.

## **entity**

A person, object, or concept about which information is stored. In a relational database, entities are represented as tables. A database includes information



	about the entities in an organization or business, and their relationships to each other.	executing application processes from reading or changing data. Contrast with share lock.
	<b>enumerated list</b> A set of DB2 objects that are defined with a LISTDEF utility control statement in which pattern-matching characters (*, %;, _ or ?) are not used.	<b>executable statement</b> An SQL statement that can be embedded in an application program, dynamically prepared and executed, or issued interactively.
	<b>environment</b> A collection of names of logical and physical resources that are used to support the performance of a function.	<b>execution context</b> In SQLJ, a Java object that can be used to control the execution of SQL statements.
	<b>environment handle</b> A handle that identifies the global context for database access. All data that is pertinent to all objects in the environment is associated with this handle.	<b>exit routine</b> A user-written (or IBM-provided default) program that receives control from DB2 to perform specific functions. Exit routines run as extensions of DB2.
	<b>equijoin</b> A join operation in which the join-condition has the form <i>expression = expression</i> . See also join, full outer join, inner join, left outer join, outer join, and right outer join.	<b>expanding conversion</b> A process that occurs when the length of a converted string is greater than that of the source string. For example, this process occurs when an ASCII mixed-data string that contains DBCS characters is converted to an EBCDIC mixed-data string; the converted string is longer because shift codes are added.
	<b>error page range</b> A range of pages that are considered to be physically damaged. DB2 does not allow users to access any pages that fall within this range.	<b>explicit hierarchical locking</b> Locking that is used to make the parent-child relationship between resources known to IRLM. This kind of locking avoids global locking overhead when no inter-DB2 interest exists on a resource.
	<b>escape character</b> The symbol, a double quotation (") for example, that is used to enclose an SQL delimited identifier.	<b>explicit privilege</b> A privilege that has a name and is held as the result of an SQL GRANT statement and revoked as the result of an SQL REVOKE statement. For example, the SELECT privilege.
	<b>exabyte</b> A unit of measure for processor, real and virtual storage capacities, and channel volume that has a value of 1 152 921 504 606 846 976 bytes or 2 <sup>60</sup> .	<b>exposed name</b> A correlation name or a table or view name for which a correlation name is not specified.
	<b>exception</b> An SQL operation that involves the EXCEPT set operator, which combines two result tables. The result of an exception operation consists of all of the rows that are in only one of the result tables.	<b>expression</b> An operand or a collection of operators and operands that yields a single value.
	<b>exception table</b> A table that holds rows that violate referential constraints or check constraints that the CHECK DATA utility finds.	<b>Extended Recovery Facility (XRF)</b> A facility that minimizes the effect of failures in z/OS, VTAM, the host processor, or high-availability applications during sessions between high-availability applications and designated terminals.
	<b>exclusive lock</b> A lock that prevents concurrently	

This facility provides an alternative subsystem to take over sessions from the failing subsystem.

**Extensible Markup Language (XML)**

A standard metalanguage for defining markup languages that is a subset of Standardized General Markup Language (SGML).

**external function**

A function that has its functional logic implemented in a programming language application that resides outside the database, in the file system of the database server. The association of the function with the external code application is specified by the EXTERNAL clause in the CREATE FUNCTION statement. External functions can be classified as external scalar functions and external table functions. Contrast with sourced function, built-in function, and SQL function.

**external procedure**

A procedure that has its procedural logic implemented in an external programming language application. The association of the procedure with the external application is specified by a CREATE PROCEDURE statement with a LANGUAGE clause that has a value other than SQL and an EXTERNAL clause that implicitly or explicitly specifies the name of the external application. Contrast with external SQL procedure and native SQL procedure.

**external routine**

A user-defined function or stored procedure that is based on code that is written in an external programming language.

**external SQL procedure**

An SQL procedure that is processed using a generated C program that is a representation of the procedure. When an external SQL procedure is called, the C program representation of the procedure is executed in a stored procedures address space. Contrast with external procedure and native SQL procedure.

**failed member state**

A state of a member of a data sharing group in which the member's task,

address space, or z/OS system terminates before the state changes from active to quiesced.

**fallback**

The process of returning to a previous release of DB2 after attempting or completing migration to a current release. Fallback is supported only from a subsystem that is in conversion mode.

**false global lock contention**

A contention indication from the coupling facility that occurs when multiple lock names are hashed to the same indicator and when no real contention exists.

**fan set**

A direct physical access path to data, which is provided by an index, hash, or link; a fan set is the means by which DB2 supports the ordering of data.

**federated database**

The combination of a DB2 server (in Linux, UNIX, and Windows environments) and multiple data sources to which the server sends queries. In a federated database system, a client application can use a single SQL statement to join data that is distributed across multiple database management systems and can view the data as if it were local.

**fetch orientation**

The specification of the desired placement of the cursor as part of a FETCH statement. The specification can be before or after the rows of the result table (with BEFORE or AFTER). The specification can also have either a single-row fetch orientation (for example, NEXT, LAST, or ABSOLUTE *n*) or a rowset fetch orientation (for example, NEXT ROWSET, LAST ROWSET, or ROWSET STARTING AT ABSOLUTE *n*).

**field procedure**

A user-written exit routine that is designed to receive a single value and transform (encode or decode) it in any way the user can specify.

**file reference variable**

A host variable that is declared with one of the derived data types (BLOB\_FILE, CLOB\_FILE, DBCLOB\_FILE); file

| reference variables direct the reading of a  
| LOB from a file or the writing of a LOB  
| into a file.

**filter factor**

A number between zero and one that estimates the proportion of rows in a table for which a predicate is true.

**fixed-length string**

| A character, graphic, or binary string  
| whose length is specified and cannot be  
| changed. Contrast with varying-length  
| string.

**FlashCopy**

| A function on the IBM Enterprise Storage  
| Server that can, in conjunction with the  
| BACKUP SYSTEM utility, create a  
| point-in-time copy of data while an  
| application is running.

**foreign key**

A column or set of columns in a dependent table of a constraint relationship. The key must have the same number of columns, with the same descriptions, as the primary key of the parent table. Each foreign key value must either match a parent key value in the related parent table or be null.

**forest** An ordered set of subtrees of XML nodes.

**forward log recovery**

The third phase of restart processing during which DB2 processes the log in a forward direction to apply all REDO log records.

**free space**

The total amount of unused space in a page; that is, the space that is not used to store records or control information is free space.

**full outer join**

The result of a join operation that includes the matched rows of both tables that are being joined and preserves the unmatched rows of both tables. See also join, equijoin, inner join, left outer join, outer join, and right outer join.

| **fullselect**

| A subselect, a fullselect in parentheses, or  
| a number of both that are combined by  
| set operators. Fullselect specifies a result  
| table. If a set operator is not used, the

result of the fullselect is the result of the specified subselect or fullselect.

**fully escaped mapping**

A mapping from an SQL identifier to an XML name when the SQL identifier is a column name.

**function**

A mapping, which is embodied as a program (the function body) that can be invoked by means of zero or more input values (arguments) to a single value (the result). See also aggregate function and scalar function.

Functions can be user-defined, built-in, or generated by DB2. (See also built-in function, cast function, external function, sourced function, SQL function, and user-defined function.)

**function definer**

The authorization ID of the owner of the schema of the function that is specified in the CREATE FUNCTION statement.

**function package**

A package that results from binding the DBRM for a function program.

**function package owner**

The authorization ID of the user who binds the function program's DBRM into a function package.

**function signature**

The logical concatenation of a fully qualified function name with the data types of all of its parameters.

**GB** Gigabyte. A value of (1 073 741 824 bytes).

**GBP** See group buffer pool.

**GBP-dependent**

The status of a page set or page set partition that is dependent on the group buffer pool. Either read/write interest is active among DB2 subsystems for this page set, or the page set has changed pages in the group buffer pool that have not yet been cast out to disk.

**generalized trace facility (GTF)**

A z/OS service program that records significant system events such as I/O interrupts, SVC interrupts, program interrupts, or external interrupts.

| **generated column**

| A column for which the database



	manager assigns the value. An example of	
	a generated column is an identity column,	
	row change timestamp column, or	
	row-begin column. See also generated	<b>group buffer pool recovery pending (GRECP)</b>
	expression column.	The state that exists after the buffer pool
		for a data sharing group is lost. When a
	<b>generated expression column</b>	page set is in this state, changes that are
	A generated column that is defined using	recorded in the log must be applied to the
	an expression. See also generated column.	affected page set before the page set can
		be used.
	<b>generic resource name</b>	
	A name that VTAM uses to represent	<b>group level</b>
	several application programs that provide	The release level of a data sharing group,
	the same function in order to handle	which is established when the first
	session distribution and balancing in a	member migrates to a new release.
	Sysplex environment.	
	<b>getpage</b>	<b>group name</b>
	An operation in which DB2 accesses a	The z/OS XCF identifier for a data
	data page.	sharing group.
	<b>global lock</b>	<b>group restart</b>
	A lock that provides concurrency control	A restart of at least one member of a data
	within and among DB2 subsystems. The	sharing group after the loss of either locks
	scope of the lock is across all DB2	or the shared communications area.
	subsystems of a data sharing group.	
	<b>global lock contention</b>	<b>GTF</b>
	Conflicts on locking requests between	See generalized trace facility.
	different DB2 members of a data sharing	
	group when those members are trying to	<b>handle</b>
	serialize shared resources.	In DB2 ODBC, a variable that refers to a
		data structure and associated resources.
	<b>governor</b>	See also statement handle, connection
	See resource limit facility.	handle, and environment handle.
	<b>graphic string</b>	<b>hash access</b>
	A sequence of DBCS characters. Graphic	Access to a table using the hash value of
	data can be further classified as	a key that is defined by the
	GRAPHIC, VARGRAPHIC, or DBCLOB.	<i>organization-clause</i> of a CREATE TABLE
		statement or ALTER TABLE statement.
	<b>GRECP</b>	<b>hash overflow index</b>
	See group buffer pool recovery pending.	A DB2 index used to track data rows that
	<b>gross lock</b>	do not fit into the fixed hash space, and
	The <i>shared</i> , <i>update</i> , or <i>exclusive</i> mode locks	therefore, reside in the hash overflow
	on a table, partition, or table space.	space. DB2 accesses the hash overflow
		index to fetch rows from the hash
	<b>group buffer pool duplexing</b>	overflow area.
	The ability to write data to two instances	
	of a group buffer pool structure: a	<b>help panel</b>
	primary group buffer pool and a	A screen of information that presents
	secondary group buffer pool. z/OS	tutorial text to assist a user at the
	publications refer to these instances as the	workstation or terminal.
	"old" (for primary) and "new" (for	
	secondary) structures.	<b>heuristic damage</b>
	<b>group buffer pool (GBP)</b>	The inconsistency in data between one or
	A coupling facility cache structure that is	more participants that results when a
	used by a data sharing group to cache	heuristic decision to resolve an indoubt
		LUW at one or more participants differs
		from the decision that is recorded at the
		coordinator.

**heuristic decision**

A decision that forces indoubt resolution at a participant by means other than automatic resynchronization between coordinator and participant.

**histogram statistics**

A way of summarizing data distribution. This technique divides up the range of possible values in a data set into intervals, such that each interval contains approximately the same percentage of the values. A set of statistics are collected for each interval.

**historical row**

A row in a history table. See also history table.

**history table**

A table that is used by the database manager to store historical versions of the rows from the associated system-period temporal table. See also historical row and system-period temporal table.

**hole** A row of the result table that cannot be accessed because of a delete or an update that has been performed on the row. See also delete hole and update hole.

**home address space**

The area of storage that z/OS currently recognizes as *dispatched*.

**host** The set of programs and resources that are available on a given TCP/IP instance.

**host expression**

A Java variable or expression that is referenced by SQL clauses in an SQLJ application program.

**host identifier**

A name that is declared in the host program.

**host language**

A programming language in which you can embed SQL statements.

**host program**

An application program that is written in a host language and that contains embedded SQL statements.

**host structure**

In an application program, a structure that is referenced by embedded SQL statements.

**host variable**

In an application program written in a host language, an application variable that is referenced by embedded SQL statements.

**host variable array**

An array of elements, each of which corresponds to a value for a column. The dimension of the array determines the maximum number of rows for which the array can be used.

**IBM System z9 Integrated Processor (zIIP)**

A specialized processor that can be used for some DB2 functions.

**IDCAMS**

An IBM program that is used to process access method services commands. It can be invoked as a job or jobstep, from a TSO terminal, or from within a user's application program.

**IDCAMS LISTCAT**

A facility for obtaining information that is contained in the access method services catalog.

**identity column**

A generated column that is defined with the AS IDENTITY clause. An identity column provides a way for the database manager to automatically generate a numeric value for each row that is inserted into a table. A table can have no more than one identity column.

**IFCID** See instrumentation facility component identifier.

**IFI** See instrumentation facility interface.

**IFI call**

An invocation of the instrumentation facility interface (IFI) by means of one of its defined functions.

**image copy**

An exact reproduction of all or part of a table space. DB2 provides utility programs to make full image copies (to copy the entire table space) or incremental image copies (to copy only those pages that have been modified since the last image copy).

**IMS attachment facility**

A DB2 subcomponent that uses z/OS subsystem interface (SSI) protocols and cross-memory linkage to process requests

from IMS to DB2 and to coordinate resource commitment.

**in-abort**

A status of a unit of recovery. If DB2 fails after a unit of recovery begins to be rolled back, but before the process is completed, DB2 continues to back out the changes during restart.

**in-commit**

A status of a unit of recovery. If DB2 fails after beginning its phase 2 commit processing, it "knows," when restarted, that changes made to data are consistent. Such units of recovery are termed *in-commit*.

**independent**

An object (row, table, or table space) that is neither a parent nor a dependent of another object.

**index** A set of pointers that are logically ordered by the values of a key. Indexes can provide faster access to data and can enforce uniqueness on the rows in a table.

**index-controlled partitioning**

A type of partitioning in which partition boundaries for a partitioned table are controlled by values that are specified on the CREATE INDEX statement. Partition limits are saved in the LIMITKEY column of the SYSIBM.SYSINDEXPART catalog table.

**index key**

The set of columns in a table that is used to determine the order of index entries.

**index partition**

A VSAM data set that is contained within a partitioning index space.

**index space**

A page set that is used to store the entries of one index.

**indicator column**

A 4-byte value that is stored in a base table in place of a LOB column.

**indicator variable**

A variable that is used to represent the null value in an application program. If the value for the selected column is null, a negative value is placed in the indicator variable.

**indoubt**

A status of a unit of recovery. If DB2 fails after it has finished its phase 1 commit processing and before it has started phase 2, only the commit coordinator knows if an individual unit of recovery is to be committed or rolled back. At restart, if DB2 lacks the information it needs to make this decision, the status of the unit of recovery is *indoubt* until DB2 obtains this information from the coordinator. More than one unit of recovery can be *indoubt* at restart.

**indoubt resolution**

The process of resolving the status of an *indoubt* logical unit of work to either the committed or the rollback state.

**inflight**

A status of a unit of recovery. If DB2 fails before its unit of recovery completes phase 1 of the commit process, it merely backs out the updates of its unit of recovery at restart. These units of recovery are termed *inflight*.

**inheritance**

The passing downstream of class resources or attributes from a parent class in the class hierarchy to a child class.

**initialization file**

For DB2 ODBC applications, a file containing values that can be set to adjust the performance of the database manager.

**inline copy**

A copy that is produced by the LOAD or REORG utility. The data set that the inline copy produces is logically equivalent to a full image copy that is produced by running the COPY utility with read-only access (SHRLEVEL REFERENCE).

**inline SQL PL**

A subset of SQL procedural language that can be used in SQL functions and dynamic compound statements. See also SQL procedural language.

**inner join**

The result of a join operation that includes only the matched rows of both tables that are being joined. See also join, equijoin, full outer join, left outer join, outer join, and right outer join.

| **inoperative package**

| In DB2 Version 9.1 for z/OS and earlier, a

| package that cannot be used because one  
| or more user-defined functions or  
| procedures that the package depends on  
| were dropped. Such a package must be  
| explicitly rebound. Contrast with invalid  
| package.

**insensitive cursor**

A cursor that is not sensitive to inserts, updates, or deletes that are made to the underlying rows of a result table after the result table has been materialized.

**insert trigger**

A trigger that is defined with the triggering SQL operation, an insert.

**install** The process of preparing a DB2 subsystem to operate as a z/OS subsystem.

| **INSTEAD OF trigger**

| A trigger that is associated with a single  
| view and is activated by an insert,  
| update, or delete operation on the view  
| and that can define how to propagate the  
| insert, update, or delete operation on the  
| view to the underlying tables of the view.  
| Contrast with BEFORE trigger and  
| AFTER trigger.

**instrumentation facility component identifier (IFCID)**

A value that names and identifies a trace record of an event that can be traced. As a parameter on the START TRACE and MODIFY TRACE commands, it specifies that the corresponding event is to be traced.

**instrumentation facility interface (IFI)**

A programming interface that enables programs to obtain online trace data about DB2, to submit DB2 commands, and to pass data to DB2.

**Interactive System Productivity Facility (ISPF)**

An IBM licensed program that provides interactive dialog services in a z/OS environment.

**inter-DB2 R/W interest**

A property of data in a table space, index, or partition that has been opened by more than one member of a data sharing group and that has been opened for writing by at least one of those members.

**intermediate database server**

The target of a request from a local

application or a remote application requester that is forwarded to another database server.

**internal resource lock manager (IRLM)**

A z/OS subsystem that DB2 uses to control communication and database locking.

| **intersection**

| An SQL operation that involves the  
| INTERSECT set operator, which combines  
| two result tables. The result of an  
| intersection operation consists of all of the  
| rows that are in both result tables.

| **invalid package**

| In DB2 Version 9.1 for z/OS and earlier, a  
| package that depends on an object (other  
| than a user-defined function) that is  
| dropped. Such a package is implicitly  
| rebound on invocation. Contrast with  
| inoperative package.

| **IP address**

| A value that uniquely identifies a TCP/IP host.

**IRLM** See internal resource lock manager.

**isolation level**

The degree to which a unit of work is isolated from the updating operations of other units of work. See also cursor stability, read stability, repeatable read, and uncommitted read.

**ISPF** See Interactive System Productivity Facility.

**iterator**

In SQLJ, an object that contains the result set of a query. An iterator is equivalent to a cursor in other host languages.

**iterator declaration clause**

In SQLJ, a statement that generates an iterator declaration class. An iterator is an object of an iterator declaration class.

**JAR** See Java Archive.

**Java Archive (JAR)**

A file format that is used for aggregating many files into a single file.

**JDBC** A Sun Microsystems database application programming interface (API) for Java that allows programs to access database management systems by using callable SQL.

**join** A relational operation that allows retrieval of data from two or more tables based on matching column values. See also equijoin, full outer join, inner join, left outer join, outer join, and right outer join.

**KB** Kilobyte. A value of 1024 bytes.

**Kerberos**

A network authentication protocol that is designed to provide strong authentication for client/server applications by using secret-key cryptography.

**Kerberos ticket**

A transparent application mechanism that transmits the identity of an initiating principal to its target. A simple ticket contains the principal's identity, a session key, a timestamp, and other information, which is sealed using the target's secret key.

**key** A column, an ordered collection of columns, or an expression that is identified in the description of a table, index, or referential constraint. The same column or expression can be part of more than one key.

**key-sequenced data set (KSDS)**

A VSAM file or data set whose records are loaded in key sequence and controlled by an index.

**KSDS** See key-sequenced data set.

**large object (LOB)**

A sequence of bytes representing bit data, single-byte characters, double-byte characters, or a mixture of single- and double-byte characters. A LOB can be up to 2 GB minus 1 byte in length. See also binary large object, character large object, and double-byte character large object.

**last agent optimization**

An optimized commit flow for either presumed-nothing or presumed-abort protocols in which the last agent, or final participant, becomes the commit coordinator. This flow saves at least one message.

**latch** A DB2 mechanism for controlling concurrent events or the use of system resources.

**LCID** See log control interval definition.

**LDS** See linear data set.

**leaf page**

An index page that contains pairs of keys and RIDs and that points to actual data. Contrast with nonleaf page.

**left outer join**

The result of a join operation that includes the matched rows of both tables that are being joined, and that preserves the unmatched rows of the first table. See also join, equijoin, full outer join, inner join, outer join, and right outer join.

**limit key**

The highest value of the partitioning key for a partition. The *partitioning key* is the column or columns that are used to determine the partitions.

**linear data set (LDS)**

A VSAM data set that contains data but no control information. A linear data set can be accessed as a byte-addressable string in virtual storage.

**linkage editor**

A computer program for creating load modules from one or more object modules or load modules by resolving cross references among the modules and, if necessary, adjusting addresses.

**link-edit**

The action of creating a loadable computer program using a linkage editor.

**list**

A type of object, which DB2 utilities can process, that identifies multiple table spaces, multiple index spaces, or both. A list is defined with the LISTDEF utility control statement.

**list structure**

A coupling facility structure that lets data be shared and manipulated as elements of a queue.

**L-lock** See logical lock.

**load module**

A program unit that is suitable for loading into main storage for execution. The output of a linkage editor.

**LOB** See large object.

**LOB locator**

A mechanism that allows an application program to manipulate a large object value in the database system. A LOB locator is a fullword integer value that



represents a single LOB value. An application program retrieves a LOB locator into a host variable and can then apply SQL operations to the associated LOB value using the locator.

**LOB lock**

A lock on a LOB value.

**LOB table space**

A table space that contains all the data for a particular LOB column in the related base table.

**local** A way of referring to any object that the local DB2 subsystem maintains. A *local table*, for example, is a table that is maintained by the local DB2 subsystem. Contrast with remote.

**locale** The definition of a subset of a user's environment that combines a CCSID and characters that are defined for a specific language and country.

**local lock**

A lock that provides intra-DB2 concurrency control, but not inter-DB2 concurrency control; that is, its scope is a single DB2.

**local subsystem**

The unique relational DBMS to which the user or application program is directly connected (in the case of DB2, by one of the DB2 attachment facilities).

**location**

The unique name of a database server. An application uses the location name to access a DB2 database server. A database alias can be used to override the location name when accessing a remote server.

**location alias**

Another name by which a database server identifies itself in the network. Applications can use this name to access a DB2 database server.

**lock** A means of controlling concurrent events or access to data. DB2 locking is performed by the IRLM.

**lock duration**

The interval over which a DB2 lock is held.

**lock escalation**

The promotion of a lock from a row, page, or LOB lock to a table space lock because

the number of page locks that are concurrently held on a given resource exceeds a preset limit.

**locking**

The process by which the integrity of data is ensured. Locking prevents concurrent users from accessing inconsistent data. See also claim, drain, and latch.

**lock mode**

A representation for the type of access that concurrently running programs can have to a resource that a DB2 lock is holding.

**lock object**

The resource that is controlled by a DB2 lock.

**lock promotion**

The process of changing the size or mode of a DB2 lock to a higher, more restrictive level.

**lock size**

The amount of data that is controlled by a DB2 lock on table data; the value can be a row, a page, a LOB, a partition, a table, or a table space.

**lock structure**

A coupling facility data structure that is composed of a series of lock entries to support shared and exclusive locking for logical resources.

**log**

A collection of records that describe the events that occur during DB2 execution and that indicate their sequence. The information thus recorded is used for recovery in the event of a failure during DB2 execution.

**log control interval definition**

A suffix of the physical log record that tells how record segments are placed in the physical control interval.

**logical claim**

A claim on a logical partition of a nonpartitioning index.

**logical index partition**

The set of all keys that reference the same data partition.

**logical lock (L-lock)**

The lock type that transactions use to control intra- and inter-DB2 data

	<p>concurrency between transactions. Contrast with physical lock (P-lock).</p>	
<b>logically complete</b>	<p>A state in which the concurrent copy process is finished with the initialization of the target objects that are being copied. The target objects are available for update.</p>	
<b>logical page list (LPL)</b>	<p>A list of pages that are in error and that cannot be referenced by applications until the pages are recovered. The page is in <i>logical error</i> because the actual media (coupling facility or disk) might not contain any errors. Usually a connection to the media has been lost.</p>	
<b>logical partition</b>	<p>A set of key or RID pairs in a nonpartitioning index that are associated with a particular partition.</p>	
<b>logical recovery pending (LRECP)</b>	<p>The state in which the data and the index keys that reference the data are inconsistent.</p>	
<b>logical unit (LU)</b>	<p>An access point through which an application program accesses the SNA network in order to communicate with another application program. See also LU name.</p>	
<b>logical unit of work</b>	<p>The processing that a program performs between synchronization points.</p>	
<b>logical unit of work identifier (LUWID)</b>	<p>A name that uniquely identifies a thread within a network. This name consists of a fully-qualified LU network name, an LUW instance number, and an LUW sequence number.</p>	
<b>log initialization</b>	<p>The first phase of restart processing during which DB2 attempts to locate the current end of the log.</p>	
<b>log record header (LRH)</b>	<p>A prefix, in every log record, that contains control information.</p>	
<b>log record sequence number (LRSN)</b>	<p>An identifier for a log record that is associated with a data sharing member. DB2 uses the LRSN for recovery in the data sharing environment.</p>	
<b>log truncation</b>	<p>A process by which an explicit starting RBA is established. This RBA is the point at which the next byte of log data is to be written.</p>	
<b>LPL</b>	See logical page list.	
<b>LRECP</b>	See logical recovery pending.	
<b>LRH</b>	See log record header.	
<b>LRSN</b>	See log record sequence number.	
<b>LU</b>	See logical unit.	
<b>LU name</b>	Logical unit name, which is the name by which VTAM refers to a node in a network.	
<b>LUW</b>	See logical unit of work.	
<b>LUWID</b>	See logical unit of work identifier.	
<b>mapping table</b>	<p>A table that the REORG utility uses to map the associations of the RIDs of data records in the original copy and in the shadow copy. This table is created by the user.</p>	
<b>mass delete</b>	The deletion of all rows of a table.	
<b>materialize</b>	<ul style="list-style-type: none"> <li>• The process of putting rows from a view or nested table expression into a work file for additional processing by a query.</li> <li>• The placement of a LOB value into contiguous storage. Because LOB values can be very large, DB2 avoids materializing LOB data until doing so becomes absolutely necessary.</li> </ul>	
<b>materialized query table</b>	<p>A table that is used to contain information that is derived and can be summarized from one or more source tables. Contrast with base table.</p>	
<b>MB</b>	Megabyte (1 048 576 bytes).	
<b>MBCS</b>	See multibyte character set.	
<b>member name</b>	The z/OS XCF identifier for a particular DB2 subsystem in a data sharing group.	
<b>menu</b>	A displayed list of available functions for	

selection by the operator. A menu is sometimes called a *menu panel*.

**metalanguage**

A language that is used to create other specialized languages.

**migration**

The process of converting a subsystem with a previous release of DB2 to an updated or current release. In this process, you can acquire the functions of the updated or current release without losing the data that you created on the previous release.

**mixed data string**

A character string that can contain both single-byte and double-byte characters.

**mode name**

A VTAM name for the collection of physical and logical characteristics and attributes of a session.

**modify locks**

An L-lock or P-lock with a MODIFY attribute. A list of these active locks is kept at all times in the coupling facility lock structure. If the requesting DB2 subsystem fails, that DB2 subsystem's modify locks are converted to retained locks.

**multibyte character set (MBCS)**

A character set that represents single characters with more than a single byte. UTF-8 is an example of an MBCS. Characters in UTF-8 can range from 1 to 4 bytes in DB2. Contrast with single-byte character set and double-byte character set. See also Unicode.

**multidimensional analysis**

The process of assessing and evaluating an enterprise on more than one level.

**Multiple Virtual Storage (MVS)**

An element of the z/OS operating system. This element is also called the Base Control Program (BCP).

**multisite update**

Distributed relational database processing in which data is updated in more than one location within a single unit of work.

**multithreading**

Multiple TCBs that are executing one copy of DB2 ODBC code concurrently

(sharing a processor) or in parallel (on separate central processors).

**MVS** See Multiple Virtual Storage.

**native SQL procedure**

An SQL procedure that is processed by converting the procedural statements to a native representation that is stored in the database directory, as is done with other SQL statements. When a native SQL procedure is called, the native representation is loaded from the directory, and DB2 executes the procedure. Contrast with external procedure and external SQL procedure.

**nested table expression**

A fullselect in a FROM clause (surrounded by parentheses).

**network identifier (NID)**

The network ID that is assigned by IMS or CICS, or if the connection type is RRSAF, the RRS unit of recovery ID (URID).

**new-function mode (NFM)**

The normal mode of operation that exists after successful completion of a version-to-version migration. At this stage, all new functions of the new version are available for use. A DB2 data sharing group cannot coexist with members that are still at the prior version level, and fallback to a prior version is not supported. Contrast with conversion mode, conversion mode\*, enabling-new-function mode, and enabling-new-function mode\*.

**NFM** See new-function mode.

**NID** See network identifier.

**node ID index**

See XML node ID index.

**nondeterministic function**

A user-defined function whose result is not solely dependent on the values of the input arguments. That is, successive invocations with the same argument values can produce a different answer. This type of function is sometimes called a *variant* function. Contrast with deterministic function (sometimes called a *not-variant function*).

**nonleaf page**

A page that contains keys and page



numbers of other pages in the index (either leaf or nonleaf pages). Nonleaf pages never point to actual data. Contrast with leaf page.

**nonpartitioned index**

An index that is not physically partitioned. Both partitioning indexes and secondary indexes can be nonpartitioned.

**nonpartitioned secondary index (NPSI)**

An index on a partitioned table space that is not the partitioning index and is not partitioned. Contrast with data-partitioned secondary index.

**nonpartitioning index**

See secondary index.

**nonscrollable cursor**

A cursor that can be moved only in a forward direction. Nonscrollable cursors are sometimes called forward-only cursors or serial cursors.

**normalization**

A key step in the task of building a logical relational database design. Normalization helps you avoid redundancies and inconsistencies in your data. An entity is normalized if it meets a set of constraints for a particular normal form (first normal form, second normal form, and so on). Contrast with denormalization.

**not-variant function**

See deterministic function.

**NPSI** See nonpartitioned secondary index.

**NUL** The null character ('\0'), which is represented by the value X'00'. In C, this character denotes the end of a string.

**null** A special value that indicates the absence of information.

**null terminator**

In C, the value that indicates the end of a string. For EBCDIC, ASCII, and Unicode UTF-8 strings, the null terminator is a single-byte value (X'00'). For Unicode UTF-16 or UCS-2 (wide) strings, the null terminator is a double-byte value (X'0000').

**ODBC**

See Open Database Connectivity.

**ODBC driver**

A dynamically-linked library (DLL) that

implements ODBC function calls and interacts with a data source.

**OLAP** See online analytical processing.

**online analytical processing (OLAP)**

The process of collecting data from one or many sources; transforming and analyzing the consolidated data quickly and interactively; and examining the results across different dimensions of the data by looking for patterns, trends, and exceptions within complex relationships of that data.

**Open Database Connectivity (ODBC)**

A Microsoft database application programming interface (API) for C that allows access to database management systems by using callable SQL. ODBC does not require the use of an SQL preprocessor. In addition, ODBC provides an architecture that lets users add modules called *database drivers*, which link the application to their choice of database management systems at run time. This means that applications no longer need to be directly linked to the modules of all the database management systems that are supported.

**ordinary identifier**

An uppercase letter followed by zero or more characters, each of which is an uppercase letter, a digit, or the underscore character. An ordinary identifier must not be a reserved word.

**ordinary token**

A numeric constant, an ordinary identifier, a host identifier, or a keyword.

**originating task**

In a parallel group, the primary agent that receives data from other execution units (referred to as *parallel tasks*) that are executing portions of the query in parallel.

**outer join**

The result of a join operation that includes the matched rows of both tables that are being joined and preserves some or all of the unmatched rows of the tables that are being joined. See also join, equijoin, full outer join, inner join, left outer join, and right outer join.

**overloaded function**

A function name for which multiple function instances exist.

**package**

An object containing a set of SQL statements that have been statically bound and that is available for processing. A package is sometimes also called an *application package*.

**package list**

An ordered list of package names that may be used to extend an application plan.

**package name**

The name of an object that is used for an application package or an SQL procedure package. An application package is a bound version of a database request module (DBRM) that is created by a BIND PACKAGE or REBIND PACKAGE command. An SQL procedural language package is created by a CREATE or ALTER PROCEDURE statement for a native SQL procedure. The name of a package consists of a location name, a collection ID, a package ID, and a version ID.

**page**

A unit of storage within a table space (4 KB, 8 KB, 16 KB, or 32 KB) or index space (4 KB, 8 KB, 16 KB, or 32 KB). In a table space, a page contains one or more rows of a table. In a LOB or XML table space, a LOB or XML value can span more than one page, but no more than one LOB or XML value is stored on a page.

**page set**

Another way to refer to a table space or index space. Each page set consists of a collection of VSAM data sets.

**page set recovery pending (PSRCP)**

A restrictive state of an index space. In this case, the entire page set must be recovered. Recovery of a logical part is prohibited.

**panel**

A predefined display image that defines the locations and characteristics of display fields on a display surface (for example, a *menu panel*).

**parallel complex**

A cluster of machines that work together to handle multiple transactions and applications.

**parallel group**

A set of consecutive operations that execute in parallel and that have the same number of parallel tasks.

**parallel I/O processing**

A form of I/O processing in which DB2 initiates multiple concurrent requests for a single user query and performs I/O processing concurrently (in *parallel*) on multiple data partitions.

**Parallel Sysplex**

A set of z/OS systems that communicate and cooperate with each other through certain multisystem hardware components and software services to process customer workloads.

**parallel task**

The execution unit that is dynamically created to process a query in parallel. A parallel task is implemented by a z/OS service request block.

**parameter marker**

A question mark (?) that appears in a statement string of a dynamic SQL statement. The question mark can appear where a variable could appear if the statement string were a static SQL statement.

**parameter-name**

An SQL identifier that designates a parameter in a routine that is written by a user. Parameter names are required for SQL procedures and SQL functions, and they are used in the body of the routine to refer to the values of the parameters. Parameter names are optional for external routines.

**parent key**

A primary key or unique key in the parent table of a referential constraint. The values of a parent key determine the valid values of the foreign key in the referential constraint.

**parent lock**

For explicit hierarchical locking, a lock that is held on a resource that might have child locks that are lower in the hierarchy. A parent lock is usually the table space lock or the partition intent lock. See also child lock.

**parent row**

A row whose primary key value is the foreign key value of a dependent row.

**parent table**

A table whose primary key is referenced by the foreign key of a dependent table.

**parent table space**

A table space that contains a parent table.  
A table space containing a dependent of that table is a dependent table space.

**participant**

An entity other than the commit coordinator that takes part in the commit process. The term participant is synonymous with agent in SNA.

**partition**

A portion of a page set. Each partition corresponds to a single, independently extendable data set. The maximum size of a partition depends on the number of partitions in the partitioned page set. All partitions of a given page set have the same maximum size.

**partition-by-growth table space**

A table space whose size can grow to accommodate data growth. DB2 for z/OS manages partition-by-growth table spaces by automatically adding new data sets when the database needs more space to satisfy an insert operation. Contrast with range-partitioned table space. See also universal table space.

**partitioned data set (PDS)**

A data set in disk storage that is divided into partitions, which are called members. Each partition can contain a program, part of a program, or data. A program library is an example of a partitioned data set.

**partitioned index**

An index that is physically partitioned. Both partitioning indexes and secondary indexes can be partitioned.

**partitioned page set**

A partitioned table space or an index space. Header pages, space map pages, data pages, and index pages reference data only within the scope of the partition.

**partitioned table space**

A table space that is based on a single

table and that is subdivided into partitions, each of which can be processed independently by utilities. Contrast with segmented table space and universal table space.

**partitioning index**

An index in which the leftmost columns are the partitioning columns of the table. The index can be partitioned or nonpartitioned.

**partner logical unit**

An access point in the SNA network that is connected to the local DB2 subsystem by way of a VTAM conversation.

**path**

See SQL path.

**PDS**

See partitioned data set.

**period**

In a table, an interval of time that is defined by two datetime columns. A period contains a begin column and an end column. See also begin column and end column.

**physical consistency**

The state of a page that is not in a partially changed state.

**physical lock (P-lock)**

A type of lock that DB2 acquires to provide consistency of data that is cached in different DB2 subsystems. Physical locks are used only in data sharing environments. Contrast with logical lock (L-lock).

**physically complete**

The state in which the concurrent copy process is completed and the output data set has been created.

**piece**

A data set of a nonpartitioned page set.

**plan**

See application plan.

**plan allocation**

The process of allocating DB2 resources to a plan in preparation for execution.

**plan member**

The bound copy of a DBRM that is identified in the member clause.

**plan name**

The name of an application plan.

**P-lock**

See physical lock.

**point of consistency**

A time when all recoverable data that an

application accesses is consistent with other data. The term point of consistency is synonymous with sync point or commit point.

**policy** See CFRM policy.

**postponed abort UR**

A unit of recovery that was in-flight or in-abort, was interrupted by system failure or cancellation, and did not complete backout during restart.

**precision**

In SQL, the total number of digits in a decimal number (called the *size* in the C language). In the C language, the number of digits to the right of the decimal point (called the *scale* in SQL). The DB2 information uses the SQL terms.

**precompilation**

A processing of application programs containing SQL statements that takes place before compilation. SQL statements are replaced with statements that are recognized by the host language compiler. Output from this precompilation includes source code that can be submitted to the compiler and the database request module (DBRM) that is input to the bind process.

**predicate**

An element of a search condition that expresses or implies a comparison operation.

**prefix** A code at the beginning of a message or record.

**preformat**

The process of preparing a VSAM linear data set for DB2 use, by writing specific data patterns.

**prepare**

The first phase of a two-phase commit process in which all participants are requested to prepare for commit.

**prepared SQL statement**

A named object that is the executable form of an SQL statement that has been processed by the PREPARE statement.

**primary authorization ID**

The authorization ID that is used to identify the application process to DB2.

**primary group buffer pool**

For a duplexed group buffer pool, the structure that is used to maintain the coherency of cached data. This structure is used for page registration and cross-invalidation. The z/OS equivalent is *old* structure. Compare with secondary group buffer pool.

**primary index**

An index that enforces the uniqueness of a primary key.

**primary key**

In a relational database, a unique, nonnull key that is part of the definition of a table. A table cannot be defined as a parent unless it has a unique key or primary key.

**principal**

An entity that can communicate securely with another entity. In Kerberos, principals are represented as entries in the Kerberos registry database and include users, servers, computers, and others.

**principal name**

The name by which a principal is known to the DCE security services.

**privilege**

The capability of performing a specific function, sometimes on a specific object. See also explicit privilege.

**privilege set**

- For the installation SYSADM ID, the set of all possible privileges.
- For any other authorization ID, including the PUBLIC authorization ID, the set of all privileges that are recorded for that ID in the DB2 catalog.

**process**

In DB2, the unit to which DB2 allocates resources and locks. Sometimes called an application process, a process involves the execution of one or more programs. The execution of an SQL statement is always associated with some process. The means of initiating and terminating a process are dependent on the environment.

**program**

A single, compilable collection of executable statements in a programming language.

**program temporary fix (PTF)**

A solution or bypass of a problem that is diagnosed as a result of a defect in a current unaltered release of a licensed program. An authorized program analysis report (APAR) fix is corrective service for an existing problem. A PTF is preventive service for problems that might be encountered by other users of the product. A PTF is *temporary*, because a permanent fix is usually not incorporated into the product until its next release.

**protected conversation**

A VTAM conversation that supports two-phase commit flows.

**PSRCP**

See page set recovery pending.

**PTF** See program temporary fix.

**QSAM**

See queued sequential access method.

**query** A component of certain SQL statements that specifies a result table.

**query block**

The part of a query that is represented by one of the FROM clauses. Each FROM clause can have multiple query blocks, depending on DB2 processing of the query.

**query CP parallelism**

Parallel execution of a single query, which is accomplished by using multiple tasks.

**query I/O parallelism**

Parallel access of data, which is accomplished by triggering multiple I/O requests within a single query.

**queued sequential access method (QSAM)**

An extended version of the basic sequential access method (BSAM). When this method is used, a queue of data blocks is formed. Input data blocks await processing, and output data blocks await transfer to auxiliary storage or to an output device.

**quiesce point**

A point at which data is consistent as a result of running the DB2 QUIESCE utility.

**RACF** Resource Access Control Facility. A component of the z/OS Security Server.

**range-partitioned table space**

A type of universal table space that is based on partitioning ranges and that contains a single table. Contrast with partition-by-growth table space. See also universal table space.

**RBA** See relative byte address.

**RCT** See resource control table.

**RDO** See resource definition online.

**read stability (RS)**

An isolation level that is similar to repeatable read but does not completely isolate an application process from all other concurrently executing application processes. See also cursor stability, repeatable read, and uncommitted read.

**rebind**

The creation of a new application plan for an application program that has been bound previously. If, for example, you have added an index for a table that your application accesses, you must rebind the application in order to take advantage of that index.

**rebuild**

The process of reallocating a coupling facility structure. For the shared communications area (SCA) and lock structure, the structure is repopulated; for the group buffer pool, changed pages are usually cast out to disk, and the new structure is populated only with changed pages that were not successfully cast out.

**record** The storage representation of a row or other data.

**record identifier (RID)**

A unique identifier that DB2 uses to identify a row of data in a table. Compare with row identifier.

**record identifier (RID) pool**

An area of main storage that is used for sorting record identifiers during list-prefetch processing.

**record length**

The sum of the length of all the columns in a table, which is the length of the data as it is physically stored in the database. Records can be fixed length or varying length, depending on how the columns are defined. If all columns are



fixed-length columns, the record is a fixed-length record. If one or more columns are varying-length columns, the record is a varying-length record.

**Resource Recovery Services attachment facility (RRSAF)**

A DB2 subcomponent that uses Resource Recovery Services to coordinate resource commitment between DB2 and all other resource managers that also use RRS in a z/OS system.

**recovery**

The process of rebuilding databases after a system failure.

**recovery log**

A collection of records that describes the events that occur during DB2 execution and indicates their sequence. The recorded information is used for recovery in the event of a failure during DB2 execution.

**recovery manager**

A subcomponent that supplies coordination services that control the interaction of DB2 resource managers during commit, abort, checkpoint, and restart processes. The recovery manager also supports the recovery mechanisms of other subsystems (for example, IMS) by acting as a participant in the other subsystem's process for protecting data that has reached a point of consistency.

A coordinator or a participant (or both), in the execution of a two-phase commit, that can access a recovery log that maintains the state of the logical unit of work and names the immediate upstream coordinator and downstream participants.

**recovery pending (RECP)**

A condition that prevents SQL access to a table space that needs to be recovered.

**recovery token**

An identifier for an element that is used in recovery (for example, NID or URID).

**RECP** See recovery pending.

**redo**

A state of a unit of recovery that indicates that changes are to be reapplied to the disk media to ensure data integrity.

**reentrant code**

Executable code that can reside in storage as one shared copy for all threads.

Reentrant code is not self-modifying and provides separate storage areas for each thread. See also threadsafe.

**referential constraint**

The requirement that nonnull values of a designated foreign key are valid only if they equal values of the primary key of a designated table.

**referential cycle**

A set of referential constraints such that each base table in the set is a descendent of itself. The tables that are involved in a referential cycle are ordered so that each table is a descendent of the one before it, and the first table is a descendent of the last table.

**referential integrity**

The state of a database in which all values of all foreign keys are valid. Maintaining referential integrity requires the enforcement of referential constraints on all operations that change the data in a table on which the referential constraints are defined.

**referential structure**

A set of tables and relationships that includes at least one table and, for every table in the set, all the relationships in which that table participates and all the tables to which it is related.

**refresh age**

The time duration between the current time and the time during which a materialized query table was last refreshed.

**registry**

See registry database.

**registry database**

A database of security information about principals, groups, organizations, accounts, and security policies.

**relational database**

A database that can be perceived as a set of tables and manipulated in accordance with the relational model of data.

**relational database management system (RDBMS)**

A collection of hardware and software that organizes and provides access to a relational database.

| **relational schema**

| See SQL schema.

**relationship**

A defined connection between the rows of a table or the rows of two tables. A relationship is the internal representation of a referential constraint.

**relative byte address (RBA)**

The offset of a data record or control interval from the beginning of the storage space that is allocated to the data set or file to which it belongs.

**remigration**

The process of returning to a current release of DB2 following a fallback to a previous release. This procedure constitutes another migration process.

**remote**

Any object that is maintained by a remote DB2 subsystem (that is, by a DB2 subsystem other than the local one). A *remote view*, for example, is a view that is maintained by a remote DB2 subsystem. Contrast with local.

**remote subsystem**

Any relational DBMS, except the local subsystem, with which the user or application can communicate. The subsystem need not be remote in any physical sense, and might even operate on the same processor under the same z/OS system.

**reoptimization**

The DB2 process of reconsidering the access path of an SQL statement at run time; during reoptimization, DB2 uses the values of host variables, parameter markers, or special registers.

| **reordered row format**

| A row format that facilitates improved  
| performance in retrieval of rows that have  
| varying-length columns. DB2 rearranges  
| the column order, as defined in the  
| CREATE TABLE statement, so that the  
| fixed-length columns are stored at the  
| beginning of the row and the  
| varying-length columns are stored at the  
| end of the row. Contrast with basic row  
| format.

**REORG pending (REORP)**

A condition that restricts SQL access and most utility access to an object that must be reorganized.

**REORP**

See REORG pending.

**repeatable read (RR)**

The isolation level that provides maximum protection from other executing application programs. When an application program executes with repeatable read protection, rows that the program references cannot be changed by other programs until the program reaches a commit point. See also cursor stability, read stability, and uncommitted read.

**repeating group**

A situation in which an entity includes multiple attributes that are inherently the same. The presence of a repeating group violates the requirement of first normal form. In an entity that satisfies the requirement of first normal form, each attribute is independent and unique in its meaning and its name. See also normalization.

**replay detection mechanism**

A method that allows a principal to detect whether a request is a valid request from a source that can be trusted or whether an untrustworthy entity has captured information from a previous exchange and is replaying the information exchange to gain access to the principal.

**request commit**

The vote that is submitted to the prepare phase if the participant has modified data and is prepared to commit or roll back.

**requester**

The source of a request to access data at a remote server. In the DB2 environment, the requester function is provided by the distributed data facility.

**resource**

The object of a lock or claim, which could be a table space, an index space, a data partition, an index partition, or a logical partition.

**resource allocation**

The part of plan allocation that deals specifically with the database resources.

### resource control table

A construct of previous versions of the CICS attachment facility that defines authorization and access attributes for transactions or transaction groups. Beginning in CICS Transaction Server Version 1.3, resources are defined by using resource definition online instead of the resource control table. See also resource definition online.

### resource definition online (RDO)

The recommended method of defining resources to CICS by creating resource definitions interactively, or by using a utility, and then storing them in the CICS definition data set. In earlier releases of CICS, resources were defined by using the resource control table (RCT), which is no longer supported.

### resource limit facility (RLF)

A portion of DB2 code that prevents dynamic manipulative SQL statements from exceeding specified time limits. The resource limit facility is sometimes called the governor.

### resource limit specification table (RLST)

A site-defined table that specifies the limits to be enforced by the resource limit facility.

### resource manager

- A function that is responsible for managing a particular resource and that guarantees the consistency of all updates made to recoverable resources within a logical unit of work. The resource that is being managed can be physical (for example, disk or main storage) or logical (for example, a particular type of system service).
- A participant, in the execution of a two-phase commit, that has recoverable resources that could have been modified. The resource manager has access to a recovery log so that it can commit or roll back the effects of the logical unit of work to the recoverable resources.

### restart pending (RESTOP)

A restrictive state of a page set or partition that indicates that restart (backout) work needs to be performed on the object.

### RESTOP

See restart pending.

### result set

The set of rows that a stored procedure returns to a client application.

### result set locator

A 4-byte value that DB2 uses to uniquely identify a query result set that a stored procedure returns.

### result table

The set of rows that are specified by a SELECT statement.

### retained lock

A MODIFY lock that a DB2 subsystem was holding at the time of a subsystem failure. The lock is retained in the coupling facility lock structure across a DB2 for z/OS failure.

### RID

See record identifier.

### RID pool

See record identifier pool.

### right outer join

The result of a join operation that includes the matched rows of both tables that are being joined and preserves the unmatched rows of the second join operand. See also join, equijoin, full outer join, inner join, left outer join, and outer join.

### RLF

See resource limit facility.

### RLST

See resource limit specification table.

### role

A database entity that groups together one or more privileges and that can be assigned to a primary authorization ID or to PUBLIC. The role is available only in a trusted context.

### rollback

The process of restoring data that was changed by SQL statements to the state at its last commit point. All locks are freed. Contrast with commit.

### root page

The index page that is at the highest level (or the beginning point) in an index.

### routine

A database object that encapsulates procedural logic and SQL statements, is stored on the database server, and can be invoked from an SQL statement or by



| using the CALL statement. The main  
| classes of routines are procedures and  
| functions.

**row** The horizontal component of a table. A  
row consists of a sequence of values, one  
for each column of the table.

| **row-begin column**  
| A generated column that is defined with  
| the AS ROW BEGIN clause. The value is  
| assigned whenever a row is inserted into  
| the table or any column in the row is  
| updated. A row-begin column is intended  
| for use as the first column of a  
| SYSTEM\_TIME period. See also generated  
| column, row-end column, and  
| transaction-start-ID column.

| **row change timestamp column**  
| A generated column that is defined with  
| the AS ROW CHANGE TIMESTAMP  
| clause. A row change timestamp column  
| provides a way for the database manager  
| to automatically generate and maintain a  
| timestamp value for each row that is  
| inserted or updated in a table. A table can  
| have no more than one row change  
| timestamp column.

| **row-end column**  
| A generated column that is defined with  
| the AS ROW END clause. The value is  
| assigned whenever a row is inserted into  
| the table or any column in the row is  
| updated. A row-end column is intended  
| for use as the second column of a  
| SYSTEM\_TIME period. See also generated  
| column, row-begin column, and  
| transaction-start-ID column.

| **rowid** A value that uniquely identifies a row in  
| a table and does not change.

**row lock**  
A lock on a single row of data.

**row-positioned fetch orientation**  
| The specification of the desired placement  
| of the cursor as part of a FETCH  
| statement, with respect to a single row  
| (for example, NEXT, LAST, or ABSOLUTE  
| *n*). Contrast with rowset-positioned fetch  
| orientation.

**rowset**  
A set of rows for which a cursor position  
is established.

**rowset cursor**  
A cursor that is defined so that one or  
more rows can be returned as a rowset  
for a single FETCH statement, and the  
cursor is positioned on the set of rows  
that is fetched.

**rowset-positioned fetch orientation**  
The specification of the desired placement  
of the cursor as part of a FETCH  
statement, with respect to a rowset (for  
example, NEXT ROWSET, LAST  
ROWSET, or ROWSET STARTING AT  
ABSOLUTE *n*). Contrast with  
row-positioned fetch orientation.

**row trigger**  
A trigger that is defined with the trigger  
granularity FOR EACH ROW.

**RRSAF**  
See Resource Recovery Services  
attachment facility.

**RS** See read stability.

**savepoint**  
A named entity that represents the state  
of data and schemas at a particular point  
in time within a unit of work.

**SBCS** See single-byte character set.

**SCA** See shared communications area.

**scalar function**  
An SQL operation that produces a single  
value from another value and is  
expressed as a function name, followed  
by a list of arguments that are enclosed in  
parentheses.

**scale** In SQL, the number of digits to the right  
of the decimal point (called the precision  
in the C language). The DB2 information  
uses the SQL definition.

**schema**  
The organization or structure of a  
database.  
  
A collection of, and a way of qualifying,  
database objects such as tables, views,  
routines, indexes or triggers that define a  
database. A database schema provides a  
logical classification of database objects.

**scrollability**  
The ability to use a cursor to fetch in  
either a forward or backward direction.  
The FETCH statement supports multiple

fetch orientations to indicate the new position of the cursor. See also fetch orientation.

**scrollable cursor**

A cursor that can be moved in both a forward and a backward direction.

**search condition**

A criterion for selecting rows from a table. A search condition consists of one or more predicates.

**secondary authorization ID**

An authorization ID that has been associated with a primary authorization ID by an authorization exit routine.

**secondary group buffer pool**

For a duplexed group buffer pool, the structure that is used to back up changed pages that are written to the primary group buffer pool. No page registration or cross-invalidation occurs using the secondary group buffer pool. The z/OS equivalent is *new* structure.

**secondary index**

A nonpartitioning index that is useful for enforcing a uniqueness constraint, for clustering data, or for providing access paths to data for queries. A secondary index can be partitioned or nonpartitioned. See also data-partitioned secondary index (DPSI) and nonpartitioned secondary index (NPSI).

**section**

The segment of a plan or package that contains the executable structures for a single SQL statement. For most SQL statements, one section in the plan exists for each SQL statement in the source program. However, for cursor-related statements, the DECLARE, OPEN, FETCH, and CLOSE statements reference the same section because they each refer to the SELECT statement that is named in the DECLARE CURSOR statement. SQL statements such as COMMIT, ROLLBACK, and some SET statements do not use a section.

| **security label**

| A classification of users' access to objects  
| or data rows in a multilevel security  
| environment."

**segment**

A group of pages that holds rows of a single table. See also segmented table space.

**segmented table space**

A table space that is divided into equal-sized groups of pages called segments. Segments are assigned to tables so that rows of different tables are never stored in the same segment. Contrast with partitioned table space and universal table space.

**self-referencing constraint**

A referential constraint that defines a relationship in which a table is a dependent of itself.

**self-referencing table**

A table with a self-referencing constraint.

**sensitive cursor**

A cursor that is sensitive to changes that are made to the database after the result table has been materialized.

**sequence**

A user-defined object that generates a sequence of numeric values according to user specifications.

**sequential data set**

A non-DB2 data set whose records are organized on the basis of their successive physical positions, such as on magnetic tape. Several of the DB2 database utilities require sequential data sets.

**sequential prefetch**

A mechanism that triggers consecutive asynchronous I/O operations. Pages are fetched before they are required, and several pages are read with a single I/O operation.

**serialized profile**

A Java object that contains SQL statements and descriptions of host variables. The SQLJ translator produces a serialized profile for each connection context.

**server** The target of a request from a remote requester. In the DB2 environment, the server function is provided by the distributed data facility, which is used to access DB2 data from remote applications.

**service class**

An eight-character identifier that is used

by the z/OS Workload Manager to associate user performance goals with a particular DDF thread or stored procedure. A service class is also used to classify work on parallelism assistants.

**service request block**

A unit of work that is scheduled to execute.

**session**

A link between two nodes in a VTAM network.

**session protocols**

The available set of SNA communication requests and responses.

**set operator**

The SQL operators UNION, EXCEPT, and INTERSECT corresponding to the relational operators union, difference, and intersection. A set operator derives a result table by combining two other result tables.

**shared communications area (SCA)**

A coupling facility list structure that a DB2 data sharing group uses for inter-DB2 communication.

**share lock**

A lock that prevents concurrently executing application processes from changing data, but not from reading data. Contrast with exclusive lock.

**shift-in character**

A special control character (X'0F') that is used in EBCDIC systems to denote that the subsequent bytes represent SBCS characters. See also shift-out character.

**shift-out character**

A special control character (X'0E') that is used in EBCDIC systems to denote that the subsequent bytes, up to the next shift-in control character, represent DBCS characters. See also shift-in character.

**sign-on**

A request that is made on behalf of an individual CICS or IMS application process by an attachment facility to enable DB2 to verify that it is authorized to use DB2 resources.

**simple page set**

A nonpartitioned page set. A simple page set initially consists of a single data set (page set piece). If and when that data set

is extended to 2 GB, another data set is created, and so on, up to a total of 32 data sets. DB2 considers the data sets to be a single contiguous linear address space containing a maximum of 64 GB. Data is stored in the next available location within this address space without regard to any partitioning scheme.

**simple table space**

A table space that is neither partitioned nor segmented. Creation of simple table spaces is not supported in DB2 Version 9.1 for z/OS. Contrast with partitioned table space, segmented table space, and universal table space.

**single-byte character set (SBCS)**

A set of characters in which each character is represented by a single byte. Contrast with double-byte character set or multibyte character set.

**single-precision floating point number**

A 32-bit approximate representation of a real number.

**SMP/E**

See System Modification Program/Extended.

**SNA** See Systems Network Architecture.

**SNA network**

The part of a network that conforms to the formats and protocols of Systems Network Architecture (SNA).

**socket** A callable TCP/IP programming interface that TCP/IP network applications use to communicate with remote TCP/IP partners.

**sourced function**

A function that is implemented by another built-in or user-defined function that is already known to the database manager. This function can be a scalar function or an aggregate function; it returns a single value from a set of values (for example, MAX or AVG). Contrast with built-in function, external function, and SQL function.

**source program**

A set of host language statements and SQL statements that is processed by an SQL precompiler.

**source table**

A table that can be a base table, a view, a table expression, or a user-defined table function.

**source type**

An existing type that DB2 uses to represent a distinct type.

**space** A sequence of one or more blank characters.

**special register**

A storage area that DB2 defines for an application process to use for storing information that can be referenced in SQL statements. Examples of special registers are SESSION\_USER and CURRENT DATE.

**specific function name**

A particular user-defined function that is known to the database manager by its specific name. Many specific user-defined functions can have the same function name. When a user-defined function is defined to the database, every function is assigned a specific name that is unique within its schema. Either the user can provide this name, or a default name is used.

**SPUFI** See SQL Processor Using File Input.

**SQL** See Structured Query Language.

**SQL authorization ID (SQL ID)**

The authorization ID that is used for checking dynamic SQL statements in some situations.

**SQLCA**

See SQL communication area.

**SQL communication area (SQLCA)**

A structure that is used to provide an application program with information about the execution of its SQL statements.

**SQL connection**

An association between an application process and a local or remote application server or database server.

**SQLDA**

See SQL descriptor area.

**SQL descriptor area (SQLDA)**

A structure that describes input variables, output variables, or the columns of a result table.

**SQL escape character**

The symbol that is used to enclose an SQL delimited identifier. This symbol is the double quotation mark ("). See also escape character.

**SQL function**

A user-defined function in which the CREATE FUNCTION statement contains the source code. The source code is a single SQL expression that evaluates to a single value. The SQL user-defined function can return the result of an expression. See also built-in function, external function, and sourced function.

**SQL ID**

See SQL authorization ID.

**SQLJ** Structured Query Language (SQL) that is embedded in the Java programming language.

**SQL path**

An ordered list of schema names that are used in the resolution of unqualified references to user-defined functions, distinct types, and stored procedures. In dynamic SQL, the SQL path is found in the CURRENT PATH special register. In static SQL, it is defined in the PATH bind option.

**SQL PL**

See SQL procedural language.

**SQL procedural language (SQL PL)**

A language extension of SQL that consists of statements and language elements that can be used to implement procedural logic in SQL statements. SQL PL provides statements for declaring variables and condition handlers, assigning values to variables, and for implementing procedural logic. See also inline SQL PL.

**SQL procedure**

A user-written program that can be invoked with the SQL CALL statement. An SQL procedure is written in the SQL procedural language. Two types of SQL procedures are supported: external SQL procedures and native SQL procedures. See also external procedure and native SQL procedure.

**SQL processing conversation**

Any conversation that requires access of DB2 data, either through an application or by dynamic query requests.

**SQL Processor Using File Input (SPUFI)**

A facility of the TSO attachment subcomponent that enables the DB2I user to execute SQL statements without embedding them in an application program.

**SQL return code**

Either SQLCODE or SQLSTATE.

**SQL routine**

A user-defined function or stored procedure that is based on code that is written in SQL.

**SQL schema**

A collection of database objects such as tables, views, indexes, functions, distinct types, schemas, or triggers that defines a database. An SQL schema provides a logical classification of database objects.

**SQL statement coprocessor**

An alternative to the DB2 precompiler that lets the user process SQL statements at compile time. The user invokes an SQL statement coprocessor by specifying a compiler option.

**SQL string delimiter**

A symbol that is used to enclose an SQL string constant. The SQL string delimiter is the apostrophe ('), except in COBOL applications, where the user assigns the symbol, which is either an apostrophe or a double quotation mark (").

**SRB** See service request block.

**stand-alone**

An attribute of a program that means that it is capable of executing separately from DB2, without using DB2 services.

**star join**

A method of joining a dimension column of a fact table to the key column of the corresponding dimension table. See also join, dimension, and star schema.

**star schema**

The combination of a fact table (which contains most of the data) and a number of dimension tables. See also star join, dimension, and dimension table.

**statement handle**

In DB2 ODBC, the data object that contains information about an SQL statement that is managed by DB2 ODBC. This includes information such as

dynamic arguments, bindings for dynamic arguments and columns, cursor information, result values, and status information. Each statement handle is associated with the connection handle.

**statement string**

For a dynamic SQL statement, the character string form of the statement.

**statement trigger**

A trigger that is defined with the trigger granularity FOR EACH STATEMENT.

**static cursor**

A named control structure that does not change the size of the result table or the order of its rows after an application opens the cursor. Contrast with dynamic cursor.

**static SQL**

SQL statements, embedded within a program, that are prepared during the program preparation process (before the program is executed). After being prepared, the SQL statement does not change (although values of variables that are specified by the statement might change).

**storage group**

A set of storage objects on which DB2 for z/OS data can be stored. A storage object can have an SMS data class, a management class, a storage class, and a list of volume serial numbers.

**stored procedure**

A user-written application program that can be invoked through the use of the SQL CALL statement. Stored procedures are sometimes called procedures.

**string** See binary string, character string, or graphic string.

**strong typing**

A process that guarantees that only user-defined functions and operations that are defined on a distinct type can be applied to that type. For example, you cannot directly compare two currency types, such as Canadian dollars and U.S. dollars. But you can provide a user-defined function to convert one currency to the other and then do the comparison.

**structure**



- A name that refers collectively to different types of DB2 objects, such as tables, databases, views, indexes, and table spaces.
- A construct that uses z/OS to map and manage storage on a coupling facility. See also cache structure, list structure, or lock structure.

### **Structured Query Language (SQL)**

A standardized language for defining and manipulating data in a relational database.

### **structure owner**

In relation to group buffer pools, the DB2 member that is responsible for the following activities:

- Coordinating rebuild, checkpoint, and damage assessment processing
- Monitoring the group buffer pool threshold and notifying castout owners when the threshold has been reached

### **subcomponent**

A group of closely related DB2 modules that work together to provide a general function.

### **subject table**

The table for which a trigger is created. When the defined triggering event occurs on this table, the trigger is activated.

### **subquery**

A SELECT statement within the WHERE or HAVING clause of another SQL statement; a nested SQL statement.

### **subselect**

That form of a query that includes only a SELECT clause, FROM clause, and optionally a WHERE clause, GROUP BY clause, HAVING clause, ORDER BY clause, or FETCH FIRST clause.

### **substitution character**

A unique character that is substituted during character conversion for any characters in the source program that do not have a match in the target coding representation.

### **subsystem**

In z/OS, a service provider that performs one or many functions, but does nothing until a request is made. For example, each WebSphere MQ for z/OS queue manager

or instance of a DB2 for z/OS database management system is a z/OS subsystem.

### **surrogate pair**

A coded representation for a single character that consists of a sequence of two 16-bit code units, in which the first value of the pair is a high-surrogate code unit in the range U+D800 through U+DBFF, and the second value is a low-surrogate code unit in the range U+DC00 through U+DFFF. Surrogate pairs provide an extension mechanism for encoding 917 476 characters without requiring the use of 32-bit characters.

### **SVC dump**

A dump that is issued when a z/OS or a DB2 functional recovery routine detects an error.

### **sync point**

See commit point.

### **syncpoint tree**

The tree of recovery managers and resource managers that are involved in a logical unit of work, starting with the recovery manager, that make the final commit decision.

### **synonym**

In SQL, an alternative name for a table or view. Synonyms can be used to refer only to objects at the subsystem in which the synonym is defined. A synonym cannot be qualified and can therefore not be used by other users. Contrast with alias.

### **Sysplex**

See Parallel Sysplex.

### **system administrator**

The person at a computer installation who designs, controls, and manages the use of the computer system.

### **system agent**

A work request that DB2 creates such as prefetch processing, deferred writes, and service tasks. See also allied agent.

### **system authorization ID**

The primary DB2 authorization ID that is used to establish a trusted connection. A system authorization ID is derived from the system user ID that is provided by an external entity, such as a middleware server.

**system conversation**

The conversation that two DB2 subsystems must establish to process system messages before any distributed processing can begin.

**system-defined routine**

In DB2 10 for z/OS and later, an object (function or procedure) for which system DBADM and SQLADM authorities have implicit execute privilege on the routine and any packages executed within the routine.

**System Modification Program/Extended (SMP/E)**

A z/OS tool for making software changes in programming systems (such as DB2) and for controlling those changes.

**system period**

A pair of columns with system-maintained values that indicates the period of time when a row is valid. See also system-period temporal table and system-period data versioning.

**system-period data versioning**

Automatic maintenance of historical data by the database manager by using a system period. See also system period and system-period temporal table.

**system-period temporal table**

A table that is defined with system-period data versioning. See also bitemporal table, system-period data versioning, and history table.

**Systems Network Architecture (SNA)**

The description of the logical structure, formats, protocols, and operational sequences for transmitting information through and controlling the configuration and operation of networks.

**table** A named data object consisting of a specific number of columns and some number of unordered rows. See also base table or temporary table. Contrast with auxiliary table, clone table, materialized query table, result table, and transition table.

**table-controlled partitioning**

A type of partitioning in which partition boundaries for a partitioned table are controlled by values that are defined in the CREATE TABLE statement.

**table function**

A function that receives a set of arguments and returns a table to the SQL statement that references the function. A table function can be referenced only in the FROM clause of a subselect.

**table locator**

A mechanism that allows access to trigger tables in SQL or from within user-defined functions. A table locator is a fullword integer value that represents a transition table.

**table space**

A page set that is used to store the records in one or more tables. See also partitioned table space, segmented table space, and universal table space.

**table space set**

A set of table spaces and partitions that should be recovered together for one of the following reasons:

- Each of them contains a table that is a parent or descendent of a table in one of the others.
- The set contains a base table and associated auxiliary tables.

A table space set can contain both types of relationships.

**task control block (TCB)**

A z/OS control block that is used to communicate information about tasks within an address space that is connected to a subsystem. See also address space connection.

**TB** Terabyte. A value of 1 099 511 627 776 bytes.

**TCB** See task control block.

**TCP/IP**

A network communication protocol that computer systems use to exchange information across telecommunication links.

**TCP/IP port**

A 2-byte value that identifies an end user or a TCP/IP network application within a TCP/IP host.

**template**

A DB2 utilities output data set descriptor that is used for dynamic allocation. A

template is defined by the TEMPLATE utility control statement.

| **temporal table**

| A table that records the period of time  
| when a row is valid.

**temporary table**

A table that holds temporary data. Temporary tables are useful for holding or sorting intermediate results from queries that contain a large number of rows. The two types of temporary table, which are created by different SQL statements, are the created temporary table and the declared temporary table. Contrast with result table. See also created temporary table and declared temporary table.

| **textual XML format**

| A representation of XML data that uses  
| character values, an approach that allows  
| for direct reading by people.

**thread** See DB2 thread.

**threadsafe**

A characteristic of code that allows multithreading both by providing private storage areas for each thread, and by properly serializing shared (global) storage areas.

**three-part name**

The full name of a table, view, or alias. It consists of a location name, a schema name, and an object name, separated by a period.

**time** A three-part value that designates a time of day in hours, minutes, and seconds.

**timeout**

Abnormal termination of either the DB2 subsystem or of an application because of the unavailability of resources. Installation specifications are set to determine both the amount of time DB2 is to wait for IRLM services after starting, and the amount of time IRLM is to wait if a resource that an application requests is unavailable. If either of these time specifications is exceeded, a timeout is declared.

**Time-Sharing Option (TSO)**

An option in z/OS that provides interactive time sharing from remote terminals.

**timestamp**

A seven-part value that consists of a date and time. The timestamp is expressed in years, months, days, hours, minutes, seconds, and microseconds.

| **timestamp with time zone**

| A two-part value that consists of a  
| timestamp and time zone. The timestamp  
| with time zone is expressed in years,  
| months, days, hours, minutes, seconds,  
| microseconds, time zone hours, and time  
| zone minutes.

**trace** A DB2 facility that provides the ability to monitor and collect DB2 monitoring, auditing, performance, accounting, statistics, and serviceability (global) data.

| **transaction**

| An atomic series of SQL statements that  
| make up a logical unit of work. All of the  
| data modifications made during a  
| transaction are either committed together  
| as a unit or rolled back as a unit.

**transaction lock**

A lock that is used to control concurrent execution of SQL statements.

**transaction program name**

In SNA LU 6.2 conversations, the name of the program at the remote logical unit that is to be the other half of the conversation.

| **transaction-start-ID column**

| A generated column that is defined with  
| the AS TRANSACTION START ID clause.  
| The value is assigned whenever a row is  
| inserted into the table or any column in  
| the row is updated. A transaction-start-ID  
| column is intended for use in a  
| system-period temporal table. See also  
| generated column, row-begin column, and  
| row-end column.

**transition table**

A temporary table that contains all the affected rows of the subject table in their state before or after the triggering event occurs. Triggered SQL statements in the trigger definition can reference the table of changed rows in the old state or the new state. Contrast with auxiliary table, base table, clone table, and materialized query table.

**transition variable**

A variable that contains a column value



of the affected row of the subject table in its state before or after the triggering event occurs. Triggered SQL statements in the trigger definition can reference the set of old values or the set of new values.

#### **tree structure**

A data structure that represents entities in nodes, with a most one parent node for each node, and with only one root node.

#### **trigger**

A database object that is associated with a single base table or view and that defines a rule. The rule consists of a set of SQL statements that run when an insert, update, or delete database operation occurs on the associated base table or view.

#### **trigger activation**

The process that occurs when the trigger event that is defined in a trigger definition is executed. Trigger activation consists of the evaluation of the triggered action condition and conditional execution of the triggered SQL statements.

#### **trigger activation time**

An indication in the trigger definition of whether the trigger should be activated before or after the triggered event.

#### **trigger body**

The set of SQL statements that is executed when a trigger is activated and its triggered action condition evaluates to true. A trigger body is also called triggered SQL statements.

#### **trigger cascading**

The process that occurs when the triggered action of a trigger causes the activation of another trigger.

#### **triggered action**

The SQL logic that is performed when a trigger is activated. The triggered action consists of an optional triggered action condition and a set of triggered SQL statements that are executed only if the condition evaluates to true.

#### **triggered action condition**

An optional part of the triggered action. This Boolean condition appears as a WHEN clause and specifies a condition that DB2 evaluates to determine if the triggered SQL statements should be executed.

#### **triggered SQL statements**

The set of SQL statements that is executed when a trigger is activated and its triggered action condition evaluates to true. Triggered SQL statements are also called the trigger body.

#### **trigger granularity**

In SQL, a characteristic of a trigger, which determines whether the trigger is activated:

- Only once for the triggering SQL statement
- Once for each row that the SQL statement modifies

#### **triggering event**

The specified operation in a trigger definition that causes the activation of that trigger. The triggering event is comprised of a triggering operation (insert, update, or delete) and a subject table or view on which the operation is performed.

#### **triggering SQL operation**

The SQL operation that causes a trigger to be activated when performed on the subject table or view.

#### **trigger package**

A package that is created when a CREATE TRIGGER statement is executed. The package is executed when the trigger is activated.

#### **trust attribute**

An attribute on which to establish trust. A trusted relationship is established based on one or more trust attributes.

#### **trusted connection**

A database connection whose attributes match the attributes of a unique trusted context defined at the DB2 database server.

#### **trusted connection reuse**

The ability to switch the current user ID on a trusted connection to a different user ID.

#### **trusted context**

A database security object that enables the establishment of a trusted relationship between a DB2 database management system and an external entity.

#### **trusted context default role**

A role associated with a trusted context.

| The privileges granted to the trusted context default role can be acquired only when a trusted connection based on the trusted context is established or reused.

| **trusted context user**  
| A user ID to which switching the current user ID on a trusted connection is permitted.

| **trusted context user-specific role**  
| A role that is associated with a specific trusted context user. It overrides the trusted context default role if the current user ID on the trusted connection matches the ID of the specific trusted context user.

| **trusted relationship**  
| A privileged relationship between two entities such as a middleware server and a database server. This relationship allows for a unique set of interactions between the two entities that would be impossible otherwise.

**TSO** See Time-Sharing Option.

**TSO attachment facility**  
A DB2 facility consisting of the DSN command processor and DB2I. Applications that are not written for the CICS or IMS environments can run under the TSO attachment facility.

**typed parameter marker**  
A parameter marker that is specified along with its target data type. It has the general form:  
CAST(? AS data-type)

**type 2 indexes**  
Indexes that are created on a release of DB2 after Version 7 or that are specified as type 2 indexes in Version 4 or later.

**UCS-2** Universal Character Set, coded in 2 octets, which means that characters are represented in 16-bits per character.

**UDF** See user-defined function.

**UDT** User-defined data type. In DB2 for z/OS, the term distinct type is used instead of user-defined data type. See distinct type.

**uncommitted read (UR)**  
The isolation level that allows an application to read uncommitted data. See also cursor stability, read stability, and repeatable read.

**underlying view**  
The view on which another view is directly or indirectly defined.

**undo** A state of a unit of recovery that indicates that the changes that the unit of recovery made to recoverable DB2 resources must be backed out.

**Unicode**  
A standard that parallels the ISO-10646 standard. Several implementations of the Unicode standard exist, all of which have the ability to represent a large percentage of the characters that are contained in the many scripts that are used throughout the world.

**union** An SQL operation that involves the UNION set operator, which combines the results of two SELECT statements. Unions are often used to merge lists of values that are obtained from two tables.

**unique constraint**  
An SQL rule that no two values in a primary key, or in the key of a unique index, can be the same.

**unique index**  
An index that ensures that no identical key values are stored in a column or a set of columns in a table.

**unit of recovery (UOR)**  
A recoverable sequence of operations within a single resource manager, such as an instance of DB2. Contrast with unit of work.

**unit of work (UOW)**  
A recoverable sequence of operations within an application process. At any time, an application process is a single unit of work, but the life of an application process can involve many units of work as a result of commit or rollback operations. In a multisite update operation, a single unit of work can include several *units of recovery*. Contrast with unit of recovery.

**universal table space**  
A table space that is both segmented and partitioned. Contrast with partitioned table space, segmented table space, partition-by-growth table space, and range-partitioned table space.

<b>unlock</b>	The act of releasing an object or system resource that was previously locked and returning it to general availability within DB2.	
<b>untyped parameter marker</b>	A parameter marker that is specified without its target data type. It has the form of a single question mark (?).	
<b>updatability</b>	The ability of a cursor to perform positioned updates and deletes. The updatability of a cursor can be influenced by the SELECT statement and the cursor sensitivity option that is specified on the DECLARE CURSOR statement.	
<b>update hole</b>	The location on which a cursor is positioned when a row in a result table is fetched again and the new values no longer satisfy the search condition. See also delete hole.	
<b>update trigger</b>	A trigger that is defined with the triggering SQL operation update.	
<b>UR</b>	See uncommitted read.	
<b>user-defined data type (UDT)</b>	See distinct type.	
<b>user-defined function (UDF)</b>	A function that is defined to DB2 by using the CREATE FUNCTION statement and that can be referenced thereafter in SQL statements. A user-defined function can be an external function, a sourced function, or an SQL function. Contrast with built-in function.	
<b>user view</b>	In logical data modeling, a model or representation of critical information that the business requires.	
<b>UTF-16</b>	Unicode Transformation Format, 16-bit encoding form, which is designed to provide code values for over a million characters and a superset of UCS-2. The CCSID value for data in UTF-16 format is 1200. DB2 for z/OS supports UTF-16 in graphic data fields.	
<b>UTF-8</b>	Unicode Transformation Format, 8-bit encoding form, which is designed for ease of use with existing ASCII-based systems.	
		The CCSID value for data in UTF-8 format is 1208. DB2 for z/OS supports UTF-8 in mixed data fields.
<b>value</b>	The smallest unit of data that is manipulated in SQL.	
<b>variable</b>	A data element that specifies a value that can be changed. A COBOL elementary data item is an example of a host variable. Contrast with constant.	
<b>variant function</b>	See nondeterministic function.	
<b>varying-length string</b>	A character, graphic, or binary string whose length varies within set limits. Contrast with fixed-length string.	
<b>version</b>	A member of a set of similar programs, DBRMs, packages, or LOBs.	
	<ul style="list-style-type: none"> <li>• <b>A version of a program</b> is the source code that is produced by precompiling the program. The program version is identified by the program name and a timestamp (consistency token).</li> <li>• <b>A version of an SQL procedural language routine</b> is produced by issuing the CREATE or ALTER PROCEDURE statement for a native SQL procedure.</li> <li>• <b>A version of a DBRM</b> is the DBRM that is produced by precompiling a program. The DBRM version is identified by the same program name and timestamp as a corresponding program version.</li> <li>• <b>A version of an application package</b> is the result of binding a DBRM within a particular database system. The application package version is identified by the same program name and consistency token as the DBRM.</li> <li>• <b>A version of a LOB</b> is a copy of a LOB value at a point in time. The version number for a LOB is stored in the auxiliary index entry for the LOB.</li> <li>• <b>A version of a record</b> is a copy of the record at a point in time.</li> </ul>	
<b>view</b>	A logical table that consists of data that is generated by a query. A view can be based on one or more underlying base tables or views, and the data in a view is	

| determined by a SELECT statement that is  
| run on the underlying base tables or  
| views.

**Virtual Storage Access Method (VSAM)**

| An access method for direct or sequential  
| processing of fixed- and varying-length  
| records on disk devices.

**Virtual Telecommunications Access Method (VTAM)**

    An IBM licensed program that controls communication and the flow of data in an SNA network (in z/OS).

**volatile table**

    A table for which SQL operations choose index access whenever possible.

**VSAM**

    See Virtual Storage Access Method.

**VTAM**

    See Virtual Telecommunications Access Method.

**warm start**

    The normal DB2 restart process, which involves reading and processing log records so that data that is under the control of DB2 is consistent. Contrast with cold start.

**WLM application environment**

    A z/OS Workload Manager attribute that is associated with one or more stored procedures. The WLM application environment determines the address space in which a given DB2 stored procedure runs.

| **WLM enclave**

| A construct that can span multiple  
| dispatchable units (service request blocks  
| and tasks) in multiple address spaces,  
| allowing them to be reported on and  
| managed by WLM as part of a single  
| work request.

**write to operator (WTO)**

    An optional user-coded service that allows a message to be written to the system console operator informing the operator of errors and unusual system conditions that might need to be corrected (in z/OS).

**WTO** See write to operator.

**WTOR**

    Write to operator (WTO) with reply.

**XCF** See cross-system coupling facility.

**XES** See cross-system extended services.

**XML** See Extensible Markup Language.

**XML attribute**

    A name-value pair within a tagged XML element that modifies certain features of the element.

| **XML column**

| A column of a table that stores XML  
| values and is defined using the data type  
| XML. The XML values that are stored in  
| XML columns are internal representations  
| of well-formed XML documents.

| **XML data type**

| A data type for XML values.

**XML element**

    A logical structure in an XML document that is delimited by a start and an end tag. Anything between the start tag and the end tag is the content of the element.

| **XML index**

| An index on an XML column that  
| provides efficient access to nodes within  
| an XML document by providing index  
| keys that are based on XML patterns.

| **XML lock**

| A column-level lock for XML data. The  
| operation of XML locks is similar to the  
| operation of LOB locks.

**XML node**

    The smallest unit of valid, complete structure in a document. For example, a node can represent an element, an attribute, or a text string.

| **XML node ID index**

| An implicitly created index, on an XML  
| table that provides efficient access to XML  
| documents and navigation among  
| multiple XML data rows in the same  
| document.

| **XML pattern**

| A slash-separated list of element names,  
| an optional attribute name (at the end), or  
| kind tests, that describe a path within an  
| XML document in an XML column. The  
| pattern is a restrictive form of path  
| expressions, and it selects nodes that  
| match the specifications. XML patterns are  
| specified to create indexes on XML  
| columns in a database.

**XML publishing function**

A function that returns an XML value from SQL values. An XML publishing function is also known as an XML constructor.

**XML schema**

In XML, a mechanism for describing and constraining the content of XML files by indicating which elements are allowed and in which combinations. XML schemas are an alternative to document type definitions (DTDs) and can be used to extend functionality in the areas of data typing, inheritance, and presentation.

**XML schema repository (XSR)**

A repository that allows the DB2 database system to store XML schemas. When registered with the XSR, these objects have a unique identifier and can be used to validate XML instance documents.

**XML serialization function**

A function that returns a serialized XML string from an XML value.

**XML table**

An auxiliary table that is implicitly created when an XML column is added to a base table. This table stores the XML data, and the column in the base table points to it.

**XML table space**

A table space that is implicitly created when an XML column is added to a base table. The table space stores the XML table. If the base table is partitioned, one partitioned table space exists for each XML column of data.

**X/Open**

An independent, worldwide open systems organization that is supported by most of the world's largest information systems suppliers, user organizations, and software companies. X/Open's goal is to increase the portability of applications by combining existing and emerging standards.

**XRF** See Extended Recovery Facility.

**XSR** See XML schema repository.

**zIIP** See IBM System z9 Integrated Processor.

**z/OS** An operating system for the System z product line that supports 64-bit real and virtual storage.

**z/OS Distributed Computing Environment (z/OS DCE)** A set of technologies that are provided by the Open Software Foundation to implement distributed computing.





---

## Index

### Special characters

- , (comma) in DB2 commands 16
- : (colon) in DB2 commands 16
- () (parentheses) in DB2 commands 16
- \* (asterisk)
  - DISPLAY THREAD command 261
  - FREE PACKAGE command 363
  - FREE QUERY command 368
  - in DB2 commands 16
  - REBIND PACKAGE command 129
- \*,\* (asterisk)
  - DISPLAY PROCEDURE command 250
  - START PROCEDURE command 490
  - STOP FUNCTION SPECIFIC command 536
  - STOP PROCEDURE command 542
- = (equal sign) in DB2 commands 16
- ' (apostrophe) in DB2 commands 16
- " (quotation mark) in DB2 commands 16

## A

- ABEND subcommand of DSN 311
- accelerators, displaying 165
- ACCESS DATABASE
  - command 49
- ACCESS option
  - START DATABASE command 469
  - START DB2 command 474
- accessibility
  - keyboard x
  - shortcut keys x
- accounting
  - trace
    - displaying 293
    - starting 497
    - stopping 549
- ACCTG option
  - DISPLAY TRACE command 297
  - MODIFY TRACE command 402
  - STOP TRACE command 552
- ACQUIRE option
  - BIND PLAN subcommand 91
  - DSNH command 348
  - REBIND PLAN subcommand 91
- ACTION option
  - BIND PACKAGE subcommand 92
  - BIND PLAN subcommand 92
  - DCLGEN subcommand 153
  - DSNH command 348
  - RECOVER INDOUBT command 424
  - STOP FUNCTION SPECIFIC command 536
  - STOP PROCEDURE command 542
- ACTIVE option
  - DISPLAY BUFFERPOOL command 174
  - DISPLAY DATABASE command 195
- ADD option
  - DCLGEN subcommand 153
  - DSNH command 351
- administrative task scheduler
  - commands
    - MODIFY admtproc, APPL=SHUTDOWN option 371
    - MODIFY admtproc, APPL=TRACE option 373
    - START admtproc 461
    - STOP admtproc 517
  - starting
    - an administrative task scheduler 461
  - stopping
    - an administrative task scheduler 517
- Administrative task scheduler commands 12
- ADVISORY option of DISPLAY DATABASE command 197
- AFTER option of DISPLAY DATABASE command 195
- ALL keyword of MODIFY irlmproc,DIAG command 381
- ALLD option of MODIFY irlmproc,STATUS command of z/OS 389
- ALLI option of MODIFY irlmproc,STATUS command of z/OS 390
- ALTER BUFFERPOOL command
  - description 53
  - example 57
  - Option descriptions 54
- ALTER GROUPBUFFERPOOL command
  - description 59
  - example 62
  - Option descriptions 59
- ALTER UTILITY command
  - description 63
  - example 65
- ambiguous cursor 100
- APCOMPARE option
  - BIND PACKAGE subcommand 93
  - REBIND PACKAGE subcommand 93
  - REBIND TRIGGER PACKAGE subcommand 93
- APOST option
  - DCLGEN subcommand 155
  - DSNH command 340
- APOSTSQL option of DSNH command 346
- application plan
  - binding 81
  - deleting 365
  - maximum size 5
  - rebinding, changing plans 411
- application program
  - START command 463
  - testing 311
- application program, preparing for DSNH CLIST processing 329
- APPNAME option
  - DISPLAY TRACE command 296
  - START TRACE command 504
  - STOP TRACE command 552
- APRETAINDUP option
  - REBIND PACKAGE subcommand 94
  - REBIND TRIGGER PACKAGE subcommand 94
- APREUSE option
  - BIND PACKAGE subcommand 95
  - REBIND PACKAGE subcommand 95
  - REBIND TRIGGER PACKAGE subcommand 95
- ARCHIVE LOG command
  - description 67

- ARCHIVE LOG command (*continued*)
  - option descriptions 68
- ASMLIB option of DSNH command 335
- ASMLOAD option of DSNH command 335
- asterisk (\*)
  - DISPLAY PROCEDURE command 250
  - START PROCEDURE command 490
  - STOP FUNCTION SPECIFIC command 536
  - STOP PROCEDURE command 542
- asterisk (\*)
  - DISPLAY THREAD command 261
  - FREE PACKAGE command 363
  - FREE QUERY command 368
  - in DB2 commands 16
  - REBIND PACKAGE command 129
- ASUSER option
  - DSN command 312
- AT option of DCLGEN subcommand 153
- AT(COMMIT) option of STOP DATABASE command 524
- AUDIT option
  - DISPLAY TRACE command 297
  - MODIFY TRACE command 402
  - STOP TRACE command 552
- audit trace
  - displaying 293
  - starting 497
  - stopping 549
- AUDTPLCY option
  - START TRACE command 300
  - STOP TRACE command 555
- AUDTPLCY option of START TRACE command 508
- AUTHID option
  - DISPLAY TRACE command 296
  - START TRACE command 502
  - STOP TRACE command 552
- authorization ID
  - secondary privileges 3
  - SQL, privileges exercised by 3
  - trusted context 3
- AUTOREC option of ALTER GROUPBUFFERPOOL command 60

## B

- BDBRMLIB option of DSNH command 349
- BDMEM option of DSNH command 348
- BIND PACKAGE subcommand of DSN
  - description 73
  - example 79
  - option descriptions 91
- BIND PLAN subcommand of DSN
  - description 81
  - example 84
  - option descriptions 91
- BIND QUERY command
  - option descriptions 88
- BIND QUERY subcommand of DSN
  - description 87
  - example 89
- binding
  - DSNH processing 329
  - initiating 73, 81
  - options for 91
- blank characters in DB2 command 16
- BLIB option of DSNH command 348
- BMEM option of DSNH command 348
- BnLIB option of DSNH command 348

- BOTH option
  - SET LOG command 446
- BSDS (bootstrap data set), recovery 421
- buffer pool
  - active and inactive 53, 173
  - altering attributes 53
  - displaying current status 173
  - parallel sequential steal threshold (VPSEQT) 54
- BUFSIZE option of START TRACE command 508

## C

- C option of DCLGEN subcommand 154
- CACHESIZE option
  - BIND PLAN subcommand 96
  - DSNH command 349
  - REBIND PLAN subcommand 96
- CANCEL OFFLOAD option of ARCHIVE LOG command 69
- CANCEL option of RECOVER POSTPONED command 427
- CANCEL THREAD command
  - description 143
  - example 148
- canceling threads, description 143
- CASTOUT option of STOP DB2 command 530
- CATENFM utility, effects of TERM command 560
- CATMAINT utility, effects of TERM command 561
- CCLINK option of DSNH command 335
- CCLLIB option of DSNH command 335
- CCLOAD option of DSNH command 335
- CCMSGs option of DSNH command 335
- CCOLIB option of DSNH command 335
- CCPLIB option of DSNH command 335
- CCPMMSGs option of DSNH command 335
- CCSID option of DSNH command 335
- CCSLIB option of DSNH command 335
- CD-ROM, books on 577
- CHANGE command of IMS
  - description 149
  - example 150
- CHECK DATA utility, effects of TERM command 561
- CHECK INDEX utility, effects of TERM command 561
- CHECK LOB utility, effects of TERM command 561
- CHKTIME option
  - SET LOG command 446
- CICS
  - commands
    - DSNC 315
    - DSNC DISCONNECT 317
    - DSNC DISPLAY 319
    - DSNC MODIFY 323
    - DSNC STOP 325
    - DSNC STRT 327
  - option of BIND and REBIND subcommands 112
  - option of DSNH command 349
  - translation step in DSNH processing 329
- CICS commands 11
- CICSCOB option of DSNH command 336
- CICSLLIB option of DSNH command 336
- CICSOPT option of DSNH command 336
- CICSPLIB option of DSNH command 336
- CICSPRE option of DSNH command 336
- CICSVER option of DSNH command 336
- CICSXLAT option of DSNH command 336
- CLAIMERS option of DISPLAY DATABASE command 193
- CLASS option
  - DISPLAY TRACE command 298
  - IFCIDs activated by trace class 504



CLASS option (*continued*)  
     MODIFY TRACE command 402  
     START TRACE command 504  
     STOP TRACE command 553  
 CLASST option of ALTER GROUPBUFFERPOOL command 61  
 CLIB option of DSNH command 336  
 CLONE option  
     START DATABASE command 468  
     STOP DATABASE command 524  
 CnLIB option of DSNH command 336  
 COB2CICS option of DSNH command 337  
 COB2LIB option of DSNH command 337  
 COB2LOAD option of DSNH command 337  
 COBICOMP option of DSNH command 336  
 COBILINK option of DSNH command 336  
 COBIPLNK option of DSNH command 336  
 COBIPMSG option of DSNH command 337  
 COBLIB option of DSNH command 337  
 COBLOAD option of DSNH command 337  
 COBSOM option of DSNH command 337  
 code, return 313  
 collection, package  
     BIND PACKAGE subcommand 128  
     REBIND PACKAGE subcommand 128  
     REBIND TRIGGER PACKAGE subcommand 419  
 COLLID option  
     REBIND PLAN subcommand 97  
 COLSUFFIX option of DCLGEN subcommand 155  
 column name, as a field name 155  
 comma option of DSNH command 339  
 command continuation character 19  
 command output 23  
 command prefix  
     description 15  
     multiple subsystems 475  
     part of a command 15  
 command recognition character (CRC) 15  
 commands 375  
     ACCESS DATABASE 49  
 commands, scope 21  
 comment  
     DCLGEN subcommand output 156  
     DSN subcommands 311  
 COMMENT option  
     DISPLAY TRACE command 297  
     MODIFY TRACE command 403  
     START TRACE command 500  
     STOP TRACE command 553  
 commit point, terminating utility 559  
 COMP option of TRACE CT command 566  
 COMPILE option of DSNH command 337  
 COMPLETE XMLSCHEMA command  
     example 38, 39  
 CONCURRENTACCESSRESOLUTION option  
     BIND PACKAGE command 97  
     BIND PLAN command 97  
     REBIND PACKAGE command 97  
     REBIND PLAN command 97  
     REBIND TRIGGER PACKAGE command 97  
 conditional restart  
     control record, effect on restart 475  
 CONNECT option of DSN command 337  
 connection  
     DB2  
         GROUP option of DSN command 312  
         RETRY option of DSN command 312  
     connection (*continued*)  
         displaying  
             connection information 259  
             group buffer pool 234  
             IRLM subsystem status 391  
             status 161  
         DSNC DISPLAY command 319  
         terminating 519  
     CONNID option  
         DISPLAY TRACE command 296  
         START TRACE command 504  
         STOP TRACE command 552  
     CONNLIST option of DISPLAY GROUPBUFFERPOOL command 229  
     CONTROL option of DSNH command 337  
     COPTION option of DSNH command 337  
     COPY option  
         BIND PACKAGE subcommand 98  
         DSNH command 351  
         SET LOG command 447  
     COPY utility, effects of TERM command 561  
     COPYVER option  
         BIND PACKAGE subcommand 98, 104  
         DSNH command 351  
     correlation ID  
         recovering threads 424  
     CORRELATION option of START TRACE command 508  
     CORRID option  
         DISPLAY TRACE command 296  
         START TRACE command 504  
         STOP TRACE command 552  
     COUNT option of SET ARCHIVE command 442  
     CP option of RUN subcommand 437  
     CPP option of DCLGEN subcommand 154  
     CPPCLASS option of DSNH command 338  
     CPPCLLIB option of DSNH command 338  
     CPPCSLIB option of DSNH command 338  
     CPPLINK option of DSNH command 338  
     CPPLLIB option of DSNH command 338  
     CPPPMMSG option of DSNH command 338  
     CPPSLIB option of DSNH command 338  
     CPPUTIL option of DSNH command 338  
     CPU option of START TRACE command 509  
     CRC (command recognition character), description 15  
     CURRENTDATA option  
         BIND PACKAGE subcommand 100  
         BIND PLAN subcommand 100  
         DSNH command 349, 351  
         REBIND PACKAGE subcommand 100  
         REBIND PLAN subcommand 100  
         REBIND TRIGGER PACKAGE subcommand 100  
     CURRENTSERVER option  
         BIND PLAN subcommand 101  
         DSNH command 349  
         REBIND PLAN subcommand 101  
     cursor, ambiguous 100  
     CYLINDER option of DSNH command 346  
  
**D**  
 data sharing  
     delays, diagnosing 381  
     displaying  
         archive log information 171  
         information about groups 219  
         status of members 219  
     identifying members with utility jobs 306

- data sharing (*continued*)
  - scope of commands 21
  - starting members 476
- database
  - displaying status 189
  - reserved names 525
  - starting 465
  - stopping 521
- DATE
  - option of DSNH command 338
- DB2 books online 577
- DB2 commands 375
  - command names 15
  - commands
    - ACCESS DATABASE 49
    - ALTER BUFFERPOOL 53
    - ALTER GROUPBUFFERPOOL 59
    - ALTER UTILITY 63
    - ARCHIVE LOG 67
    - CANCEL THREAD 143
    - DISPLAY ACCEL 165
    - DISPLAY ARCHIVE 171
    - DISPLAY BUFFERPOOL 173
    - DISPLAY DATABASE 189
    - DISPLAY DDF 209
    - DISPLAY GROUP 219
    - DISPLAY GROUPBUFFERPOOL 227
    - DISPLAY LOCATION 241
    - DISPLAY LOG 247
    - DISPLAY PROCEDURE 249
    - DISPLAY PROFILE 255
    - DISPLAY RLIMIT 257
    - DISPLAY THREAD 259
    - DISPLAY TRACE 293
    - DISPLAY UTILITY 305
    - MODIFY TRACE 401
    - RECOVER BSDS 421
    - RECOVER INDOUBT 423, 426
    - RECOVER POSTPONED 427
    - REFRESH DB2,EARLY 429
    - RESET INDOUBT 433
    - SET SYSPARM 451
    - START ACCEL 457
    - START DATABASE 465
    - START DB2 473
    - START DDF 477
    - START FUNCTION SPECIFIC 479
    - START PROCEDURE 489
    - START PROFILE 493
    - START RLIMIT 495
    - START TRACE 497
    - STOP ACCEL 515
    - STOP DATABASE 521
    - STOP DB2 529
    - STOP DDF 531
    - STOP FUNCTION SPECIFIC 535
    - STOP PROCEDURE 541
    - STOP PROFILE 545
    - STOP RLIMIT 547
    - STOP TRACE 549
    - TERM UTILITY 559
  - completion messages 10
  - description of 15
  - DISPLAY FUNCTION SPECIFIC 213
  - entering from supported environments 10
  - scope 21
  - separator 15
- DB2 Information Center for z/OS solutions 577
- DB2 precompiler 15
- DBM1 option of START DB2 command 474
- DBPROTOCOL option
  - BIND PACKAGE subcommand 102
  - BIND PLAN subcommand 102
  - DSNH command 349
  - REBIND PACKAGE subcommand 102
  - REBIND PLAN subcommand 102
- DBRM (database request module)
  - BIND PLAN subcommand 125
- DBRMLIB option of DSNH command 339
- DCLGEN subcommand of DSN 156
  - declaring an indicator variable array 156
  - description 151
  - example 158
  - forming field names 155
  - option descriptions 152
- DDF 375
- DDF (distributed data facility), displaying 209
- DDF (distributed data facility), modifying 375
- DEADLINE option of ALTER UTILITY command 64
- DEADLOCK option of START irlmproc command 483
- DECARTH option of DSNH command 339
- DECIMAL
  - option of DSNH command 339
- DECOMPOSE XML DOCUMENT command
  - example 41
- DECP option 474
- DEFAULT option of SET ARCHIVE command 442
- DEFER
  - option of BIND PLAN subcommand 103
- DEFER option
  - BIND PACKAGE subcommand 102
  - BIND PLAN subcommand 102
  - DSNH command 349
  - REBIND PACKAGE subcommand 102
  - REBIND PLAN subcommand 102
- degree of parallel processing 104
- DEGREE option
  - BIND PACKAGE subcommand 104
  - BIND PLAN subcommand 104
  - DSNH command 349
  - REBIND PACKAGE subcommand 104
  - REBIND PLAN subcommand 104
- DELAY
  - keyword of MODIFY irlmproc,DIAG command 381
- DELAY option of ALTER UTILITY command 65
- deleting, IMS units of recovery 149
- DELIMIT option of DSNH command 340
- DEPLOY option
  - BIND PACKAGE subcommand 104
- deregistering IRLM 380
- description 375
- DEST option
  - DISPLAY TRACE command 297
  - START TRACE command 500
  - STOP TRACE command 553
- DESTINATION option of DSNH MODIFY command 323
- DETAIL option
  - DISPLAY BUFFERPOOL command 174
  - DISPLAY GROUP command 219
  - DISPLAY LOCATION command 242
  - DISPLAY THREAD command 264
  - DISPLAY TRACE command 297
- DIAG keyword of MODIFY irlmproc,DIAG command 381
- DIAGNOSE utility, TERM command effects 561

- diagnostic dumps, IRLM 381
- disability x
- DISABLE option
  - BIND PACKAGE subcommand 111
  - BIND PLAN subcommand 111
  - DSNH command 349
  - REBIND PACKAGE subcommand 111
  - REBIND PLAN subcommand 111
- disabling a function permanently 537
- DISCONNECT option
  - BIND PLAN subcommand 106
  - DSNH command 349
  - REBIND PLAN subcommand 106
- DISPLAY ACCEL command
  - description 165
  - example 166
  - Option descriptions 166
- DISPLAY ARCHIVE command 171
- DISPLAY BUFFERPOOL command
  - description 173
  - Option descriptions 174
  - Output 176
- DISPLAY command of IMS
  - description 161
  - example 163
  - Option descriptions 161
- DISPLAY DATABASE command
  - description 189
  - example 201
  - Option descriptions 192
- DISPLAY DDF command
  - description 209
  - example 210
  - Option descriptions 209
  - Output 209
- DISPLAY FUNCTION SPECIFIC command
  - description 213
  - Examples 216
  - Output 215
- DISPLAY GROUP command
  - description 219
  - Examples 220
- DISPLAY GROUPBUFFERPOOL command
  - description 227
  - option descriptions 228
  - output 230
  - summary report example 234
- DISPLAY LOCATION command
  - description 241
  - example 242
  - option descriptions 242
- DISPLAY LOG command
  - description 247
  - example 247
- DISPLAY NET command of VTAM 145
- DISPLAY PROCEDURE command
  - description 249
  - example 252
  - Option descriptions 250
  - Output 251
- DISPLAY PROFILE command
  - example 255
- DISPLAY RLIMIT command 257
- DISPLAY THREAD command
  - description 259
  - example 266
  - Option descriptions 261
- DISPLAY THREAD command *(continued)*
  - Output 266
- DISPLAY TRACE command
  - description 293
  - example 302
  - Option descriptions 296
- DISPLAY UTILITY command
  - description 305
  - example 308
  - option descriptions 305
  - Output 306
- displaying
  - information about
    - accelerators 165
    - archive logs 171
    - communications database and resource limit facility 197
    - data sharing group 219
    - data-partitioned secondary indexes 198
    - DB2 functions 213
    - DB2 threads 259
    - DDF 209
    - logical partitions 198
    - logs 247
    - resource limit facility (governor) 257
    - restricted objects 197
    - stored procedures 249
    - threads with remote locations 241
    - trace activity 293
  - status of
    - buffer pools 173
    - DB2 databases 189
    - DB2 utilities 305
    - group buffer pools 227
- DIST option of START DB2 command 475
- DISTRIBUTED option of START TRACE command 509
- DLIBATCH option
  - BIND and REBIND subcommands 112
  - DSNH command 349
- DSN command
  - ASUSER option 312
  - GROUP option 312
  - TEST option 312
- DSN command of TSO
  - abbreviations 19
  - description 311
  - example 314
  - Option descriptions 312
  - parsing subcommands 19
  - return codes 313
  - subcommands
    - ABEND 311
    - BIND PACKAGE 73
    - BIND PLAN 81
    - BIND QUERY 87
    - DCLGEN 151
    - END 359
    - FREE PACKAGE 361
    - FREE PLAN 365
    - FREE QUERY 367
    - REBIND PACKAGE 405
    - REBIND PLAN 411
    - REBIND TRIGGER PACKAGE 417
    - RUN 437
    - SPUFI 453
- DSN commands 9
- DSN subcommands 9

- dsnb461i 185
- dsnb462i 185
- DSNC command of CICS 315
- DSNC DISCONNECT command of CICS
  - description 317
  - Example 318
- DSNC DISPLAY command of CICS
  - description 319
  - example 322
  - Option descriptions 319
  - Output 320
- DSNC MODIFY command of CICS
  - description 323
  - example 324
  - Option descriptions 323
- DSNC STOP command of CICS
  - description 325
  - Example 325
  - Option descriptions 325
- DSNC STRT command of CICS
  - description 327
  - example 327
  - option descriptions 327
- DSNDB01 database, authority needed to start 466
- DSNDB06 database, authority needed to start 466
- DSNH command of TSO
  - data set names 334
  - description 329
  - example 354
  - option descriptions 331
- DSNZPARM
  - option of START DB2 command 474
- DUMP option
  - CANCEL THREAD command 144
  - MODIFY irlmproc,ABEND command 379
- dump, IRLM diagnostic 381
- DWQT option of ALTER BUFFERPOOL command 55
- DYNAMICRULES option
  - BIND PACKAGE subcommand 106
  - DSNH command 349
  - REBIND PACKAGE subcommand 106

**E**

- EARLY
  - option of REFRESH DB2 command 430
- EARLY option of START DB2 command 430
- ENABLE option
  - BIND PACKAGE subcommand 111
  - BIND PLAN subcommand 111
  - DSNH command 349
  - REBIND PACKAGE subcommand 111
  - REBIND PLAN subcommand 111
- ENCODING option
  - BIND PACKAGE subcommand 113
  - BIND PLAN subcommand 113
  - REBIND PACKAGE subcommand 113
  - REBIND PLAN subcommand 113
- END subcommand of DSN
  - description 359
  - Example 359
- ENTRY option of DSNH command 340
- escape character
  - APOST option of DCLGEN subcommand 155
  - QUOTE option of DCLGEN subcommand 155
- establishing connections between IMS and a subsystem 463

- EXPLAIN option
  - BIND PACKAGE subcommand 115
  - BIND PLAN subcommand 115
  - DSNH command 349, 352
  - ONLY option 115
  - REBIND PACKAGE subcommand 115
  - REBIND PLAN subcommand 115
  - REBIND TRIGGER PACKAGE subcommand 115
- extended MCS consoles, DB2 support of 10
- EXTENDEDINDICATOR option
  - BIND PACKAGE subcommand 117
  - REBIND PACKAGE subcommand 117

## F

- FLAG option
  - BIND PACKAGE subcommand 117
  - BIND PLAN subcommand 117
  - DSNH command 340, 349
  - FREE PACKAGE subcommand 363
  - FREE PLAN subcommand 366
  - REBIND PACKAGE subcommand 117
  - REBIND PLAN subcommand 117
  - REBIND TRIGGER PACKAGE subcommand 117
- FORCE option
  - DSNC STOP command 325
  - RESET INDOUBT command 434
  - START DATABASE command 469
  - STOP DB2 command 529
  - STOP DDF command 531
- FORTLIB option of DSNH command 340
- FORTLOAD option of DSNH command 340
- FREE PACKAGE subcommand of DSN
  - description 361
  - example 363
  - option descriptions 362
- FREE PLAN subcommand of DSN
  - description 365
  - Example 366
  - Option descriptions 365
- FREE QUERY subcommand of DSN
  - description 367
  - option descriptions 367
- functions, displaying information about 213

## G

- GBPCACHE option of ALTER GROUPBUFFERPOOL command 60
- GBPCHKPT option of ALTER GROUPBUFFERPOOL command 61
- GBPOOLT option of ALTER GROUPBUFFERPOOL command 61
- GDETAIL option of DISPLAY GROUPBUFFERPOOL command 229
- general-use programming information, described 585
- GENERIC option
  - BIND PACKAGE subcommand 118
  - REBIND PACKAGE subcommand 118
- GLOBAL option of START irlmproc command 486
- group buffer pool RECOVER-pending (GRECP)
  - status, removing using START DATABASE command 470
- group detail report of DISPLAY GROUPBUFFERPOOL command 230
- GROUP option
  - DSN command 312

- group, scope of command 21
- GTF option
  - DISPLAY TRACE command 297
  - START TRACE command 501
  - STOP TRACE command 553

## H

- HOST option of DSNH command 340

## I

- I/O processing, parallel, DEGREE option of bind subcommands 104
- IBMCOB option of DCLGEN subcommand 154
- ID option
  - RECOVER INDOUBT command 424
  - START RLIMIT command 495
- IFCID (instrumentation facility component identifier), identifiers by trace class 504
- IFCID option
  - MODIFY TRACE command 402
  - START TRACE command 507
- IFI
  - issuing commands 25
- IMMEDWRITE option
  - BIND PACKAGE subcommand 119
  - BIND PLAN subcommand 119
  - REBIND PACKAGE subcommand 119
  - REBIND PLAN subcommand 119
- IMS
  - commands
    - CHANGE 149
    - DISPLAY 161
    - SSR 455
    - START 463
    - STOP 519
    - TRACE 563
  - facilities, events tracing 563
- IMS commands 11
- IMSBMP option
  - BIND and REBIND subcommands 112
  - DSNH command 349
- IMSMPP option
  - BIND and REBIND subcommands 112
  - DSNH command 350
- IMSPRE option of DSNH command 341
- INACTIVE option
  - DISPLAY THREAD command 262
- INCLUDE statement of DCLGEN subcommand output 156
- indicator variable, array declaration in DCLGEN 156
- INDOUBT option
  - DISPLAY THREAD command 262
- indoubt thread, recovering 423
- INDVAR option of DCLGEN subcommand 156
- information about 375
- INPUT option of DSNH command 341
- INTERVAL option of DISPLAY BUFFERPOOL command 175
- invalidated packages 116
- IPADDR option
  - DISPLAY LOCATION command 242
  - RESET INDOUBT command 435
- IRLM (internal resource lock manager)
  - commands
    - MODIFY irlmproc,ABEND 379
    - MODIFY irlmproc,DIAG 381

- IRLM (internal resource lock manager) *(continued)*
  - commands *(continued)*

- MODIFY irlmproc,PURGE option 383
  - MODIFY irlmproc,SET option 385
  - MODIFY irlmproc,STATUS option 389
  - START irlmproc 483
  - STOP irlmproc 539
  - TRACE CT 565
- CSA
  - setting maximum amount of 385
- delays, diagnosing 381
- deregistering 379
- diagnostic dumps 381
- locks, releasing 383
- modifying, diagnostic trace 565
- restarting
  - effect on CSA value 387
- starting
  - an IRLM component 483
  - diagnostic trace 565
- status
  - checking 385
- status, checking 389
- stopping
  - diagnostic trace 565
  - normal 539
- terminating
  - abnormal 379
  - normal 539
- trace buffers, setting number of 385
- IRLM commands 12
- IRLMGRP option of START irlmproc command 484
- IRLMID option of START irlmproc command 484
- IRLMNM option of START irlmproc command 484
- ISOLATION
  - option of BIND PLAN subcommand
    - description 121
- ISOLATION option
  - BIND PACKAGE subcommand 121
  - DSNH command 350
  - REBIND PACKAGE subcommand 121
  - REBIND PLAN subcommand 121
  - REBIND TRIGGER PACKAGE subcommand 121

## K

- KEEPDYNAMIC option
  - BIND PACKAGE subcommand 122
  - BIND PLAN subcommand 122
  - DSNH command 350
  - REBIND PACKAGE subcommand 122
  - REBIND PLAN subcommand 122
- keywords 18

## L

- LABEL option of DCLGEN subcommand 155
- LANGUAGE option of DCLGEN subcommand 154
- LEAVE option of DSNH command 347
- library 577
- LIBRARY option
  - BIND PACKAGE subcommand 123
  - BIND PLAN subcommand 123
  - DCLGEN subcommand 153
  - RUN subcommand 438
- LIGHT option, START DB2 command 474



- light restart, with ARM 475
- LIMIT option
  - DISPLAY THREAD command 264
- LIMIT option of DISPLAY DATABASE command 195
- LINECOUNT option of DSNH command 341
- LINK option of DSNH command 341
- link-editing, processing 329
- LIST option of DISPLAY BUFFERPOOL command 175
- LLIB option of DSNH command 341
- LnLIB option of DSNH command 341
- LOAD option
  - DSNH command 341
  - SET SYSPARM command 452
- LOAD utility, effects of TERM command 561
- LOCAL option
  - CANCEL THREAD command 143
- LOCAL option of START irlmproc command 486
- location name
  - BIND PACKAGE subcommand 112, 128
  - DISPLAY LOCATION command 242
  - REBIND PACKAGE subcommand 112, 128
  - REBIND TRIGGER PACKAGE subcommand 419
- LOCATION option
  - DISPLAY THREAD command 263
  - DISPLAY TRACE command 296
  - RESET INDOUBT command 434
  - START TRACE command 503
- location statistics 497
- LOCKS option of DISPLAY DATABASE command 194
- LOCKTABL option of START irlmproc command 484
- LOG option of TRACE command 563
- logical page list (LPL) 470
- logical partitions, displaying 198
- LOGLOAD option
  - SET LOG command 446
- LONGLOG option of ALTER UTILITY option 65
- LOOKUP option
  - BIND QUERY subcommand 124
- LOPTION option of DSNH command 341
- LPL (logical page list)
  - option of DISPLAY DATABASE command 194
  - recovering pages
    - using START DATABASE command 470
- LSTATS option of DISPLAY BUFFERPOOL command 175
- LTE option of MODIFY irlmproc,SET command of z/OS 386
- LTE option of START irlmproc command 485
- LUNAME option
  - DISPLAY LOCATION command 242
  - RESET INDOUBT command 434
- LUWID option
  - DISPLAY THREAD command 263
  - RECOVER INDOUBT command 425
  - RESET INDOUBT command 435

## M

- MACRO option of DSNH command 341
- MAINT option of MODIFY irlmproc,STATUS command of z/OS 390
- MAXCSA option of START irlmproc command 485
- MAXRO option of ALTER UTILITY command 65
- MAXUSRS option of START irlmproc command 485
- MCS consoles, scope of commands 21
- MDETAIL option of DISPLAY GROUPBUFFERPOOL command 229
- member detail report of DISPLAY GROUPBUFFERPOOL command 232

- MEMBER option
  - BIND PACKAGE subcommand 125
  - BIND PLAN subcommand 125
  - DISPLAY UTILITY command 306
- member, scope of command 21
- MERSECOPY utility, effects of TERM command 561
- message
  - DB2 commands 10
  - DCLGEN subcommand 154
  - DISPLAY UTILITY command 306
  - DSN command of TSO 313
  - DSNH command 340
  - FLAG option of bind subcommands 117
  - FREE PACKAGE subcommand 363
  - FREE PLAN subcommand 366
  - MODIFY irlmproc,STATUS command 391
  - RUN subcommand 439
- message by identifier
  - DSN7106I 220
  - DSN9022I 10
  - DSN9023I 10
  - DSNJ315I 70
  - DSNJ316I 70
  - DSNJ317I 70
  - DSNJ318I 70
  - DSNL440I to DSNL449I 435
  - DSNL448I 434
  - DSNL450I 144
  - DSNT392I 199
  - DSNT500I 200
  - DSNT501I 200
  - DSNT736I 525
  - DSNU100I 306
  - DSNU105I 306
  - DSNU106I 306
  - DSNW133I 500
  - DSNX943I 251
  - DSNX950I 252
- MLT option of MODIFY irlmproc,SET command of z/OS 386
- MODE option
  - ARCHIVE LOG command 68
  - STOP DB2 command 529
  - STOP DDF command 531
- MODIFY admtproc
  - shutdown 371
  - Trace 373
- MODIFY admtproc command of z/OS
  - Examples 373
- MODIFY admtproc,,APPL=SHUTDOWN command of z/OS
  - description 371
- MODIFY admtproc,,APPL=TRACE command of z/OS
  - description 373
- MODIFY admtproc,APPL=SHUTDOWN command of z/OS
  - Examples 371
- MODIFY DDF 375
- MODIFY DDF command 375
- MODIFY irlmproc,ABEND command of z/OS
  - description 379
  - Example 380
- MODIFY irlmproc,DIAG command of z/OS
  - description 381
  - Example 382
- MODIFY irlmproc,PURGE command of z/OS
  - description 383
  - Example 383
- MODIFY irlmproc,SET command of z/OS
  - description 385

- MODIFY irlmproc,SET command of z/OS *(continued)*
  - example 387
  - option descriptions 383, 385
- MODIFY irlmproc,STATUS command of z/OS
  - description 389
  - example 391
  - Option descriptions 389
- MODIFY RECOVERY utility, effects of TERM command 561
- MODIFY STATISTICS utility, effects of TERM command 561
- MODIFY TRACE command
  - description 401
  - Example 403
- modifying 375
- MONITOR option
  - DISPLAY TRACE command 297
  - MODIFY TRACE command 402
  - STOP TRACE command 553
- monitor trace
  - displaying 293
  - starting 497
  - stopping 549
- MSTR option of START DB2 command 474

## N

- NAMES option of DCLGEN subcommand 154
- NEWFUN option of DSNH command 341
- NEWLOG option
  - SET LOG command 447
- NID (network ID) option of RECOVER INDOUBT command 425
- NO LIMIT option of SET ARCHIVE command 442
- NO option
  - START irlmproc command 486
- NOBACKOUT option of CANCEL THREAD command 144
- NODEFER option
  - BIND PACKAGE subcommand 102
  - BIND PLAN subcommand 102
  - DSNH command 350
  - REBIND PACKAGE subcommand 102
  - REBIND PLAN subcommand 102
- NODISCON option of START irlmproc command 486
- NODUMP option of MODIFY irlmproc,ABEND command 379
- NOFOR option of DSNH command 342
- NOINDOUBTS option
  - START DB2 command 473
- NONE keyword of MODIFY irlmproc,DIAG command 382
- NOWRAP option of TRACE CT command 566

## O

- OASN option
  - CHANGE command 149
  - DISPLAY command 161
- online 577
- online books 577
- ONLY option of DISPLAY DATABASE command 194
- OP option
  - START TRACE command 501
  - STOP TRACE command 553
- operands, DB2 commands 15
- OPTHINT option
  - BIND PACKAGE subcommand 126
  - BIND PLAN subcommand 126
  - DSNH command 350

- OPTHINT option *(continued)*
  - REBIND PACKAGE subcommand 126
  - REBIND PLAN subcommand 126
- OPTIONS option
  - BIND PACKAGE subcommand 126
  - DSNH command 342
- OUTNAME option of DSNH command 342
- output size 23
- OVERVIEW option of DISPLAY DATABASE command 195
- OWNER option
  - BIND PACKAGE subcommand 127
  - BIND PLAN subcommand 127
  - DCLGEN subcommand 153
  - DSNH command 350
  - REBIND PACKAGE subcommand 127
  - REBIND PLAN subcommand 127

## P

- P irlmproc command. 539
- package
  - binding, initiating 73
  - identifier
    - BIND PACKAGE subcommand 128
    - REBIND TRIGGER PACKAGE subcommand 419
  - rebinding 405
  - rebinding trigger 417
  - replacing version of 93
- PACKAGE option of DSNH command 352
- PACTION option of DSNH command 351
- parallel processing
  - DEGREE option of bind subcommands 104
  - VPSEQT option of ALTER BUFFERPOOL command 54
- parameter, passing to application program 438
- parameters 18
- PARM option of START DB2 command 474
- PARMS option
  - DSNH command 342
  - RUN subcommand 438
- parsing rules, DB2 commands 15
- PART option
  - DISPLAY DATABASE command 194
  - START DATABASE command 468
  - STOP DATABASE command 524
- partial-location name, DISPLAY LOCATION command 242
- PASS option of DSNH command 342
- PATH option
  - BIND PACKAGE subcommand 129
  - BIND PLAN subcommand 129
  - DSNH command 350
  - REBIND PACKAGE subcommand 129
  - REBIND PLAN subcommand 129
- PATHDEFAULT option
  - REBIND PACKAGE subcommand 131
  - REBIND PLAN subcommand 131
- PBIND option of DSNH command 351
- PC option of START irlmproc command 486
- PCICS option of DSNH command 351
- PCLOAD option of DSNH command 342
- PDBPROTOCOL option of DSNH command 351
- PDBRMLIB option of DSNH command 351
- PDEFER option of DSNH command 351
- PDEGREE option of DSNH command 351
- PDISABLE option of DSNH command 351
- PDLIBATCH option of DSNH command 351
- PDMEM option of DSNH command 351
- PDYNAMICRULES option of DSNH command 352

- PENABLE option of DSNH command 352
- PERFM option
  - DISPLAY TRACE command 297
  - MODIFY TRACE command 402
  - STOP TRACE command 552
- performance trace
  - displaying 293
  - stopping 549
- performance, trace
  - starting 497
- PFLAG option of DSNH command 352
- PGPROT option of START irlmproc command 486
- PGSTEAL option of ALTER BUFFERPOOL command 55
- phases of execution for DSNH processing 329
- PIMSBMP option of DSNH command 352
- PIMSMPP option of DSNH command 352
- PISOLATION option of DSNH command 352
- PKEEPDYNAMIC option of DSNH command 352
- PKGCOL option
  - DISPLAY TRACE command 296
  - START TRACE command 502
  - STOP TRACE command 552
- PKGLOC option
  - DISPLAY TRACE command 296
  - START TRACE command 502
  - STOP TRACE command 552
- PKGPROG option
  - DISPLAY TRACE command 296
  - START TRACE command 502
  - STOP TRACE command 552
- PKLIST option
  - BIND PLAN subcommand 131
  - DSNH command 350
  - REBIND PLAN subcommand 131
- PL/I application program, macro processing step for DSNH 329
- PLAN
  - option of DSNH command 342
- PLAN option
  - BIND PLAN subcommand 133
  - DISPLAY TRACE command 296
  - DSNC DISPLAY command 319
  - DSNH command 350
  - REBIND PLAN subcommand 133
  - RUN subcommand 438
  - START TRACE command 502
  - STOP TRACE command 552
- PLANMGMT option
  - BIND PACKAGE subcommand 133
  - REBIND PACKAGE subcommand 133
  - REBIND TRIGGER PACKAGE subcommand 133
- PLANMGMTSCOPE option
  - FREE PACKAGE subcommand 363
- PLI option of DCLGEN subcommand 154
- PLI2LIB option of DSNH command 343
- PLIB option of DSNH command 342
- PLLIB option of DSNH command 343
- PLILOAD option of DSNH command 343
- PLIPLNK option of DSNH command 343
- PLIPMSG option of DSNH command 343
- PLOCK keyword of MODIFY irlmproc,DIAG command 381
- PnLIB option of DSNH command 342
- PNODEFER option of DSNH command 352
- POPTHINT option of DSNH command 352
- POPTION option of DSNH command 343
- POSTPONED option
  - DISPLAY THREAD command 262

- postponed units of recovery, recovering 427
- POWNER option of DSNH command 352
- PPATH option of DSNH command 352
- PQUALIFIER option of DSNH command 352
- PRECOMP option of DSNH command 343
- precompiler
  - DSNH command options 342
  - invoking DSNH 329
  - producing members for 125
- PRELEASE option of DSNH command 352
- PRELINK option of DSNH command 343
- PREOPT option of DSNH command 352
- PRINT option of DSNH command 344
- privilege set of a process 3
- PROC option
  - DISPLAY THREAD command 262
- processing, parallel
  - VPPSEQT option of ALTER BUFFERPOOL command 54
- product-sensitive programming information, described 585
- PROGRAM option of RUN subcommand 437
- programming interface information, described 585
- PSECPAC option of DSNH command 344
- PSPACE option of DSNH command 344
- PSPI symbols 585
- PVALIDATE option of DSNH command 353
- PVT option of MODIFY irlmproc,SET command of z/OS 386

## Q

- QUALIFIER option
  - BIND PACKAGE subcommand 134
  - BIND PLAN subcommand 134
  - DSNH command 350
  - REBIND PACKAGE subcommand 134
  - REBIND PLAN subcommand 134
- query
  - binding 87
  - removal 367
- QUEUE option of STOP FUNCTION SPECIFIC 536, 537
- QUIESCE option
  - DSNC STOP command 325
  - STOP DB2 command 529
  - STOP DDF command 531
- QUIESCE utility, effects of TERM command 561
- QUOTE option
  - DCLGEN subcommand 155
  - DSNH command 340

## R

- RATIO option of ALTER GROUPBUFFERPOOL command 60
- RCTERM option of DSNH command 344
- REBIND PACKAGE subcommand of DSN
  - description 405
  - example 408
  - option descriptions 91
- REBIND PLAN subcommand of DSN
  - description 411
  - option descriptions 91
- REBIND TRIGGER PACKAGE subcommand of DSN
  - description 417
  - example 419
  - option descriptions 91
- rebinding
  - initiating 405, 411
  - options for 91



- REBUILD INDEX utility, TERM command effects 561
- recognition character 15
- recorded in DB2 recovery log 45
- RECOVER BSDS command
  - description 421
  - Example 421
- RECOVER INDOUBT command
  - description 423
  - example 426
  - Option descriptions 424
- RECOVER POSTPONED command
  - description 427
  - Example 428
- RECOVER utility, TERM command effects 561
- recovery
  - BSDS 421
  - indoubt threads 423
  - postponed units of recovery 427
- REFRESH DB2 command
  - example 430
- REFRESH DB2, EARLY DB2 command
  - description 429
- REFRESH DB2,EARLY command
  - Option descriptions 430
- REGISTER XMLSCHEMA command
  - example 40, 41
- REJECT option of STOP FUNCTION SPECIFIC 536, 537
- RELEASE option
  - BIND PACKAGE subcommand 135
  - BIND PLAN subcommand
    - description 135
  - DSNH command 350
  - REBIND PACKAGE subcommand 135
  - REBIND PLAN subcommand 135
  - REBIND TRIGGER PACKAGE subcommand 135
- RELOAD option of SET SYSPARM command 452
- REMOTE option of DSNH command 352
- REMOVE XMLSCHEMA command
  - example 42
- REOPT option
  - BIND PACKAGE subcommand 137, 138
  - BIND PLAN subcommand 137, 138
  - DSNH command 350
  - REBIND PACKAGE subcommand 137, 138
  - REBIND PLAN subcommand 137, 138
- REORG INDEX utility, effects of TERM command 561
- REORG TABLESPACE utility, effects of TERM command 561
- REPAIR utility, effects of TERM command 561
- REPLACE option
  - DCLGEN subcommand 154
  - DSNH command 348, 351
- replacing, version of a package 93
- REPLVER option
  - BIND PACKAGE subcommand 92
  - DSNH command 353
  - effect of 93
- REPORT utility, effects of TERM command 561
- reports
  - group detail report 230
  - member detail report 232
  - summary report
    - DISPLAY GROUPBUFFERPOOL command 234
- RES option of STOP TRACE command 553
- RESET GENERICLU command
  - description 431
  - example 432
  - Option descriptions 431
- RESET INDOUBT command
  - description 433
  - Option descriptions 434
- RESET option of CHANGE command 150
- resource limit facility
  - displaying 197
- restarting
  - CICS attachment facility 325
  - connections between IMS and a subsystem 463
  - status of DB2 resources 429, 473
  - terminated utility job steps 560
- RESTRICT option of DISPLAY DATABASE command 195
- RESUME option
  - SET LOG command 447
- RETAIN option
  - BIND PLAN subcommand 92
  - DSNH command 350
- retained locks 194
- RETRY option
  - DSN command 312
- return code
  - DSN command 313
  - RUN subcommand of DSN 313
- RMARGIN option of DCLGEN subcommand 156
- RMID option
  - DISPLAY TRACE command 298
  - START TRACE command 500
  - STOP TRACE command 553
- RO option of START DATABASE command 469
- ROLE option
  - DISPLAY TRACE command 296
  - START TRACE command 504
  - STOP TRACE command 552
- RRSURID option of DISPLAY THREAD command 264
- RUN
  - subcommand of DSN
    - description 437
    - example 439
    - Option descriptions 437
    - return codes 313
- RUN option
  - DSNH command parameters 345
- RUNIN option of DSNH command 345
- running DSNH processing 329
- RUNOUT option of DSNH command 345
- RUNSTATS utility, effects of TERM command 561
- RW option of START DATABASE command 469

## S

- scanning rules, DB2 commands 15
- schema.partial-name option
  - DISPLAY FUNCTION SPECIFIC command 214
  - START FUNCTION SPECIFIC command 480
  - STOP FUNCTION SPECIFIC command 536
- schema.specific-function-name option
  - DISPLAY FUNCTION SPECIFIC command 214
  - START FUNCTION SPECIFIC command 480
  - STOP FUNCTION SPECIFIC command 536
- schema.specific-function-name option of DISPLAY FUNCTION SPECIFIC command 214
- scope of commands 21
- SCOPE option
  - ARCHIVE LOG command 69
  - DISPLAY FUNCTION SPECIFIC command 214
  - DISPLAY PROCEDURE command 250
  - DISPLAY THREAD command 261

SCOPE option (*continued*)  
  DISPLAY TRACE command 297  
  START FUNCTION SPECIFIC command 480  
  START irlmproc command 486  
  START PROCEDURE command 490  
  STOP FUNCTION SPECIFIC command 536  
  STOP PROCEDURE command 543  
  STOP TRACE command 553  
secondary authorization ID 3  
SET ARCHIVE command  
  description 441  
  example 443  
  Option descriptions 442  
SET LOG command  
  description 445  
  example 448  
  Option descriptions 446  
SET SYSPARM command  
  description 451  
  example 452  
  Option descriptions 452  
shortcut keys  
  keyboard x  
SINGLE option  
  SET LOG command 446  
SMF option  
  DISPLAY TRACE command 297  
  START TRACE command 501  
  STOP TRACE command 553  
softcopy publications 577  
SOMDLI option of DSNH command 345  
SOURCE option of DSNH command 345  
SPACENAM option  
  DISPLAY DATABASE command 192  
  START DATABASE command 467  
  STOP DATABASE command 523  
SPACEUN option of DSNH command 346  
SPUFI, description 453  
SQL option of DSNH command 346  
SQLDELIM option of DSNH command 346  
SQLERROR option  
  BIND PACKAGE subcommand 139  
  DSNH command 353  
SQLRULES option  
  BIND PLAN subcommand 140  
  REBIND PLAN subcommand 140  
SQLRULES option of DSNH command 350  
SRC (subsystem recognition character) 15  
SRV option  
  DISPLAY TRACE command 297  
  START TRACE command 501  
  STOP TRACE command 553  
SSR command of IMS, description 455  
START ACCEL command 457  
  Option descriptions 458  
START admtproc command of z/OS  
  description 461  
  Examples 462  
START command of IMS 463  
START DATABASE command  
  description 465  
  example 472  
  Option descriptions 467  
  recovering object in group buffer pool 470  
  recovering pages on logical page list 470  
START DB2 command  
  description 473  
START DB2 command (*continued*)  
  example 476  
  Option descriptions 474  
START DDF command 477  
START FUNCTION SPECIFIC command  
  description 479  
  example 481  
  Option descriptions 480  
START irlmproc command of z/OS  
  description 483  
  Examples 486  
  Option descriptions 483  
START PROCEDURE command  
  description 489  
  example 491  
  Option descriptions 490  
START PROFILE command  
  example 493  
START RLIMIT command  
  description 495  
  Example 496  
START TRACE command  
  description 497  
  example 512  
  Option descriptions 499  
STARTUP option of SET SYSPARM command 452  
STAT option  
  DISPLAY TRACE command 297  
  MODIFY TRACE command 402  
  STOP TRACE command 552  
STATISTICS option of DSNH DISPLAY command 319  
statistics trace  
  stopping 549  
statistics, trace  
  displaying 293  
  starting 497  
status  
  checking, IRLM 389  
  cross-system coupling facility (XCF), status of  
  members 220  
  shown by DISPLAY DATABASE 199  
STDSQL option of DSNH command 346  
STOP ACCEL command  
  description 515  
  example 516  
STOP admtproc command of z/OS  
  description 517  
  Examples 517  
STOP command of IMS 519  
STOP DATABASE command  
  description 521  
  example 526  
  Option descriptions 522  
STOP DB2 command  
  description 529  
  Example 530  
STOP DDF command  
  description 531  
  example 533  
STOP FUNCTION SPECIFIC command  
  description 535  
  Examples 537  
  limitations of 537  
  Option descriptions 536  
STOP irlmproc command of z/OS 539  
STOP PROCEDURE command  
  description 541

- STOP PROCEDURE command (*continued*)
  - example 543
  - Option descriptions 542
- STOP PROFILE command
  - example 545
- STOP RLIMIT command 547
- STOP TRACE command
  - description 549
  - example 558
  - Option descriptions 552
- STOR option of MODIFY irlmproc,STATUS command of z/OS 390
- stored procedure
  - displaying status 249
  - starting 489
  - stopping 541
- STOSPACE utility, effects of TERM command 561
- string, delimiter
  - COBOL 155
  - SQL 155
- STRUCTURE option of DCLGEN subcommand 154
- SUB option of TRACE CT command 566
- SUBS option of TRACE command 563
- SUBSYS option
  - CHANGE command 149
  - DISPLAY command 161
  - START command 463
  - STOP command 519
- SUFFIX option of DSNH command 346
- summary report
  - DISPLAY GROUPBUFFERPOOL command 229, 234
- SUSPEND option
  - SET LOG command 447
  - STOP DDF command 532
- SWITCH option
  - REBIND PACKAGE subcommand 141
  - REBIND TRIGGER PACKAGE subcommand 141
- syntax diagram
  - how to read xi
- SYSTEM option
  - DISPLAY THREAD command 262
  - DSN command 312
  - DSNH command 346

## T

- TABLE option of DCLGEN subcommand 152
- TDATA option of START TRACE command 508
- TERM option of DSNH command 347
- TERM UTILITY command
  - description 559
  - example 561
- terminating
  - connections between IMS and a subsystem 519
  - databases 521
  - DB2, description 529
  - IRLM
    - abnormal 379
    - normal 539
  - stored procedures 541
  - trace activity 549
  - utilities, description 559
- TEST option
  - DSN command 312
- thread
  - ACTIVE option
    - DISPLAY THREAD command 259

- thread (*continued*)
  - canceled 143
  - displaying 259
  - message
    - DISPLAY THREAD with ACTIVE 259
- TIME option
  - ARCHIVE LOG command 68
  - DSNH command 347
  - SET ARCHIVE command 442
- TIMEOUT option of MODIFY irlmproc,SET command of z/OS 386
- TNO option
  - DISPLAY TRACE command 298
  - MODIFY TRACE command 402
- trace
  - changing active traces 401
  - displaying 293
  - events 563
  - IFCIDs activated by trace class 504
  - starting 497
  - stopping 549
- TRACE command of IMS
  - description 563
  - example 563
- TRACE CT command of z/OS
  - description 565
  - example 567
  - Option descriptions 565
- TRACE option
  - MODIFY irlmproc,SET command of z/OS 386
  - MODIFY irlmproc,STATUS command of z/OS 390
  - START irlmproc command 486
  - START TRACE command 509
- TRACK option of DSNH 346
- TRANSACTION option
  - DSNC DISPLAY command 319
  - DSNC MODIFY command 323
- trusted context
  - authorization IDs 4
- TSO CLIST commands 13
- TSO CLISTs of DSNH 329
- TSO option of DSNH command 345
- TYPE option
  - DISPLAY GROUPBUFFERPOOL command 228
  - DISPLAY THREAD command 261

## U

- unit of recovery, in IMS 162
- unit of work
  - displaying an outstanding 161
  - resetting
    - IMS 149
    - indoubt 149
- UNLOAD utility, effects of TERM command 561
- USE option of DISPLAY DATABASE command 193
- USERID option
  - DISPLAY TRACE command 296
  - START TRACE command 503
  - STOP TRACE command 552
- UT option of START DATABASE command 469
- utilities
  - displaying status 305
  - identifier 560
  - terminating 559

## V

VALIDATE option  
    BIND PACKAGE subcommand 142  
    BIND PLAN subcommand 142  
    DSNH command 350  
    REBIND PACKAGE subcommand 142  
    REBIND PLAN subcommand 142  
VDWQT option of ALTER BUFFERPOOL command 55  
version of a package  
    BIND PACKAGE subcommand 128  
    REBIND PACKAGE subcommand 128  
VERSION option of DSNH command 347  
VPPSEQT option of ALTER BUFFERPOOL command 54  
VPSEQT option of ALTER BUFFERPOOL command 54  
VPSIZE option of ALTER BUFFERPOOL command 54  
VPXPSEQT option of ALTER BUFFERPOOL command 54  
VSAM (virtual storage access method) password, DCLGEN  
    subcommand 153  
VTAM (Virtual Telecommunications Access Method), DISPLAY  
    NET command 145

## W

WAIT option of ARCHIVE LOG command 69  
WEPR option of DISPLAY DATABASE command 194  
WORKUNIT option of DSNH command 347  
WRAP option of TRACE CT command 565  
WRKSTN option  
    DISPLAY TRACE command 296  
    START TRACE command 504  
    STOP TRACE command 552  
WSECPAC option of DSNH command 347  
WSPACE option of DSNH command 347  
WTRSTART option of TRACE CT command 565  
WTRSTOP option of TRACE CT command 566

## X

XAPPNAME option  
    DISPLAY TRACE command 296  
    START TRACE command 504  
    STOP TRACE command 552  
XAUTHID option  
    DISPLAY TRACE command 296  
    START TRACE command 502  
    STOP TRACE command 552  
XCF (cross-system coupling facility), status of members 220  
XCONNID option  
    DISPLAY TRACE command 296  
    START TRACE command 504  
    STOP TRACE command 552  
XCORRID option  
    DISPLAY TRACE command 296  
    START TRACE command 504  
    STOP TRACE command 552  
XLIB option of DSNH command 347  
XLOC option  
    START TRACE command 296, 503  
XPKGCOL option  
    DISPLAY TRACE command 296  
    START TRACE command 502  
    STOP TRACE command 552  
XPKGLOC option  
    DISPLAY TRACE command 296  
    START TRACE command 502  
    STOP TRACE command 552

XPKGPROG option  
    DISPLAY TRACE command 296  
    START TRACE command 502  
    STOP TRACE command 552  
XPLAN option  
    DISPLAY TRACE command 296  
    START TRACE command 502  
    STOP TRACE command 552  
XREF option of DSNH command 347  
XROLE option  
    DISPLAY TRACE command 296  
    START TRACE command 504  
    STOP TRACE command 552  
XUSERID option  
    DISPLAY TRACE command 296  
    START TRACE command 503  
    STOP TRACE command 552  
XWRKSTN option  
    DISPLAY TRACE command 296  
    START TRACE command 504  
    STOP TRACE command 552

## Y

YES option  
    START DB2 command 474  
    START irlmproc command 486

## Z

z/OS commands 371, 373  
    MODIFY irlmproc,ABEND 379  
    MODIFY irlmproc,DIAG 381  
    MODIFY irlmproc,PURGE 383  
    MODIFY irlmproc,SET 385  
    MODIFY irlmproc,STATUS 389  
    START admtproc 461  
    START irlmproc 483  
    STOP admtproc 517  
    STOP irlmproc 539  
    TRACE CT 565





Product Number: 5605-DB2  
5697-P31

Printed in USA

SC19-2972-05



Spine information:

DB2 10 for z/OS

Command Reference

