DB2 Universal Database for OS/390

# Command Reference

*Version 6*

SC26-9006-01

**Note!**

Before using this information and the product it supports, be sure to read the general information under Appendix A, “Notices” on page 345.

**Second Edition, Softcopy Only (April 2000)**

This edition applies to Version 6 of DB2 Universal Database Server for OS/390, 5645-DB2, 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.

This softcopy version is based on the printed edition of the book and includes the changes indicated in the printed version by vertical bars. Additional changes made to this softcopy version of the manual since the hardcopy manual was published are indicated by the hash (#) symbol in the left-hand margin. Editorial changes that have no technical significance are not noted.

# Contents

# Chapter 1. Introduction to this book and the DB2 for OS/390 library

This chapter contains specific information about this book, a general overview of the library for DB2® for OS/390®, a summary of changes to DB2, and a description of naming conventions, privileges, authorization IDs, and the bind process.

## Who should read this book

This book presents reference information for the tasks of system administration, database administration, and operation. It presents detailed information on commands, including syntax, option descriptions, and examples for each command.

## How to use this book

# This book is intended to serve as a reference for people who understand system
# administration, database administration, or application programming in the DB2
# environment. You should have some knowledge of:

# • CICS, IMS, or TSO
# • A programming language (Assembler language, PL/I, COBOL, APL2®, BASIC,
# FORTRAN, PROLOG, or C)
# • MVS Job Control Language (JCL)
# • Structured Query Language (SQL)

## # Product terminology and citations

In this book, DB2 Universal Database Server for OS/390 is referred to as "DB2 for OS/390." In cases where the context makes the meaning clear, DB2 for OS/390 is referred to as "DB2." When this book refers to other books in this library, a short title is used. (For example, "See *DB2 SQL Reference*" is a citation to *IBM DATABASE 2 Universal Database Server for OS/390 SQL Reference*.)

References in this book to "DB2 UDB" relate to the DB2 Universal Database™ product that is available on the AIX®, OS/2®, and Windows NT™ operating systems. When this book refers to books about the DB2 UDB product, the citation includes the complete title and order number.

The following terms are used as indicated:

| | |
|---|---|
| **DB2®** | Represents either the DB2 licensed program or a particular DB2 subsystem. |
| **C and C language** | Represent the C programming language. |
| **CICS®** | Represents CICS/ESA® and CICS Transaction Server for OS/390 Release 1. |
| **IMS™** | Represents IMS/ESA®. |
| **MVS** | Represents the MVS element of OS/390. |

## How to read the syntax diagrams

The following rules apply to the syntax diagrams used in this book:

- 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.

  Diagrams of syntactical units other than complete statements start with the ▶─── symbol and end with the ───▶ symbol.

- 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.

  ```
                     ┌─optional_item─┐
  ▶▶──required_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.

  ```
  ▶▶──required_item──┬─required_choice1─┬────────────────────▶◀
                     └─required_choice2─┘
  ```

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

  ```
  ▶▶──required_item──┬──────────────────┬────────────────────▶◀
                     ├─optional_choice1─┤
                     └─optional_choice2─┘
  ```

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

  ```
                     ┌─default_choice──┐
  ▶▶──required_item──┼─────────────────┼─────────────────────▶◀
                     ├─optional_choice─┤
                     └─optional_choice─┘
  ```

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

  ```
                     ┌───────────────┐
  ▶▶──required_item──▼─repeatable_item─┴─────────────────────▶◀
  ```

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

```
                   ┌─,───────────────┐
►►──required_item───▼─repeatable_item─┴─────────────────────────►◄
```

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

- Keywords appear in uppercase (for example, `FROM`). They 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.

## How to use the DB2 library

Titles of books in the library begin with DB2 Universal Database for OS/390 Version 6. 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. For a complete list of books in the library, and the sections in each book, see the bibliography at the back of this book.

Throughout the library, the DB2 for OS/390 licensed program and a particular DB2 for MVS/ESA subsystem are each referred to as “DB2.” In each case, the context makes the meaning clear.

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 associated with DB2 are grouped into the following major categories (but supplemental information relating to all of the below tasks for new releases of DB2 can be found in *DB2 Release Guide*):

***Installation:*** If you are involved with DB2 only to install the system, *DB2 Installation Guide* might be all you need.

If you will be using data sharing then you also need *DB2 Data Sharing: Planning and Administration*, which describes installation considerations for data sharing.

***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 SPUFI, 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 Query Management Facility (QMF™) or some other program, and the library for that program might provide all the instruction or reference material they need. For a list of the titles in the QMF library, see the bibliography at the end of this book.

***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 *DB2 Application*

*Programming and SQL Guide*, *DB2 SQL Reference*, and *DB2 ODBC Guide and Reference* just as end users do.

Application programmers also need instructions on many other topics:

- How to transfer data between DB2 and a host program—written in COBOL, C, or FORTRAN, for example
- How to prepare to compile a program that embeds SQL statements
- How to process data from two systems simultaneously, say DB2 and IMS or DB2 and CICS®
- How to write distributed applications across platforms
- How to write applications that use DB2 ODBC to access DB2 servers
- How to write applications that use Open Database Connectivity (ODBC) to access DB2 servers
- How to write applications in the Java™ programming language to access DB2 servers

The material needed for writing a host program containing SQL is in *DB2 Application Programming and SQL Guide* and in *DB2 Application Programming Guide and Reference for Java™*. The material needed for writing applications that use DB2 ODBC or ODBC to access DB2 servers is in *DB2 ODBC Guide and Reference*. For handling errors, see *DB2 Messages and Codes* .

Information about writing applications across platforms can be found in *Distributed Relational Database Architecture™: Application Programming Guide*.

***System and Database Administration:*** *Administration* covers almost everything else. *DB2 Administration Guide* divides those tasks among the following sections:

- Section 2 (Volume 1) of *DB2 Administration Guide* discusses the decisions that must be made when designing a database and tells how to bring the design into being by creating DB2 objects, loading data, and adjusting to changes.
- Section 3 (Volume 1) of *DB2 Administration Guide* describes ways of controlling access to the DB2 system and to data within DB2, to audit aspects of DB2 usage, and to answer other security and auditing concerns.
- Section 4 (Volume 1) of *DB2 Administration Guide* 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.
- Section 5 (Volume 2) of *DB2 Administration Guide* 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.

In addition, the appendixes in *DB2 Administration Guide* contain valuable information on DB2 sample tables, National Language Support (NLS), writing exit routines, interpreting DB2 trace output, and character conversion for distributed data.

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, then 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, then 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 Codes*, which lists messages and codes issued by DB2, with explanations and suggested responses.

***Diagnosis:*** Diagnosticians 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* and *DB2 Messages and Codes*.

## How to obtain DB2 information

## DB2 on the Web

Stay current with the latest information about DB2. View the DB2 home page on the World Wide Web. News items keep you informed about the latest enhancements to the product. Product announcements, press releases, fact sheets, and technical articles help you plan your database management strategy.

You can view and search DB2 publications on the Web, or you can download and print many of the most current DB2 books. Follow links to other Web sites with more information about DB2 family and OS/390 solutions. Access DB2 on the Web at the following address:

http://www.ibm.com/software/db2os390

## DB2 publications

The DB2 publications for DB2 Universal Database Server for OS/390 are available in both hardcopy and softcopy format.

### BookManager® format

Using online books on CD-ROM, you can read, search across books, print portions of the text, and make notes in these BookManager books. With the appropriate BookManager READ product or IBM Library Readers, you can view these books in the OS/390, VM, OS/2, DOS, AIX, and Windows™ environments. You can also view many of the DB2 BookManager books on the Web.

### PDF format

Many of the DB2 books are available in Portable Document Format (PDF) for viewing or printing from CD-ROM or the Web. Download the PDF books to your intranet for distribution throughout your enterprise.

### CD-ROMs

Books for Version 6 of DB2 Universal Database Server for OS/390 are available on CD-ROMs:

- *DB2 UDB for OS/390 Version 6 Licensed Online Book*, LK3T-3519, containing *DB2 UDB for OS/390 Version 6 Diagnosis Guide and Reference* in BookManager format, for ordering with the product.
- *DB2 UDB Server for OS/390 Version 6 Online and PDF Library*, SK3T-3518, a collection of books for the DB2 server in BookManager and PDF formats.

Periodically, the books will be refreshed on subsequent editions of these CD-ROMs.

The books for Version 6 of DB2 UDB Server for OS/390 are also available on the following collection kits that contain online books for many IBM products:

- *Online Library Omnibus Edition OS/390 Collection*, SK2T-6700, in English
- *IBM Online Library MVS Collection Kit*, SK88-8002, in Japanese, for viewing on DOS and Windows operating systems.

## DB2 education

IBM Education and Training offers a wide variety of classroom courses to help you quickly and efficiently gain DB2 expertise. Classes are scheduled in cities all over the world. You can find class information, by country, at the IBM Learning Services Web site:

http://www.ibm.com/services/learning/

For more information, including the current local schedule, please contact your IBM representative.

Classes can also be taught at your location, at a time that suits your needs. Courses can even be customized to meet your exact requirements. The *All-in-One Education and Training Catalog* describes the DB2 curriculum in the United States. You can inquire about or enroll in these courses by calling 1-800-IBM-TEACH (1-800-426-8322).

## How to order the DB2 library

You can order DB2 publications and CD-ROMs through your IBM representative or the IBM branch office serving your locality. If you are located 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 1-800-565-1234.

To order additional copies of licensed publications, specify the SOFTWARE option. To order additional publications or CD-ROMs, specify the PUBLICATIONS and SLSS option. Be prepared to give your customer number, the product number, and the feature code(s) or order numbers you want.

## Summary of changes to DB2 UDB for OS/390 Version 6

DB2 UDB for OS/390 Version 6 delivers an enhanced relational database server solution for OS/390. This release focuses on greater capacity, performance improvements for utilities and queries, easier database management, more powerful network computing, and DB2 family compatibility with rich new object-oriented capability, triggers, and more built-in functions.

## Capacity improvements

**16-terabyte tables** provide a significant increase to table capacity for partitioned and LOB table spaces and indexes, and for nonpartitioning indexes.

**Buffer pools in data spaces** provide virtual storage constraint relief for the ssnmDBM1 address space, and data spaces increase the maximum amount of virtual buffer pool space allowed.

## Performance and availability

**Improved partition rebalancing** lets you redistribute partitioned data with minimal impact to data availability. One REORG of a range of partitions both reorganizes and rebalances the partitions.

You can **change checkpoint frequency dynamically** using the new SET LOG command and initiate checkpoints any time while your subsystem remains available.

**Utilities that are faster, more parallel, easier to use**:

- **Faster backup and recovery** enables COPY and RECOVER to process a list of objects in parallel, and recover indexes and table spaces at the same time from image copies and the log.
- **Parallel index build** reduces the elapsed time of LOAD and REORG jobs of table spaces, or partitions of table spaces, that have more than one index; the elapsed time of REBUILD INDEX jobs is also reduced.
- Tests show **decreased elapsed and processor time for online REORG**.
- **Inline statistics** embeds statistics collection into utility jobs, making table spaces available sooner.
- You can **determine when to run REORG** by specifying threshold limits for relevant statistics from the DB2 catalog.

**Query performance enhancements** include:

- **Query parallelism extensions** for complex queries, such as outer joins and queries that use nonpartitioned tables
- **Improved workload balancing in a Parallel Sysplex**® that reduces elapsed time for a single query that is split across active DB2 members
- **Improved data transfer** that lets you request multiple DRDA query blocks when performing high-volume operations
- The ability to use an **index to access predicates with noncorrelated IN subqueries**
- **Faster query processing** of queries that include join operations

**More performance and availability enhancements** include:

- **Faster restart and recovery** with the ability to postpone backout work during restart, and a faster log apply process
- **Increased flexibility with 8-KB and 16-KB page sizes** for balancing different workload requirements more efficiently, and for controlling traffic to the coupling facility for some workloads
- **Direct-row access** using the new ROWID data type to re-access a row directly without using the index or scanning the table
- **Ability to retain prior access path** when you rebind a statement. You almost always get the same or a better access path. For the exceptional cases, Version 6 of DB2 for OS/390 lets you retain the access path from a prior BIND by using rows in an Explain table as input to optimization.
- An **increased log output buffer size** (from 1000 4-KB to 100000 4-KB buffers) that improves log read and write performance

## Data sharing enhancements

**More caching options** use the coupling facility to improve performance in a data sharing environment for some applications by writing changed pages directly to DASD.

**Control of space map copy maintenance** with a new option avoids tracking of page changes, thereby optimizing performance of data sharing applications.

## User productivity

**Predictive governing** capabilities enhance the resource limit facility to help evaluate resource consumption for queries that run against large volumes of data.

**Statement cost estimation** of processing resource that is needed for an SQL statement helps you to determine error and warning thresholds for governing, and to decide which statements need tuning.

A **default buffer pool** for user data and indexes isolates user data from the DB2 catalog and directory, and separating user data from system data helps you make better tuning decisions.

**More information available for monitoring DB2** includes data set I/O activity in traces, both for batch reporting and online monitors.

**Better integration of DB2 and Workload Manager delay reporting** enables DB2 to notify Workload Manager about the current state of a work request.

**More tables are allowed in SQL statements** SELECT, UPDATE, INSERT, and DELETE, and in views. DB2 increases the limit from 15 to 225 tables. The number of tables and views in a subselect is not changed.

**Improved DB2 UDB family compatibility** includes SQL extensions, such as:

- A VALUES clause of INSERT that supports any expression
- A new VALUES INTO statement

**Easier recovery management** lets you achieve a single point of recovery and recover data at a remote site more easily.

**Enhanced database commands** extend support for pattern-matching characters (*) and let you filter display output.

You can easily **process dynamic SQL in batch mode** with the new object form of DSNTEP2 shipped with DB2 for OS/390.

## Network computing

**SQLJ**, the newest Java implementation for the OS/390 environment, supports SQL embedded in the Java programming language. With SQLJ, your Java programs benefit from the superior performance, manageability, and authorization available to static SQL, and they are easy to write.

**DRDA® support for three-part names** offers more functionality to applications using three-part names for remote access and improves the performance of client/server applications.

**Stored procedure enhancements** include the ability to create and modify stored procedure definitions, make nested calls for stored procedures and user-defined functions, and imbed CALL statements in application programs or dynamically invoke CALL statements from IBM's ODBC and CLI drivers.

**DB2 ODBC extensions** include new and modified APIs and new data types to support the object-relational extensions.

**ODBC access to DB2 for OS/390 catalog data** improves the performance of your ODBC catalog queries by redirecting them to shadow copies of DB2 catalog tables.

**Better performance for ODBC applications** reduces the number of network messages that are exchanged when an application executes dynamic SQL.

**Improvements for dynamically prepared SQL statements** include a new special register that you use to implicitly qualify names of distinct types, user-defined functions, and stored procedures.

**DDF connection pooling** uses a new type of inactive thread that improves performance for large volumes of inbound DDF connections.

## Object-relational extensions and active data

The object extensions of DB2 offer the benefits of object-oriented technology while increasing the strength of your relational database with an enriched set of data types and functions. Complementing these extensions is a powerful mechanism, triggers, that brings application logic into the database that governs the following new structures:

- **Large objects (LOBs)** are well suited to represent large, complex structures in DB2 tables. Now you can make effective use of multimedia by storing objects such as complex documents, videos, images, and voice. Some key elements of LOB support include:
  - LOB data types for storing byte strings up to 2 GB in size
  - LOB locators for easily manipulating LOB values in manageable pieces
  - Auxiliary tables (that reside in LOB table spaces) for storing LOB values

- **Distinct types** (which are sometimes called user-defined data types), like built-in data types, describe the data that is stored in columns of tables where the instances (or objects) of these data types are stored. They ensure that only those functions and operators that are explicitly defined on a distinct type can be applied to its instances.
- **User-defined functions**, like built-in functions or operators, support manipulation of distinct type instances (and built-in data types) in SQL queries.
- **New and extended built-in functions** improve the power of the SQL language with about 100 new built-in functions, extensions to existing functions, and sample user-defined functions.

**Triggers** automatically execute a set of SQL statements whenever a specified event occurs. These statements validate and edit database changes, read and modify the database, and invoke functions that perform operations inside and outside the database.

You can use the **DB2 Extenders** feature of DB2 for OS/390 to store and manipulate image, audio, video, and text objects. The extenders automatically capture and maintain object information and provide a rich body of APIs.

## More function

Some function and capability is available to both Version 6 and Version 5 users. Learn how to obtain these functions now, prior to migrating to Version 6, by visiting the following Web site:

http://www.software.ibm.com/data/db2/os390/v5apar.html

## Features of DB2 for OS/390

DB2 for OS/390 Version 6 offers a number of tools, which are optional features of the server, that are shipped to you automatically when you order DB2 Universal Database for OS/390:

- DB2 Management Tools Package, which includes the following elements:
  - DB2 UDB Control Center
  - DB2 Stored Procedures Builder
  - DB2 Installer
  - DB2 Visual Explain
  - DB2 Estimator
- Net.Data® for OS/390

You can install and use these features in a “Try and Buy” program for up to 90 days without paying license charges:

- Query Management Facility
- DB2 DataPropagator™
- DB2 Performance Monitor
- DB2 Buffer Pool Tool
- DB2 Administration Tool

## Migration considerations

Migration to Version 6 eliminates all type 1 indexes, shared read-only data, data set passwords, use of host variables without the colon, and RECOVER INDEX usage. You can migrate to Version 6 only from a Version 5 subsystem.

## Summary of changes to this book

Specific changes to this publication, reflecting the functional enhancements described above, are summarized below.

"Chapter 2. Commands" on page 17 introduces the following new commands:

- DISPLAY FUNCTION SPECIFIC (DB2)
- DISPLAY LOG (DB2)
- MODIFY irlmproc,DIAG,DELAY (MVS IRLM)
- MODIFY irlmproc,SET (MVS IRLM)
- REBIND TRIGGER PACKAGE (DSN)
- RECOVER POSTPONED (DB2)
- SET LOG (DB2)
- START FUNCTION SPECIFIC (DB2)
- STOP FUNCTION SPECIFIC (DB2)

New options are available for the following commands:

| | |
|---|---|
| ALTER BUFFERPOOL (DB2) | DISPLAY THREAD (DB2) |
| ALTER GROUPBUFFERPOOL (DB2) | DSNH (TSO CLIST) |
| BIND PACKAGE (DSN) | REBIND PACKAGE (DSN) |
| BIND PLAN (DSN) | REBIND PLAN (DSN) |
| DISPLAY BUFFERPOOL (DB2) | START DATABASE (DB2) |
| DISPLAY DATABASE (DB2) | START PROCEDURE (DB2) |
| DISPLAY GROUPBUFFERPOOL (DB2) | STOP DATABASE (DB2) |
| DISPLAY PROCEDURE (DB2) | STOP PROCEDURE (DB2) |

## Naming conventions

When a parameter refers to an object created by SQL statements (for example, tables, table spaces, and indexes), SQL syntactical naming conventions are followed.

This section describes naming conventions unique to commands. Characters are classified as *letters*, *digits*, or *special characters*.

- A *letter* is any one of the uppercase characters A through Z (plus the three characters reserved as alphabetic extenders for national languages, #, @, and $ in the United States).
- A *digit* is any one of the characters 0 through 9.
- A *special character* is any character other than a letter or a digit.

See Chapter 3 of *DB2 SQL Reference* for an additional explanation of long identifiers, short identifiers, and location identifiers.

*authorization-id*
A short identifier of 1 to 8 letters, digits, or the underscore that identifies a set of privileges. An authorization ID must begin with a letter.

*collection-id*
An SQL long identifier of 1 to 18 letters, digits, or the underscore that identifies a collection of packages; therefore, a collection ID is a qualifier for a package ID. A collection ID must begin with a letter.

A collection ID should not begin with DSN; this can sometimes conflict with DB2-provided collection IDs. If a collection ID beginning with DSN is specified, DB2 issues a warning message.

*connection-name*
An identifier of 1 to 8 characters that identifies an address space connection to DB2. A connection identifier is one of the following:

- For DSN processes running in TSO foreground, the connection name “TSO” is used.
- For DSN processes running in TSO batch, the connection name BATCH is used.
- For the call attachment facility (CAF), the connection name DB2CALL is used.
- For the Recoverable Resource Manager Services attachment facility (RRSAF), the connection name RRSAF is used.
- For IMS and CICS processes, the connection name is the system identification name.

See Section 4 (Volume 1) of *DB2 Administration Guide* for more information about connection names.

*correlation-id*
An identifier of 1 to 12 characters that identifies a process within an address space connection. A correlation ID must begin with a letter.

A correlation ID can be one of the following:

- For DSN processes running in TSO foreground, the correlation ID is the TSO logon identifier.
- For DSN processes running in TSO batch, the correlation ID is the job name.
- For CAF processes, the correlation ID is the TSO logon identifier.
- For RRSAF processes, the correlation ID is the value specified during signon.
- For IMS processes, the correlation ID is *pst#.psbname*.
- For CICS processes, the correlation ID is *identifier.thread_number.transaction_identifier.*

See Section 4 (Volume 1) of *DB2 Administration Guide* for more information about correlation IDs.

*data-set-name*
An identifier of 1 to 44 characters that identifies a data set.

*dbrm-member-name*
: An identifier of 1 to 8 letters or digits that identifies a member of a partitioned data set. (MVS requires this naming convention.)

  A DBRM member name should not begin with DSN; this can sometimes conflict with DB2-provided DBRM member names. If a DBRM member name beginning with DSN is specified, DB2 issues a warning message.

*dbrm-pds-name*
: An identifier of 1 to 44 characters that identifies a partitioned data set.

*ddname*
: An identifier of 1 to 8 characters that designates the name of a DD statement.

*hexadecimal-constant*
: A sequence of digits or any of the letters from A to F (uppercase or lowercase).

*hexadecimal-string*
: An X followed by a sequence of characters that begins and ends with an apostrophe. The characters between the string delimiters must be a hexadecimal number.

*ip address (or Internet address)*
: A 4 byte value that uniquely identifies a TCP/IP host within the TCP/IP network. IP addresses are usually displayed in a format called *dotted decimal*, where each byte of the IP address is displayed in decimal format with a period delimiting each number.

*location-name*
: A location identifier of 1 to 16 letters (but excluding the alphabetic extenders), digits or the underscore that identifies an instance of a data base management system. A location name must begin with a letter.

*luname*
: An SQL short identifier of 1 to 8 characters that identifies a logical unit name. An luname must begin with a letter.

*luwid*
: A fully qualified LU network name and an LUW instance number.

  The LU network name consists of an optional 8 character network ID, a period, and an 8 character network LU name. If you indicate no network ID, no period is required. The LUW instance number consists of 12 hex characters that uniquely identify the unit of work.

*member-name*
: An identifier of 1 to 8 characters that identifies either a member of a partitioned data set (MVS requires this naming convention) or a member of a data sharing group.

  A name for a member of a partitioned data set should not begin with DSN; this can sometimes conflict with DB2-provided member names. If a name beginning with DSN is specified, DB2 issues a warning message.

*package-id*
: An SQL short identifier of 1 to 8 letters, digits, or underscores that identifies a package. For packages created under DB2, a package ID is the name of the program whose precompilation produced the package's DBRM. A package ID must begin with a letter. (MVS requires this naming convention.)

A package ID should not begin with DSN; this can sometimes conflict with DB2-provided package IDs. If a package ID beginning with DSN is specified, DB2 issues a warning message.

*package-name*
A name given to the object created during the bind process of a single package. A package name consists of a location name, a collection ID, and a package ID separated by periods. An additional attribute, a version ID, allows for multiple versions of a package to have the same name.

*plan-name*
An SQL short identifier of 1 to 8 letters, digits or underscores that identifies an application plan. A plan name must begin with a letter.

A plan name should not begin with DSN; this can sometimes conflict with DB2-provided plan names. If a plan name beginning with DSN is specified, DB2 issues a warning message.

*qualifier-name*
An SQL short identifier of 1 to 8 letters, digits, or the underscore that identifies the implicit qualifier for unqualified table names, views, indexes, and aliases.

*string*
A sequence of characters that begins and ends with an apostrophe.

*subsystem-name*
An identifier that specifies the DB2 subsystem as it is known to MVS.

*table-name*
A qualified or unqualified name that designates a table. A table name can contain one or two parts, depending upon its qualification. The first part is the authorization ID that designates the owner of the table; the second part is an SQL long identifier. A period must separate each of the parts.

*table-space-name*
A short identifier that designates a table space of an identified database. If a database is not identified, a table space name specifies a table space of database DSNDB04.

*utility-id*
An identifier of 1 to 16 characters that uniquely identifies a utility process within DB2. A utility ID must begin with a letter, and the identifier can contain periods.

*version-id*
An SQL identifier of 1 to 64 letters, digits, lowercase alphabetic letters, underscores, periods, dashes, or colons that is assigned to a package when the package is created. The version ID that is assigned is taken from the version ID associated with the program being bound. Version IDs are specified for programs as a parameter of the DB2 precompile.

## Privileges and authorization IDs

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. We use the term *process* to represent any or all of those.

A process is represented to DB2 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. We use *"the privilege set of a process"* to mean 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 it is the primary authorization ID that 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 that ID.
- Secondary authorization IDs, which are optional, can hold additional privileges available to the process. A secondary authorization ID is often a Resource Access Control Facility 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 in the database.

  DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.
- An SQL authorization ID (SQL ID) holds the privileges exercised when issuing certain dynamic SQL statements. This ID plays little part in the commands described in this book.

Within DB2, a process can be represented by a primary authorization ID and possibly one or more secondary IDs. For detailed instructions on how to associate a process with one or more IDs, and how to grant privileges to those IDs, see Section 3 (Volume 1) of *DB2 Administration Guide*.

A privilege or authority is granted to, or revoked from, an identifier by executing an SQL GRANT or REVOKE statement. For the complete syntax of those statements, see Chapter 6 of *DB2 SQL Reference*.

## The bind process

The bind process establishes a relationship between an application program and its relational data. This step is necessary before you can execute your program. Currently, DB2 allows you two basic ways of binding a program: to a package, or directly to an application plan. If your application uses DRDA access to distribute data, then you must use packages.

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 application program can be run. DBRMs must be bound to a plan or package.

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. There are no restrictions to the number of DBRMs that can be included in a plan. However, packages provide a more flexible method for handling large numbers of DBRMs within a plan. As a general rule, it is suggested that the EDM pool be at least 10 times the size of the largest DBD or plan, whichever is greater. For further information, see Section 2 of *DB2 Installation Guide*.

The BIND PACKAGE subcommand allows you to bind DBRMs individually. It gives you the ability to test different versions of an application without extensive rebinding. Package binding is also the only method for binding applications at remote sites.

Even when they are bound into packages, all programs must be designated in an application plan. BIND PLAN establishes the relationship between DB2 and all DBRMs or packages in that plan. Plans can specify explicitly named DBRMs, packages, collections of packages, or a combination of these elements. The plan contains information about the designated DBRMs or packages and about the data the application program intends to use. It is stored in the DB2 catalog.

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

- **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 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 needed to access the DB2 data your program wants to process.** In selecting an access path, DB2 considers indexes, table sizes, and other factors. DB2 considers all indexes 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. For a detailed explanation of binding with DSNH CLIST, see Chapter 2. Commands. A complete description of the bind process can be found in Section 6 of *DB2 Administration Guide*. Further information on BIND can be found in “BIND PACKAGE (DSN)” on page 47 and in “BIND PLAN (DSN)” on page 52. Information about specific options for BIND PLAN and BIND PACKAGE can be found in “Options of BIND and REBIND for PLAN, PACKAGE, and TRIGGER PACKAGE” on page 57.

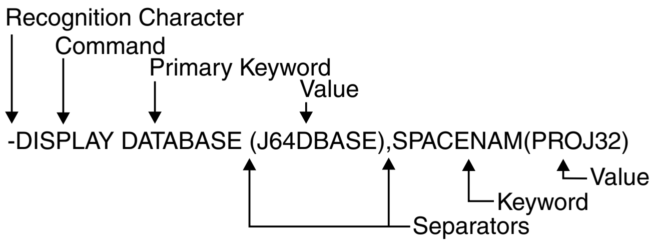
# Chapter 2. Commands

This chapter contains syntax diagrams, semantic descriptions, rules, and usage examples of commands, organized alphabetically by command name.

The tables at the beginning of this chapter summarize the commands that follow. Each table lists commands of one type, describes their functions, and refers to the page on which a complete description begins.

## DB2 command parsing

DB2 commands follow a pattern like this:



## Parts of a DB2 command

The parts of a command are:

- *Recognition character*. It is shown as a hyphen throughout this book, with the following exceptions:
  - If the command is issued from an MVS console, the recognition character must be the *command prefix*.

    # In DB2 Version 6, the command prefix can be up to eight characters. The default is '-DSN1'. However, the majority of examples in this book 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. For more information, see Section 2 of *DB2 Installation Guide*.
  - 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. For more information, see *IMS/ESA Customization Guide.*

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

- *Command name*. Command names have abbreviations, which are provided in the command descriptions in this chapter.
- *Operands*. These are combinations of keywords and parameters that can be specified for the command.
  – *Keywords* can be required or optional. They must be entered exactly as shown in the descriptions of the commands.
  – 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

The following characters have special meaning for the syntax of DB2 commands:

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 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.

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

**"** 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 means any of the following usages:

* 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)

**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*)

***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 operation is specifically permitted.

**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 are general examples of valid 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 above

Do not use more than one parameter after an equal sign, or an error condition will occur.

## Scope of commands

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

**Member** Many commands used in a data sharing environment have *member* scope because they affect only the DB2 for which they are issued. For example, a DISPLAY THREAD command displays only those threads that exist for the member 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 commands that have group scope are:

ALTER GROUPBUFFERPOOL (DB2)
BIND PACKAGE (DSN)
BIND PLAN (DSN)
DCLGEN (DSN)
DISPLAY DATABASE (DB2)
DISPLAY GROUP (DB2)
DISPLAY GROUPBUFFERPOOL (DB2)
DISPLAY UTILITY (DB2)
FREE PACKAGE (DSN)
FREE PLAN (DSN)
MODIFY irlmproc,DIAG,DELAY (MVS IRLM)
REBIND PACKAGE (DSN)
REBIND PLAN (DSN)
REBIND TRIGGER PACKAGE (DSN)
START DATABASE (DB2)
STOP DATABASE (DB2)

These commands have either group or member scope, depending on what option you specify with them:

ARCHIVE LOG (DB2)
MODIFY irlmproc,STATUS (MVS IRLM)
TERM UTILITY (DB2)

All other commands have member scope. The description of each command includes its scope. For more details on data sharing, see *DB2 Data Sharing: Planning and Administration*.

## Output from DB2 commands

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.

| ## Issuing commands 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 more information about submitting DB2 commands through IFI, see Appendix E
| (Volume 2) of DB2 Administration Guide.

## DSN subcommand parsing

The parsing of DSN subcommands conforms to standard TSO command parsing conventions. For information about TSO command parsing, see *OS/390 TSO/E Programming Services.*

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.

### Abbreviations

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. **Recommendation:** To avoid potential problems, avoid abbreviating
# keywords.

## Description of commands

The commands are divided into these categories:

- The DSN command and its subcommands
- DB2 commands
- IMS commands
- CICS attachment facility commands
- MVS IRLM commands
- TSO CLISTs

## The DSN command and its subcommands

| ***Environment:*** DSN is the DB2 command processor and executes as a TSO
| command processor. All of its 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.

| *Table 1 (Page 1 of 2). DSN command and subcommands*

| DSN command or Subcommand | Function | Refer to page |
|---|---|---|
| BIND | Builds an application package or plan | 47, 52 |

Table 1 (Page 2 of 2). DSN command and subcommands

| DSN command or Subcommand | Function | Refer to page |
|---|---|---|
| DB2 commands | Execute a DB2 command | Table 2 on page 22 |
| DCLGEN | (DECLARATIONS GENERATOR) Produces declarations for tables or views | 94 |
| DSN | Starts a DSN session | 183 |
| END | Ends the DSN session | 224 |
| FREE | Deletes an application package or plan | 225, 228 |
| REBIND | Updates an application package or plan | 245, 249 |
| REBIND TRIGGER PACKAGE | Updates an application trigger package | 253 |
| RUN | Executes an application program | 267 |
| SPUFI | Executes the SQL Processor Using File Input | 275 |
| * | A comment | 183 |

## DB2 commands

***Environment:*** The command START DB2 can be issued only from an MVS console. All other DB2 commands can be issued from:

- An MVS console or MVS 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 issued from an MVS console are not associated with any secondary authorization IDs.

***Extended MCS Consoles:*** The extended MCS console feature of MVS lets an MVS 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. For more information on extended MCS consoles, see Section 4 (Volume 1) of DB2 Administration Guide, and *OS/390 MVS Planning: Operations*.

Table 2 (Page 1 of 3). DB2 commands

| DB2 command | Function | Refer to page |
|---|---|---|
| -ALTER BUFFERPOOL | Alters attributes for the buffer pools | 27 |
| -ALTER GROUPBUFFERPOOL | Alters attributes for the group buffer pools | 34 |
| -ALTER UTILITY | Alters parameter values of the REORG utility | 38 |

*Table 2 (Page 2 of 3). DB2 commands*

| DB2 command | Function | Refer to page |
|---|---|---|
| -ARCHIVE LOG | Enables a site to close a current active log and open the next available log data set | 41 |
| -CANCEL THREAD | Cancels processing for specific local or distributed threads | 88 |
| -DISPLAY ARCHIVE | Displays information about archive log processing | 105 |
| -DISPLAY BUFFERPOOL | Displays information about the buffer pools | 107 |
| -DISPLAY DATABASE | Displays status information about DB2 databases | 116 |
| -DISPLAY FUNCTION SPECIFIC | Displays statistics about external user-defined functions | 129 |
| -DISPLAY GROUP | Displays information about the data sharing group to which a DB2 subsystem belongs | 133 |
| -DISPLAY GROUPBUFFERPOOL | Displays status information about DB2 group buffer pools | 138 |
| -DISPLAY LOCATION | Displays status information about distributed threads | 151 |
| -DISPLAY LOG | Displays log information and status of the offload task | 155 |
| -DISPLAYPROCEDURE | Displays status information about stored procedures | 157 |
| -DISPLAY RLIMIT | Displays status information about the resource limit facility (governor) | 161 |
| -DISPLAY THREAD | Displays information about DB2 threads | 162 |
| -DISPLAY TRACE | Displays information about DB2 traces | 174 |
| -DISPLAY UTILITY | Displays status information about a DB2 utility | 179 |
| -MODIFY TRACE | Changes the IFCIDs (trace events) associated with a particular active trace | 242 |
| -RECOVER BSDS | Reestablishes dual bootstrap data sets | 256 |
| -RECOVER INDOUBT | Recovers threads left indoubt | 257 |
| -RECOVERPOSTPONED | Completes back-out processing for units of recovery left incomplete during an earlier restart | 260 |
| -RESET GENERICLU | Purges information stored by VTAM in the coupling facility | 262 |
| -RESET INDOUBT | Purges information displayed in the indoubt thread report generated by the -DISPLAY THREAD command | 264 |
| -SET ARCHIVE | Controls the allocation of tape units and the deallocation time of the tape units for archive log processing | 270 |
| -SET LOG | Modifies the checkpoint frequency | 273 |
| -START DATABASE | Makes the specified database available for use | 278 |

Table 2 (Page 3 of 3). DB2 commands

| DB2 command | Function | Refer to page |
|---|---|---|
| -START DB2 | Initializes the DB2 subsystem (can be issued only from an MVS console) | 285 |
| -START DDF | Starts the distributed data facility | 288 |
| -START FUNCTION SPECIFIC | Activates an external function that is stopped | 289 |
| -START PROCEDURE | Activates the definition of stopped or cached stored procedures | 295 |
| -START RLIMIT | Starts the resource limit facility (governor) | 298 |
| -START TRACE | Initiates DB2 trace activity | 300 |
| -STOP DATABASE | Makes specified databases unavailable for applications | 311 |
| -STOP DB2 | Stops the DB2 subsystem | 317 |
| -STOP DDF | Stops the distributed data facility | 319 |
| -STOP FUNCTION SPECIFIC | Stops the acceptance of SQL statements for specified functions | 322 |
| -STOP PROCEDURE | Stops the acceptance of SQL CALL statements for stored procedures | 327 |
| -STOP RLIMIT | Stops the resource limit facility (governor) | 330 |
| -STOP TRACE | Stops trace activity | 331 |
| -TERM UTILITY | Terminates execution of a utility | 336 |

***Completion Messages:*** Message DSN9022I indicates the normal end of DB2 command processing; DSN9023I indicates the abnormal end of DB2 command processing.

## IMS commands

***Environment:*** Each IMS command can be issued from an IMS terminal.

Table 3. IMS commands

| IMS command | Function | Refer to page |
|---|---|---|
| /CHANGE | Resets an indoubt recovery unit | 92 |
| /DISPLAY | Displays the status of the connection between IMS and the specified subsystem (DB2), or displays the outstanding recovery units associated with the specified subsystem (DB2) | 102 |
| /SSR | Allows the IMS operator to enter an external subsystem (DB2) command | 276 |
| /START | Makes available the connection between IMS and the specified external subsystem (DB2) | 277 |
| /STOP | Prevents application programs from accessing the external subsystem's (DB2's) resources | 310 |
| /TRACE | Allows users to direct and control IMS tracing activities | 339 |

## CICS attachment facility commands

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

*Table 4. CICS attachment facility commands*

| CICS attachment facility commands | Function | Refer to page |
|---|---|---|
| DSNC | Allows you to enter DB2 commands from CICS | 186 |
| DSNC DISCONNECT | Disconnects threads | 187 |
| DSNC DISPLAY | Displays information on CICS transactions | 189 |
| DSNC MODIFY | Modifies the ERRDEST entry in the resource control table (RCT), or modifies the maximum active thread value associated with a given transaction or group name | 193 |
| DSNC STOP | Stops the CICS attachment facility | 195 |
| DSNC STRT | Starts the CICS attachment facility | 196 |

## MVS IRLM commands

***Environment:*** Each MVS IRLM command can be issued from an MVS console.

| *Table 5. MVS commands affecting the IRLM*

| MVS command | Function | Refer to page |
|---|---|---|
| MODIFY irlmproc, ABEND | Abends IRLM | 230 |
| \| MODIFY irlmproc, DIAG,DELAY | Initiates diagnostic dumps for IRLM subsystems | 232 |
| MODIFY irlmproc, SET | Dynamically sets the maximum CSA or the number of trace buffers allowed for IRLM | 234 |
| MODIFY irlmproc, STATUS | Displays IRLM status | 236 |
| START irlmproc | Starts an IRLM component with an installation-supplied procedure | 291 |
| STOP irlmproc | Shuts down IRLM normally | 325 |
| TRACE CT | Starts, stops, or modifies IRLM tracing | 341 |

## TSO CLISTs

*Table 6 (Page 1 of 2). TSO CLISTs*

| CLIST | Function | Refer to page |
|---|---|---|
| DSNH | Prepares a program for execution, and executes it if it runs under TSO. Runs under TSO in foreground or background. | 198 |

*Table 6 (Page 2 of 2). TSO CLISTs*

| **CLIST** | **Function** | **Refer to page** |
|---|---|---|
| DSNU | Generates JCL to execute DB2 utility jobs. Can be executed directly or by using DB2I. For details on this command procedure, see *DB2 Utility Guide and Reference*. | |

## -ALTER BUFFERPOOL (DB2)

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

**Abbreviation**: -ALT BPOOL

## Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
  ►►──ALTER BUFFERPOOL──(──bpname──)──┬─────────────────┬──┬──────────────────────────┬──────────────►
#                                     └─VPSIZE(integer)─┘  └─VPTYPE──┬─PRIMARY───┬─┘
|                                                                    └─DATASPACE─┘

  ►──┬─────────────────┬──┬─────────────────┬──┬──────────────────┬──┬───────────────────┬──────────►
     └─HPSIZE(integer)─┘  └─VPSEQT(integer)─┘  └─VPPSEQT(integer)─┘  └─VPXPSEQT(integer)─┘

  ►──┬─────────────────┬──┬───────────────┬──┬─────────────────────────────┬──┬──────────────────────┬──►
|    └─HPSEQT(integer)─┘  └─DWQT(integer)─┘  └─VDWQT(integer1,integer2)─┘  └─CASTOUT(──┬─YES─┬──)─┘
                                                                                       └─NO──┘

  ►──┬──────────────────────┬─────────────────────────────────────────────────────────────────────►◄
#    └─PGSTEAL──┬─LRU──┬────┘
|               └─FIFO─┘
```

## Option descriptions

**(*bpname*)**
Names the buffer pool to alter.

- 4-KB page buffer pools are named BP0, BP1, ..., BP49.
- | 8-KB page buffer pools are named BP8K0, BP8K1, ..., BP8K9.
- | 16-KB page buffer pools are named BP16K0, BP16K1, ..., BP16K9.
- 32-KB page buffer pools are named BP32K, BP32K1, ..., BP32K9.

**VPSIZE (***integer***)**
Changes the virtual buffer pool size.

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

*integer* can range from 0 to 400000 for 4-KB page buffer pools other than BP0.
| For BP0, the minimum value is 56. For 8-KB page buffer pools, the range is
| from 0 to 200000. For 16-KB page buffer pools, the range is from 0 to 100000. For 32-KB page buffer pools, the range is from 0 to 50000.

DB2 limits the total VPSIZE for all buffer pools to 1.6 GB. However, the amount of available real and virtual storage can further limit the amount of buffer pool storage DB2 can acquire.

If you specify VPSIZE as 0 for an active buffer pool (other than BP0), 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.

| **VPTYPE**
| Specifies the type of virtual buffer pool to be allocated. Changes to the
| VPTYPE value apply after DB2 reallocates the buffer pools.

| If the specified virtual pool is currently allocated, VPTYPE and VPSIZE are
| mutually exclusive. In order to change both the type and size specifications,
| follow the procedure in 31.

# When DB2 is installed, the **default** is **VPTYPE (PRIMARY)**. However, when
# you issue the ALTER BUFFERPOOL command, you must explicitly specify
# either PRIMARY or DATASPACE when the VPTYPE option is used.

| **PRIMARY**
| The virtual buffer pool is to be allocated in the DB2 database services
| address space.

| **DATASPACE**
| The virtual buffer pool is to be allocated in one or more DB2-associated
| data spaces.

**HPSIZE (***integer***)**
Changes the hiperpool size. If the buffer pool is active at the time the command is issued, the hiperpool is created, expanded, contracted, or deleted depending on the new HPSIZE value.

*integer* specifies the number of buffers for a hiperpool. For 4-KB-page
| hiperpools, the acceptable values range from 0 to 2097152. For 8-KB-page
| hiperpools, the range is from 0 to 1048576. For 16-KB-page hiperpools, the
| range is from 0 to 524288. For 32-KB-page hiperpools, the range is from 0 to
| 262144.

DB2 limits the total HPSIZE for all buffer pools to 8 GB. However, the amount of available expanded or real storage can further limit the amount of buffer pool storage DB2 can acquire. A value of 0 can be specified for all buffer pools.

**VPSEQT (***integer***)**
Changes the sequential steal threshold for the virtual buffer pool.

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

When VPSEQT=0, sequentially accessed pages are not kept in the virtual 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 virtual buffer pool. This threshold determines how much of the virtual buffer pool is used for parallel processing operations.

*integer* specifies the parallel sequential threshold for the virtual buffer pool. It 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 virtual buffer pool. This threshold determines the portion of the virtual buffer pool that is used for processing queries that originate on other members of the data sharing group. It is valid and effective only when DB2 is on a data sharing mode; it is ignored when DB2 is not on a data sharing mode.

*integer* specifies the assisting parallel sequential threshold for the virtual buffer pool. It 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 bufferpool cannot be used to assist another DB2 with parallel processing.

**HPSEQT (***integer***)**
Changes the hiperpool sequential steal threshold.

*integer* specifies the sequential steal threshold for the hiperpool. It is expressed as a percentage of the total hiperpool size, and valid values range from 0 to 100. This threshold affects the allocation of hiperpool buffers when casting out sequentially accessed pages from the virtual pool. If the number of buffers in the hiperpool containing sequentially accessed pages exceeds the threshold, the buffer allocation for a sequentially accessed page attempts to reuse one of these buffers rather than a buffer containing a non-sequentially accessed page. The initial default is 80.

When HPSEQT=0, sequentially accessed pages are not cast out to the hiperpool.

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

**DWQT (*integer*)**
Changes the buffer pool's deferred write threshold.

*integer* specifies the deferred write threshold for the virtual buffer pool. It is expressed as a percentage of the total virtual buffer pool size, and valid values range from 0 to 90. This threshold determines when deferred writes begin, based on the number of unavailable (non-stealable) buffers. When the count of non-stealable buffers exceeds the threshold, deferred writes begin. The initial default value is 50.

| **VDWQT (*integer1*,*integer2*)**
| *integer1* specifies the vertical deferred write threshold for the virtual buffer pool. It is expressed as a percentage of the total virtual 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 10. 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 virtual buffer pool. It is expressed as an absolute number of buffers. You can use *integer2* when you want a relatively low threshold value for a large virtual 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, then the *integer1* value of 0 is used as the threshold.

**CASTOUT**
Changes the CASTOUT attribute of the hiperspaces used to back the hiperpool.

# When DB2 is installed, the **default** is **CASTOUT (YES)** for all hiperpools. However, when you issue the ALTER BUFFERPOOL command, you must explicitly specify either YES or NO when the CASTOUT option is used.

**(YES)**
Allows MVS to discard data cached in the hiperpool when a shortage of expanded storage arises. When data is discarded, hiperspace backing expanded storage pages is released.

**(NO)**
# Directs MVS to assign a high priority to keeping the data cached in the hiperpool. Because this could severely limit the availability of expanded storage to other processes on the system, specify CASTOUT(NO) only for buffer pools associated with databases for which response time is critical.

| **PGSTEAL**
| Specifies the page stealing algorithm DB2 uses for the virtual buffer pool.

# When DB2 is installed, the **default** is **PGSTEAL (LRU)**. However, when you

# issue the ALTER BUFFERPOOL command, you must explicitly specify either
# LRU or FIFO when the PGSTEAL option is used.

| **LRU**
| Specifies the virtual buffer pool buffers should be managed using the least
| recently used (LRU) algorithm.

| **FIFO**
| Specifies the virtual buffer pool buffers should be managed using the first in
| first out (FIFO) algorithm.

## Usage Notes

The following sections contain 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.

***Insufficient virtual storage:*** If insufficient virtual storage is detected while expanding a virtual buffer pool or while creating or expanding a hiperpool, DB2 issues an error message, and the process terminates, leaving the virtual buffer pool or hiperpool with a smaller size than was requested. Similarly, DB2 issues an error message if it is unable to create a hiperspace.

***Contracting an active virtual buffer pool:*** If you use ALTER BUFFERPOOL to contract the size of an active virtual buffer pool, DB2 contracts the pool by marking active buffers as "to be deleted," which means 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. Once 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 hiperpool attributes:*** If you use ALTER BUFFERPOOL to alter the CASTOUT attribute of an already active hiperpool, DB2 marks the hiperpool as temporarily unavailable, so transactions are executed using the buffer pool without its backing hiperpool. The hiperspaces backing the hiperpool are deleted and re-created with the new CASTOUT attribute. All pages that were cached in the hiperpool are purged before the hiperpool is activated again.

When DB2 is restarted, if a hiperpool could not be created, DB2 issues a warning message indicating the reason. Under this condition, DB2 comes up with only the virtual bufferpool created. Use ALTER BUFFERPOOL to set the HPSIZE value to 0 to avoid any further messages.

***Altering attributes stored in the BSDS:*** The virtual bufferpool and hiperpool attributes that are stored in the BSDS cannot be changed off line.

***Altering the VPTYPE for a virtual buffer pool:*** If the virtual pool is currently allocated and you change the VPTYPE attribute, the VPTYPE specification becomes effective only at the next allocation of the virtual buffer pool. To alter the VPTYPE attribute for a buffer pool and have it take effect, use the following procedure:

1. Issue the ALTER BUFFERPOOL command, specifying a new value for
# VPTYPE. If the virtual pool is currently allocated, the new specification takes
# effect the next time the buffer pool is allocated. If the buffer pool is currently
# allocated, proceed to step 2.
2. If the target buffer pool is not BP0:
   a. Issue the following command to deallocate the buffer pool:
      ```
      -ALTER BUFFERPOOL(bpname) VPSIZE(0)
      ```
   b. Issue the following command to redefine the buffer pool with the new type and new size:
      ```
      -ALTER BUFFERPOOL(bpname) VPSIZE(newsize) VPTYPE(newtype)
      ```
3. If the target buffer pool is BP0:
   a. Issue the following command to specify the new type:
      ```
      -ALTER BUFFERPOOL(0) VPTYPE(newtype)
      ```
#  b. Deallocate BP0 by either issuing the STOP DATABASE command to stop
#  all table spaces and indexes that are using BP0, or stop DB2.
   c. Depending on the action taken in step 3b, issue START DATABASE commands to start the stopped table spaces and indexes, or restart DB2.
   d. Optionally, issue the following command to specify the new size.
      ```
      -ALTER BUFFERPOOL(BP0) VPSIZE(newsize)
      ```
#  If you skip this step, the previous size is used.

| ***Converting an active buffer pool to use a data space:***

# 1. Delete the active buffer pool, BPx.
# 2. Stop all data sets that are using BPx.
# 3. Issue a new ALTER BUFFERPOOL command with VPTYPE(D).
# 4. Start all previous stopped data sets.

***Relating VPPSEQT and VPXSEQT:***The Table 7 on page 33 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 virtual buffer pool that can be used to assist another DB2 with parallel processing. In addition, VPPSEQT is affected by changing VPSIZE and VPSEQT; therefore, VPXPSEQT is also affected by VPSIZE
# and VPSEQT. For more information on the relationships of the various thresholds
# and possible configurations, see Chapter 7 of *DB2 Data Sharing: Planning and*
# *Administration*.

Table 7. Relationship between VPPSEQT and VPXPSEQT

| If VPPSEQT is ... | and VPXPSEQT is ... | the percentage of the virtual buffer pool available to assist Sysplex query parallelism is... |
|---|---|---|
| 50 | 50 | 25 |
| 50 | 100 | 50 |
| 100 | 50 | 50 |
| anything | 0 | 0 |
| 0 | anything | 0 |

## Examples

***Example 1:*** Set the virtual buffer pool and hiperpool for BP0 to 1000 and 10000 buffers, respectively.

```
-ALTER BUFFERPOOL(BP0) VPSIZE(1000) HPSIZE(10000)
```

***Example 2:*** Set the sequential steal threshold of the virtual buffer pool for BP0 to 75 percent of the virtual pool size, while disabling caching of sequentially accessed pages in the hiperpool.

```
-ALTER BUFFERPOOL(BP0) VPSEQT(75) HPSEQT(0)
```

***Example 3:*** Set the hiperpool size for BP4 to 10000 buffers and explicitly specify that cached data in the hiperpool for BP4 can be discarded.

```
-ALTER BUFFERPOOL(BP4) HPSIZE(10000) CASTOUT(YES)
```

# ***Example 4:*** Delete BP1. Be very careful when using this option because when you
# specify a 0 size for an active buffer pool, DB2 quiesces all current database access
# and fails all subsequent page set open requests.

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

## -ALTER GROUPBUFFERPOOL (DB2)

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

**Abbreviation**: -ALT GBPOOL

### Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following authorities:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
►►──ALTER GROUPBUFFERPOOL──(──┬─gbpname────────┬──)──┬────────────────────────┬──────────────►
                              └─structure-name─┘     └─GBPCACHE(──┬─YES─┬──)─┘
                                                                  └─NO──┘

►──┬───────────────────────┬──┬──────────────┬──┬─────────────────┬──┬──────────────────┬──►
   └─AUTOREC(──┬─YES─┬──)─┘  └─RATIO(ratio)─┘  └─CLASST(integer)─┘  └─GBPOOLT(integer)─┘
               └─NO──┘

►──┬───────────────────┬───────────────────────────────────────────────────────────────►◄
   └─GBPCHKPT(integer)─┘
```

### Option descriptions

**(*gbpname*)**
Names the DB2 group buffer pool.

- 4-KB group buffer pools are named GBP0, GBP1, ... , GBP49.
- 8-KB group buffer pools are named GBP8K0, GBP8K1, ... , GBP8K9.
- 16-KB group buffer pools are named GBP16K0, GBP16K1, ... , GBP16K9.
- 32-KB group buffer pools are named GBP32K, GBP32K1, ... , 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*.

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

| **(YES)**
| *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. For
| more information, see Table 8.

| *Table 8. Precedence of a no-data-caching specification*

| Group buffer pool specification | Page set specification | Attribute that takes precedence |
|---|---|---|
| GBPCACHE(NO) | GBPCACHE CHANGED<br>GBPCACHE ALL | GBPCACHE(NO) |
| GBPCACHE(YES) | GBPCACHE NONE | GBPCACHE NONE |

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

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

**(YES)**
# Indicates that DB2 is 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. Enter 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 desired ratio in the group buffer pool of the number of directory entries to the number of data pages; that is, how many directory entries there are 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. In numbers greater than 25, any digits after the decimal point are ignored; for example, 25.98 is treated as 25. The default 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.

**CLASST (*integer*)**
Changes the threshold at which class castout is started. It is expressed as a percentage of the size of the number of data entries; *integer* can range from 0 to 90. The default is 10.

As an example, CLASST(5) starts class castout when the number of pages in that class equals 5% of the group buffer pool page capacity.

**GBPOOLT (*integer*)**
Changes the threshold at which data in the group buffer pool is cast out to DASD. It is expressed as a percentage of the number of data entries; *integer* can range from 0 to 90. The default is 50.

As an 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 explic-
# itly specified for the GBPCHKPT option, the default value is 8 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 command does not change any option that is not explicitly specified; the default is to leave the value unchanged. When the command is first issued for a group buffer pool or a structure, the option defaults are as follows:

| Option | Value |
|---|---|
| GBPCACHE | YES |
| RATIO | 5 |
| CLASST | 10 (%) |
| GBPOOLT | 50 (%) |
| GBPCHKPT | 8 (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. Refer to Table 9 for more information.

*Table 9. Changing group buffer pool attributes*

| Keyword | New value takes effect | Applicable if GBPCACHE(NO)? |
|---|---|---|
| GBPCACHE | at next allocation 1 | N/A |
| AUTOREC | immediately | No |
| RATIO | at next allocation 2 | No 3 |
| CLASST | immediately | No 3 |
| GBPOOLT | immediately | No 3 |
| GBPCHKPT | immediately | No 3 |

**Notes:**

1. You can use the MVS 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 MVS 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 1:*** For group buffer pool 0, change the ratio of directory entries to data pages to 1 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 2:*** For group buffer pool 2, change the class castout threshold to 5% and the group buffer pool castout threshold to 30%. The new values take effect immediately.

```
-DB1G ALTER GROUPBUFFERPOOL (GBP2) CLASST(5) GBPOOLT(30)
```

***Example 3:*** Assume that the DB2 group name is DSNCAT. For group buffer pool 3, change the class castout threshold to 5%. The new value takes effect immediately. Because the group name is DSNCAT, the coupling facility structure name is DSNCAT_GBP3. Also, in the event of a structure failure, 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(5) AUTOREC(YES)
```

***Example 4:*** For group buffer pool 32K, change the GBP checkpoint frequency to 5 minutes. The new value takes effect immediately. Here, with AUTOREC(NO) specified, you are in effect taking control of the recovery process rather than DB2 in the event of a structure failure. You might choose to do this to determine what pagesets or partitions are in a GRECP status or that have pages on the logical page list and before entering the START DATABASE command to enable DB2 to recover the data with the options you specify.

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

## -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. Specifically, this command changes the values of DEADLINE, MAXRO, LONGLOG, and DELAY. For more information about those parameters and the REORG utility, see *DB2 Utility Guide and Reference*.

REORG can be altered only from the DB2 on which it is running.

**Abbreviation**: -ALT UTIL

### Environment

This command can be issued from an MVS 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, the primary or some secondary authorization ID of the process must be the ID that originally submitted the utility job, or the privilege set of the process must include one of the following authorities:

- DBMAINT, DBCTRL, or DBADM authority
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands that are issued from an MVS console are not associated with any 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

```
►►──ALTER──UTILITY──(──utility-id──)──REORG──┬────────────────────────────────┬──────────────►
                                             └─DEADLINE(──┬─NONE──────┬──)─┘
                                                          └─timestamp─┘

►──┬──────────────────────────┬──┬─────────────────────────────┬──┬──────────────────────┬──►◄
   └─MAXRO(──┬─integer─┬──)─┘  └─LONGLOG(──┬─CONTINUE─┬──)─┘  └─DELAY(───integer───)─┘
             └─DEFER───┘                    ├─TERM─────┤
                                            └─DRAIN────┘
```

### Option descriptions

**(***utility-id***)**
: Is the utility identifier, or the UID parameter, used when creating the utility job step.

  This job must execute 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*. For the control file name that is associated with each utility, see *DB2 Utility Guide and Reference.*

If *utility-id* was created by default by the EXEC statement that executed DSNUTLIB, then this token has the form *userid.jobname*.

**DEADLINE**
Specifies the deadline by which the user wants the switch phase of reorganization to start. If DB2 estimates that the switch phase will not start by the deadline, DB2 terminates reorganization. The default is the most recently specified value of DEADLINE.

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. For more information on the format for specifying a timestamp, see the discussion of data types in *DB2 SQL Reference*.

**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.

**(***integer***)**
Is the number of *seconds*. The default is the most recently specified value of MAXRO.

**(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 most recently specified value of LONGLOG.

**(CONTINUE)**
Specifies that DB2 continues performing reorganization.

**(TERM)**
Specifies that DB2 terminates reorganization after the delay.

**(DRAIN)**
Specifies that DB2 drains the write claim class after the delay. 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 REORG sends the LONGLOG message to the console and the time when REORG performs the action specified by the LONGLOG parameter.

*integer* is the number of *seconds*. The value must be nonnegative. The default is the most recently specified value of DELAY.

## Usage note

# REORG can be altered only from the DB2 subsystem on which it is running.

## Example

The following example 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
```

# In this example:

| • MAXRO(240) changes the maximum tolerable time for the last iteration of log processing to 240 seconds (4 minutes).
- LONGLOG DRAIN changes the action that DB2 performs (if reorganization's reading of the log is not catching up to applications' writing of the log quickly enough) to draining of the write claim class.
- DELAY was not specified and therefore, the example does not change the existing delay between sending of the LONGLOG message to the console and performing the action specified by LONGLOG.
- DEADLINE was not specified and the example does not change the existing deadline (if any) of the last iteration of log processing.

## -ARCHIVE LOG (DB2)

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. Once 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.

For further information regarding the ARCHIVE LOG command, see Section 4 (Volume 1) of *DB2 Administration Guide*.

**Abbreviation**: -ARC LOG

## Environment

This command can be issued from an MVS 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 MVS 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, the privilege set of this process must include one of the following:

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

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
                ┌─SCOPE(MEMBER)─────────────────────────────────────────┐
 ►►──ARCHIVE LOG─┼───────────────────────────────────────────────────────┼──────────────────────────►◄
                ├─SCOPE(GROUP)──────────────────────────────────────────┤
                ├─MODE(QUIESCE)──┬────────────┬──┬──────────────────────┬┤
                │                └─TIME(nnn)──┘  │        ┌─NO──┐       ││
                │                                └─WAIT(──┴─YES─┴──)────┘│
                └─CANCEL OFFLOAD────────────────────────────────────────┘
```

## Option descriptions

**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.

# **<u>(MEMBER)</u>**

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.

**MODE(QUIESCE)**

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

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

If there are no indoubt URs left 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 up.

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

If you do not use the TIME option to specify the quiesce time period, the default is the value specified in the field QUIESCE PERIOD of installation panel DSNTIPA.

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

**TIME(*nnn*)**

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

The **default** is the period specified in the field QUIESCE PERIOD of installation panel DSNTIPA. See Section 2 of *DB2 Installation Guide* for more information on this field.

*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 time-out 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 return control when the quiesce processing begins.

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

**(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.

**(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 enter additional DB2 commands once the ARCHIVE LOG command returns control to you.

**CANCEL OFFLOAD**
Cancels any offloading currently in progress and restarts the offload process, beginning with the oldest active log data set that has not been offloaded and proceeding through all active log data sets that need offloading. 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. See Section 4 (Volume 1) of *DB2 Administration Guide* for additional information on backup and recovery.

***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 execute 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. We recommend using ARCHIVE LOG CANCEL OFFLOAD only if the offload task no longer seems to be functioning or if you want to restart a previous offload attempt that failed.

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

***Interaction with -DISPLAY THREAD:*** By issuing message DSNV400I, the command DISPLAY THREAD indicates 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, SCOPE(GROUP) is assumed.

## Examples

***Example 1:*** Truncate the current active log data sets and initiate an asynchronous job to offload the truncated data sets. No quiesce processing occurs.

```
-ARCHIVE LOG
```

***Example 2:*** Initiate 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.

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.

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

***Example 3:*** Initiate 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.

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.

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

***Example 4:*** In a data sharing environment, initiate 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 5:*** In a data sharing environment, truncate 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 6:*** In a data sharing environment, truncate 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)
```

## 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. For more information on using BIND PACKAGE, see Section 6 of *DB2 Application Programming and SQL Guide*.

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

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

Table 10 summarizes the authorization required to run BIND PACKAGE, depending on the bind options you specify, and in the case of the ADD option, the value of field BIND NEW PACKAGE.

*Table 10 (Page 1 of 2). 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 | The primary authorization ID (default owner) must have one of the following to add a new package or new version of an existing package to a collection:<br>• The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections<br>• SYSADM or SYSCTRL authority |
| | BIND | The primary authorization ID (default owner) must have one of the following to add a new package or a new version of an existing package to a collection:<br>• The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections<br>• SYSADM or SYSCTRL authority<br>• PACKADM authority on the collection or on all collections<br>• The BIND package privilege (can only add a new version of an existing package) |

Table 10 (Page 2 of 2). Summary of privileges needed for BIND PACKAGE options

| Bind option | Installation panel field BIND NEW PACKAGE | Authorization required to run BIND PACKAGE |
|---|---|---|
| ADD, specifying an OWNER other than the primary authorization ID (1) | BINDADD | If the binder does not have SYSADM or SYSCTRL authority, the authorization ID of the OWNER must have one of the following to add a new package or new version of an existing package to a collection: • The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections • SYSADM or SYSCTRL authority |
|  | BIND | If the binder does not have SYSADM or SYSCTRL authority, the authorization ID of the OWNER must have one of the following to add a new package or new version of an existing package to a collection: • The BINDADD system privilege and either the CREATE IN privilege or PACKADM authority on the collection or on all collections • SYSADM or SYSCTRL authority • PACKADM authority on the collection or on all collections • The BIND package privilege (can only add a new version of an existing package) |
| REPLACE, using the default owner or primary authorization ID | BINDADD or BIND | Primary authorization ID must have one of the following: • Ownership of the package • BIND privilege on the package • PACKADM authority on the collection or on all collections • SYSADM or SYSCTRL authority |
| REPLACE, specifying an OWNER other than the primary authorization ID (1) | BINDADD or BIND | If the binder does not have SYSADM or SYSCTRL authority, the authorization ID of the OWNER must have one of the following: • BIND privilege on the package • PACKADM authority on the collection or on all collections • SYSADM or SYSCTRL authority |
| COPY | BINDADD or BIND | The primary or secondary authorization ID of the binder or OWNER must have one of the following on the package being copied: • Ownership of the package • COPY privilege on the package • BINDAGENT privilege from the owner of the package • PACKADM authority on the collection or on all collections • SYSADM or SYSCTRL authority |

**Notes:**

1. If any of the authorization IDs of the process has the 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, 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.

| For additional information on the required authorization to execute BIND PACKAGE see Section 5 (Volume 2) of *DB2 Administration Guide*.

## Syntax

```
►►──BIND PACKAGE──(──┬─────────────────┬──collection-id──)──┬─────────────────────────────┬──────►
                     └─location-name.──┘                    └─OWNER(authorization-id)─────┘

►──┬──────────────────────────────┬──────────────────────────────────────────────────────────────►
   └─QUALIFIER(qualifier-name)────┘

►──┬─MEMBER(dbrm-member-name)──┬───────────────────────────┬──────────────────────────────────┬──►
   │                           └─LIBRARY(dbrm-pds-name)────┘                                  │
   └─COPY(collection-id.package-id)──┬─────────────────────────┬──┬─────────────────────────┬─┘
                                     └─COPYVER(version-id)─────┘  │           ┌─COMPOSITE─┐ │
                                                                  └─OPTIONS(──┴─COMMAND───┴─)─┘

►──┬──────────────────────┬──┬────────────────────────────────────────────────────┬──────────────►
   ├─DEFER(PREPARE)───────┤  └─ACTION──┬─(REPLACE)──┬───────────────────────┬──┬──┘
   └─NODEFER(PREPARE)─────┘            │            └─REPLVER(version-id)───┘  │
                                       └─(ADD)─────────────────────────────────┘

|  ►──┬───────────────────────────────┬──┬──────────────────────────────────┬──┬──────────────────────┬──►
      │               ┌─YES─┐         │  └─DBPROTOCOL──(──┬─DRDA────┬──)───┘  │         ┌─1───┐      │
      └─CURRENTDATA(──┴─NO──┴──)──────┘                   └─PRIVATE─┘         └─DEGREE(──┴─ANY─┴──)──┘

►──┬─────────────────────────────────────┬───────────────────────────────────────────────────────►
   └─DYNAMICRULES(──┬─RUN────────┬──)────┘
|                   ├─BIND───────┤
|                   ├─DEFINEBIND─┤
|                   ├─DEFINERUN──┤
|                   ├─INVOKEBIND─┤
|                   └─INVOKERUN──┘

►──┬─────────────────────────────────────────────────────────────────────────────────────────┬───►
   │ ┌─ENABLE(*)──────────────────────────────────┐                                          │
   └─┴─┬─ENABLE──┬──(───▼─┬─BATCH─────┬──)────────┴──▼─┬──────────────────────────────────────┬┘
       └─DISABLE─┘        ├─DLIBATCH──┤                │              ┌─,───────────────┐     │
                          ├─DB2CALL───┤                ├─DLIBATCH(───▼─connection-name─┴──)─┤
                          ├─CICS──────┤                │          ┌─,──────┐                 │
                          ├─IMS───────┤                ├─CICS(───▼─applid─┴──)──────────────┤
                          ├─IMSBMP────┤                │            ┌─,─────┐               │
                          ├─IMSMPP────┤                ├─IMSBMP(───▼─imsid─┴──)─────────────┤
                          ├─REMOTE────┤                │            ┌─,─────┐               │
                          └─RRSAF─────┘                ├─IMSMPP(───▼─imsid─┴──)─────────────┤
                                                       │            ┌─,─────────────────┐   │
                                                       └─REMOTE(───▼─┬─location-name─┬─┴──)─┘
                                                                      └─<luname>──────┘

►──┬──────────────────────────┬──┬─────────────────────┬──┬──────────────────────────────┬──┬────────────────────────────┬──►
#  │           ┌─NO──┐        │  │        ┌─I─┐        │  └─IMMEDWRITE(──┬─NO──┬──)─────┘  └─ISOLATION(──┬─RR─┬──)──────┘
#  └─EXPLAIN(──┴─YES─┴──)─────┘  └─FLAG(──┼─W─┼──)─────┘                 ├─PH1─┤                        ├─RS─┤
#                                         ├─E─┤                          └─YES─┘                        ├─CS─┤
                                          └─C─┘                                                         ├─UR─┤
                                                                                                        └─NC─┘

                                     ┌─NOREOPT(VARS)─┐
►──┬──────────────────────────────┬──┤               ├──┬──────────────────────────────┬─────────►
|  │               ┌─NO──┐        │  └─REOPT(VARS)───┘  └─OPTHINT──(──'hint-id'──)─────┘
   └─KEEPDYNAMIC(──┴─YES─┴──)─────┘

►──┬───────────────────────────────────┬──┬──────────────────────────────┬──┬──────────────────────────────────┬──►
|  │        ┌─,───────────────┐        │  └─RELEASE(──┬─COMMIT─────┬──)──┘  │           ┌─NOPACKAGE─┐          │
|  └─PATH(──▼─┬─schema-name─┬─┴──)────┘               └─DEALLOCATE─┘        └─SQLERROR(──┴─CONTINUE──┴──)──────┘
              └─USER────────┘

►──┬────────────────────────────┬────────────────────────────────────────────────────────────────►◄
   │            ┌─RUN──┐        │
   └─VALIDATE(──┴─BIND─┴──)─────┘
```

## Option descriptions

For descriptions of the options shown in the syntax diagram, see “Options of BIND and REBIND for PLAN, PACKAGE, and TRIGGER PACKAGE” on page 57.

## Examples

***Example 1:*** Replace version APRIL_VERSION of package TEST.DSN8BC61 at local location USIBMSTODB22 with another version of the package. The new version (or it could be the same) is in the DBRM DSN8BC61. 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 (DSN8BC61) -
  ACTION (REPLACE) REPLVER (APRIL_VERSION) -
  QUALIFIER (PRODUCTN) -
  ENABLE (BATCH, CICS) CICS (CON1)
```

***Example 2:*** UR isolation acquires almost no locks. It is fast and causes little contention, but it reads uncommitted data. Do not use ISOLATION(UR) unless you are sure that your applications and end users can accept the logically inconsistent data that can occur, such as in the case of this example.

Assume that a supervisor routinely executes SQL statements using SPUFI to check the status of parts as they go through the assembly process and to update a table with the results of her inspection. She does not need to know the exact status of the parts; a small margin of error is acceptable.

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

If the supervisor executes a version of SPUFI that is bound with ISOLATION(UR), the query for the status of the parts executes without acquiring locks using UR isolation level and the update executes using CS isolation level. Thus, the query does not inadvertently hold locks in the production table interfering 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)
```

## 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. For more information on using BIND PLAN, see Section 6 of *DB2 Application Programming and SQL Guide*.

## 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[1] for BIND PLAN to build a plan without producing error messages. (The SYSADM authority includes this authorization.) 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.

Table 11 explains the authorization required to run BIND PLAN, depending on the options specified.

*Table 11 (Page 1 of 2). 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:<br>• BINDADD privilege<br>• SYSADM or SYSCTRL authority |
| ADD, specifying an OWNER other than the primary authorization ID | If the binder does not have SYSADM or SYSCTRL authority, the authorization ID of the new OWNER must have one of the following:<br>• BINDADD privilege<br>• SYSADM or SYSCTRL authority<br><br>**Specifying the OWNER:** If any of the authorization IDs of the process has the 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, then *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. |
| REPLACE, using the default owner or primary authorization ID | Primary authorization ID of the process must have one of the following:<br>• Ownership of the plan<br>• BIND privilege on the plan<br>• SYSADM or SYSCTRL authority |

[1] This excludes statements included in DBRMs that are bound to packages included in the package list of the plan.

*Table 11 (Page 2 of 2). Summary of privileges needed for BIND PLAN options*

| Option | Authorization required to run BIND PLAN |
|---|---|
| REPLACE, specifying an OWNER other than the primary authorization ID | If the binder does not have SYSADM or SYSCTRL authority, the authorization ID of the OWNER must have one of the following:<br>• Ownership of the plan<br>• BIND privilege on the plan<br>• SYSADM or SYSCTRL authority<br><br>**Specifying the OWNER:** If any of the authorization IDs of the process has the 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, then *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. |
| PKLIST, specifying individual packages | Authorization ID of the process must include one of the following:<br>• EXECUTE authority on each package specified in the PKLIST<br>• PACKADM authority on specific collections that contain the packages or on all collections<br>• SYSADM authority |
| PKLIST, specifying (*), indicating all packages in the collection | Authorization ID of the process must include one of the following:<br>• EXECUTE authority on *collection-id*.*<br>• PACKADM authority on specific collections that contain the packages or on all collections<br>• SYSADM authority |

For additional information on the required authorization to execute BIND PLAN see Section 5 (Volume 2) of *DB2 Administration Guide*.

## Syntax

```
  ►►──BIND──┬─────────────────┬──┬────────────────────────┬──┬───────────────────────────┬──────────────►
            └─PLAN(plan-name)─┘  └─OWNER(authorization-id)─┘  └─QUALIFIER(qualifier-name)─┘

     ┌───────────────────────────────────────────────────────────────────────────┐
     │               ┌─,────────────────┐                                        │
  ►──▼──┬─MEMBER(──▼─dbrm-member-name─┴─)──┬──────────────────────────────────┬──┼──────────────►
        │                                  │            ┌─,─────────────┐     │  │
        │                                  └─LIBRARY(──▼─dbrm-pds-name─┴─)──┘  │
        │          ┌─,────────────────────────────────────────────────────────┐   │
        └─PKLIST(──▼─┬─────────────────┬──┬─collection-id─┬──┬─.package-id─┬─┴─)─┘
                     ├─location-name.──┤  └─*─────────────┘  └─.*──────────┘
                     └─*.──────────────┘

     ┌─NODEFER(PREPARE)─┐
  ►──┴─DEFER(PREPARE)───┴──┬─────────────────────────────┬──┬───────────────────────────────────────────┬──►
                           │          ┌─USE──────┐       │  │             ┌─(──REPLACE──)───┬──────────┬┤
                           └─ACQUIRE(─┴─ALLOCATE─┴─)─────┘  │             │                 └─RETAIN───┘│
                                                            └─ACTION──────┴─(ADD)────────────────────────┘

  ►──┬──────────────────────────┬──┬──────────────────────────┬──┬───────────────────────────────────┬──►
     └─CACHESIZE(decimal-value)─┘  │               ┌─YES─┐    │  └─CURRENTSERVER(location-name)─┘
                                   └─CURRENTDATA(──┴─NO──┴─)──┘

  ►──┬──────────────────────────────┬──┬──────────────────────┬──┬───────────────────────────────────┬──►
|    └─DBPROTOCOL──(──┬─DRDA────┬─)─┘  │          ┌─1───┐     │  │             ┌─EXPLICIT────┐       │
                      └─PRIVATE─┘      └─DEGREE(──┴─ANY─┴─)───┘  └─DISCONNECT(──┼─AUTOMATIC───┼──)────┘
                                                                               └─CONDITIONAL─┘

  ►──┬────────────────────────────────┬────────────────────────────────────────────────────────────────►
     │                ┌─RUN──┐        │
     └─DYNAMICRULES(──┴─BIND─┴─)──────┘

  ►──┬─────────────────────────────────────────────────────────────────────────────────────────────┬───►
     │ ┌─ENABLE(*)─────────────────────────────────┐                                               │
     └─┴─┬─ENABLE──┬──(──▼─┬─BATCH────┬─┴──)──────┴──▼─┬─────────────────────────────────────────┬─┴─┘
         └─DISABLE─┘       ├─DLIBATCH─┤               │            ┌─,───────────────┐           │
                           ├─DB2CALL──┤               ├─DLIBATCH(──▼─connection-name─┴─)─────────┤
                           ├─CICS─────┤               │        ┌─,──────┐                        │
                           ├─IMS──────┤               ├─CICS(──▼─applid─┴─)──────────────────────┤
                           ├─IMSBMP───┤               │          ┌─,─────┐                       │
                           ├─IMSMPP───┤               ├─IMSBMP(──▼─imsid─┴─)─────────────────────┤
                           └─RRSAF────┘               │          ┌─,─────┐                       │
                                                      └─IMSMPP(──▼─imsid─┴─)─────────────────────┘

  ►──┬─────────────────────────┬──┬──────────────────┬──┬──────────────────────────────┬──┬──────────────────────────────┬──►
#    │            ┌─NO──┐      │  │       ┌─I─┐      │  └─IMMEDWRITE(──┬─NO──┬──)──────┘  └─ISOLATION(──┬─RR─┬──)───────┘
#    └─EXPLAIN(───┴─YES─┴─)────┘  └─FLAG(─┼─W─┼──)───┘                 ├─PH1─┤                        ├─RS─┤
#                                         ├─E─┤                        └─YES─┘                        ├─CS─┤
                                          └─C─┘                                                       └─UR─┘

                                           ┌─NOREOPT(VARS)─┐
  ►──┬─────────────────────────────┬──┴─REOPT(VARS)───┴──┬──────────────────────────┬──────────────────►
|    │                ┌─NO──┐       │                        └─OPTHINT──(──'hint-id'──)─┘
     └─KEEPDYNAMIC(───┴─YES─┴─)─────┘

  ►──┬───────────────────────────────────┬──┬────────────────────────────────┬──┬─────────────────────────┬──►
|    │       ┌─,─────────────┐           │  │           ┌─COMMIT─────┐       │  │            ┌─DB2─┐      │
|    └─PATH(─▼─┬─schema-name─┴─┴──)──────┘  └─RELEASE(──┴─DEALLOCATE─┴─)────┘  └─SQLRULES(──┴─STD─┴─)────┘
|              └─USER──────┘

  ►──┬──────────────────────────┬──────────────────────────────────────────────────────────────────────►◄
     │             ┌─RUN──┐     │
     └─VALIDATE(───┴─BIND─┴─)───┘
```

## Option descriptions

For descriptions of the options shown in the syntax diagram, see “Options of BIND and REBIND for PLAN, PACKAGE, and TRIGGER PACKAGE” on page 57.

## Examples

***Example 1:*** This subcommand creates a new plan called IMSONLY. The SQL statements for the plan are in the DBRM member DSN8BC61. 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 referenced in the DBRM.

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. Since 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) -
  MEMBER(DSN8BC61) -
  ACTION(ADD) -
  ISOLATION(CS) -
  OWNER(DEPTM92) -
  QUALIFIER(PRODUCTN) -
  CACHESIZE -
  ENABLE(IMS)
```

***Example 2:*** If the DBRM of plan IMSONLY in example 1 contains both embedded and dynamic SQL statements and you want to allow other users to run the plan, you must grant the EXECUTE privilege on plan IMSONLY to those users' authorization IDs. However, because the EXECUTE privilege on a plan is sufficient authority to run embedded SQL statements in a DBRM but is not sufficient authority to run dynamic SQL statements, you must also do one of the following:

- Use the SQL GRANT statement to grant the necessary privileges on the objects (tables, views, aliases, and indexes) referenced in the dynamic SQL statements to the users' authorization IDs, or
- BIND the plan IMSONLY with the option DYNAMICRULES(BIND) as follows:

```
BIND PLAN(IMSONLY) -
  MEMBER(DSN8BC61) -
  ACTION(ADD) -
  ISOLATION(CS) -
  OWNER(DEPTM92) -
  QUALIFIER(PRODUCTN) -
  CACHESIZE(0) -
  ENABLE(IMS) -
  DYNAMICRULES(BIND)
```

To allow other users having only the EXECUTE privilege on a plan to run both the embedded and dynamic SQL statements, you must bind that plan with the option DYNAMICRULES(BIND). When DYNAMICRULES(BIND) is in effect for plan IMSONLY:

- A single authorization ID, the authorization ID for DEPTM92, is used for authorization checking of both the embedded and dynamic SQL statements in the DBRM.
- PRODUCTN is the implicit qualifier of unqualified object names referenced in both the embedded and dynamic SQL statements in the DBRM.

***Example 3:*** This 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 DBRM. A cache size of 0 indicates that users will not run the plan repeatedly.

The option ENABLE(CICS) CICS(CON1) runs the plan only from CICS VTAM® node CON1 which is specified in the APPLID parameter of the CICS SIT table. If you attempt to run the plan from another environment or from another CICS VTAM note, the run attempt fails.

```
BIND PLAN(CICSONLY) -
  MEMBER(DSN8BC61) -
  ACTION(ADD) -
  ISOLATION(CS) -
  OWNER(DEPTM12) -
  QUALIFIER(TESTSYS) -
  CACHESIZE(0) -
  ENABLE(CICS) CICS(CON1)
```

## Options of BIND and REBIND for PLAN, PACKAGE, and TRIGGER PACKAGE

This section lists the options you can use for binding or rebinding plans and packages. Some of them are common for both bind and rebind and 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 you are rebinding.

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.

***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.

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

Determines whether to acquire resources for DBRMs specified in the MEMBER list when the application first accesses them or when the plan is allocated. Local or remote packages associated with the plan acquire their resources when the application first accesses them.

**(<u>USE</u>)**
Acquires table space locks only when the application program bound to the plan first uses them.

**(ALLOCATE)**
Acquires all table space locks when the plan is allocated. The value has no effect on dynamic SQL statements, which always use ACQUIRE(USE).

If you use ACQUIRE(ALLOCATE), you must also use RELEASE(DEALLOCATE). ACQUIRE(ALLOCATE) can increase the plan size, because additional items become resident in the plan.

***Defaults:***

| **Process** | **Default value** |
|---|---|
| BIND PLAN | USE |
| BIND PACKAGE | N/A |
| REBIND PLAN | existing value |
| 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). See Section 5 (Volume 2) of *DB2 Administration Guide*.

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

For more information about:

- How the option affects locking and concurrency, see Section 5 (Volume 2) of *DB2 Administration Guide* or Section 5 of *DB2 Application Programming and SQL Guide*.
- How the option improves the performance of selective partition locking, see Section 5 (Volume 2) of *DB2 Administration Guide* or Section 5 of *DB2 Application Programming and SQL Guide*.
- Estimating the size of a plan, see Section 2 of *DB2 Administration Guide*.

| | | |
|---|---|---|
| **ACTION** | (**REPLACE**)<br>(**REPLACE**) **RPLVER** (BIND PACKAGE only)<br>(**REPLACE**) **RETAIN** (BIND PLAN only)<br>(**ADD**) | **On: BIND PLAN and PACKAGE** |

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):*** This section describes the effect of ACTION(REPLACE) REPLVER in four situations. Here, DBRM1 is the member name and A and B represent the names of two versions of the package. Suppose you bind version A with this 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.

| | | |
|---|---|---|
| **CACHESIZE** | (*value of field PLAN AUTH CACHE*)<br>(*decimal-value*) | **On: BIND and REBIND PLAN** |

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:***

| Process | Default value |
|---|---|
| BIND PLAN | Value of field PLAN AUTH CACHE on installation panel DSNTIPP, which has a default of 0 |
| BIND PACKAGE | N/A |

| | |
|---|---|
| REBIND PLAN | Existing value |
| REBIND PACKAGE | N/A |

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

For additional information on determining an optimal cache size, see Section 6 of *DB2 Application Programming and SQL Guide*.

| **COPY** | (*collection-id.package-id*)<br>(*collection-id.package-id*) **COPYVER** | **On: BIND PACKAGE** |
|---|---|---|

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.

***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 Version 6 to some other server that does not support all the new BIND options in Version 6, use the new 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 Version 6.

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

| | | |
|---|---|---|
| **CURRENTDATA** | (<u>**YES**</u>)<br>(**NO**) | **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.

For more information about updating the current row of a cursor, block fetching, and data currency, see Section 5 of *DB2 Application Programming and SQL Guide*.

**(<u>YES</u>)** 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.

**(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

***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 | YES |
| BIND PACKAGE | YES |
| REBIND PLAN | Existing value |
| REBIND PACKAGE | Existing value |

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

| | | |
|---|---|---|
| **CURRENTSERVER** | (*location-name*) | **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. When the plan runs, the requester implicitly uses a type 1 CONNECT statement to that location.

You should use CURRENTSERVER to cause a local application to use data from a remote server without changing the application. Avoid using CURRENTSERVER with applications that contain explicit CONNECT statements. The implicit type 1 CONNECT statement that is used by CURRENTSERVER causes any explicit CONNECT statement issued in the application to be type 1, even if the application was precompiled with the default type 2.

*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 type 1 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) |
| BIND PACKAGE | N/A |
| REBIND PLAN | Existing value |
| REBIND PACKAGE | N/A |

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

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

| Specifies which protocol to use when connecting to a remote site that is identified by a three-part name statement.

| For DRDA, a package must be bound to each remote site that is referenced by a three-part name statement. Specify DRDA® to inform DB2 that the three-part name statements in the plan or package are to be converted to DRDA protocol.

| 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.

**(PRIVATE)**
DBPROTOCOL PRIVATE is passed on BIND PACKAGE, BIND PLAN, REBIND PACKAGE, or REBIND PLAN invocation.

***Defaults:***

| **Process** | **Default value** |
|---|---|
| BIND PLAN | DRDA |
| BIND PACKAGE | System default |
| REBIND PLAN | Value that was specified the last time the plan was bound |
| REBIND PACKAGE | Value that was specified the last time the package was bound |

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

## **DEFER(PREPARE)** **NODEFER(PREPARE)** — **On: BIND and REBIND PLAN and PACKAGE**

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 prepares 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 NOREOPT(VARS) applies:

- For local bind the package inherits the plan's option at runtime.
- For remote bind the default is NODEFER(PREPARE) at the remote DB2 server.

If neither DEFER nor NODEFER is specified and REOPT(VARS) applies, DEFER(PREPARE) is the default value.

You cannot use both DEFER(PREPARE) and NODEFER(PREPARE). In addition, you cannot use both NODEFER(PREPARE) and REOPT(VARS).

| | |
|---|---|
| **NODEFER(PREPARE)** | Does not defer preparation. |
| **DEFER(PREPARE)** | Defers preparation. |

***DEFER(PREPARE) and distributed processing:*** To improve performance, consider using DEFER(PREPARE) when binding dynamic or static SQL for DB2 private protocol access and when binding dynamic SQL for DRDA access. Specify the bind option DEFER(PREPARE) instead of NODEFER(PREPARE). DB2 does not prepare the dynamic SQL statement until that statement executes. This reduces network traffic, which improves the performance of the dynamic SQL statement.

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** | (<u>**1**</u>)<br>(**ANY**) | **On: BIND and REBIND PLAN and PACKAGE** |

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

For plans, the value of DEGREE applies only to the DBRMs bound directly to the plan (named in the MEMBER option on BIND PLAN), and has no affect on PKLIST names. 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.

| | |
|---|---|
| **(<u>1</u>)** | 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%–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.

| | | |
|---|---|---|
| **DISCONNECT** | (<u>**EXPLICIT**</u>)<br>(**AUTOMATIC**)<br>(**CONDITIONAL**) | **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** | (**RUN**)<br>(**BIND**)<br>(**DEFINEBIND**) (BIND and REBIND PACKAGE only)<br>(**DEFINERUN**) (BIND and REBIND PACKAGE only)<br>(**INVOKEBIND**) (BIND and REBIND PACKAGE only)<br>(**INVOKERUN**) (BIND and REBIND PACKAGE only) | **On: BIND and REBIND PLAN and 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 12 on page 68.

**(<u>RUN</u>)** 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 authorization ID of the application process and the SQL authorization ID (the value of the CURRENT SQLID special register) as the implicit qualifier 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 plan or 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 plan or package owner as the implicit qualifier.
- The attribute values that are described in Common attribute values for bind, define, and invoke behaviors on page 67.

The values of the authorization ID and the qualifier for unqualified objects are the same as those that are used for embedded or static SQL statements.

| **(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 state-
| ments in the application package.
| - The default qualifier for unqualified objects is the user-defined func-
| tion or stored procedure owner.
| - The attribute values that are described in Common attribute values
| for bind, define, and invoke behaviors on page 67.

# When the package is run as a stand-alone program, DB2 processes
# dynamic SQL statements using bind behavior, which is described in 66.

**(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 66.

When the package is run as a stand-alone program, DB2 processes dynamic SQL statements using run behavior, which is described in 66.

**(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 on page 67.

When the package is run as a stand-alone program, DB2 processes dynamic SQL statements using bind behavior, which is described in 66.

**(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 67.

When the package is run as a stand-alone program, DB2 processes dynamic SQL statements using run behavior, which is described in 66.

***Common attribute values for bind, define, and invoke behavior***: The following attribute values apply to dynamic SQL statements in plans or packages that have bind, define, or invoke behavior:

- You can execute the statement SET CURRENT SQLID in a package or plan 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 PLAN_TABLE.
- 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. For a list of the application
# programming defaults that the USE FOR DYNAMICRULES option affects, see
# Section 6 of *DB2 Application Programming and SQL Guide*.

| • 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.

For a plan or package that uses DB2 private protocol access, DB2 sends a DYNAMICRULES value of BIND or RUN to the server at run time, using the following rules:

- If the DYNAMICRULES value with which the package is bound is BIND, DEFINEBIND, or INVOKEBIND, DB2 sends a value of BIND to the server.
- If the DYNAMICRULES value with which the package is bound is RUN, DEFINERUN, or INVOKERUN, DB2 sends a value of RUN to the server.

Table 12 summarizes the dynamic SQL statement attribute values for each behavior. For more information about the dynamic SQL attributes that are affected by the DYNAMICRULES option, see Section 6 of *DB2 Application Programming and SQL Guide*.

*Table 12. Definitions of dynamic SQL statement behaviors*

| | **Value for dynamic SQL attributes** | | | |
|---|---|---|---|---|
| **Dynamic SQL attribute** | **Bind behavior** | **Run behavior** | **Define behavior** | **Invoke behavior** |
| Authorization ID | Package OWNER | Current SQLID | User-defined function or stored procedure owner | Authorization ID of invoker |
| Default qualifier for unqualified objects | Bind OWNER or QUALIFIER value | Current SQLID | User-defined function or stored procedure owner | Authorization ID of invoker |
| CURRENT SQLID | Initialized to Primary authid. SET SQLID is allowed. | Initialized to Primary authid. SET SQLID is allowed. | Initialized to Primary authid. SET SQLID is allowed. | Initialized to Primary authid. SET SQLID is allowed. |
| Source for application programming options | As determined by the DSNHDECP parameter DYNRULS | Install panel DSNTIPF/DSNHDECP application defaults | As determined by the DSNHDECP parameter DYNRULS | As determined by the DSNHDECP parameter DYNRULS |
| Can execute GRANT, REVOKE, CREATE, ALTER, DROP, RENAME? | No | Yes | No | No |

***Defaults:***

| **Process** | **Default value** |
|---|---|
| BIND PLAN | RUN |
| BIND PACKAGE | Plan value |
| REBIND PLAN | Existing value |
| REBIND PACKAGE | Existing value |

The default for a package on a remote server is RUN.

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

---

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

---

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.

# **<u>ENABLE</u>**
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:***

**(<u>*</u>)** Specifies all valid connection types. Use only with ENABLE.

**(BATCH)**
Indicates that all TSO connections are either enabled or disabled for the plan or package.

**(CICS)**
# Identifies the CICS connection. All CICS VTAM node names specified in the
# CICS SIT table are either enabled or disabled for the plan or package.

**(CICS) CICS(***applid***, ...)**
Identifies the CICS VTAM node name specified in the APPLID parameter of the CICS SIT table. The CICS VTAM node identified by *applid* is either enabled or disabled for the plan or package.

**(DB2CALL)**
Indicates that the call attachment facility (CAF) connection is either enabled or disabled for the plan or package.

**(DLIBATCH)**
Identifies the Data Language I (DL/I) Batch Support Facility connection. All connection identifiers from the DDITV02 data set or the job name in the JCL that the DL/I batch support system needs to have are either enabled or disabled for the plan or package.

**(DLIBATCH) DLIBATCH(***connection-name***, ...)**
Specifies the connection identifier as from the DDITV02 data set or the job name in the JCL that the DL/I batch support system needs to have. The DL/I batch connection identified by *connection-name* is either enabled or disabled for the plan or package.

**(IMS)**
Specifies that all Information Management System (IMS) connections, DLIBATCH, IMSBMP, and IMSMPP are either enabled or disabled for the plan or package.

**(IMSBMP)**
Specifies the IMS connection for the Batch Message Program (BMP) region. All IMS BMP connections identified by the value of IMSID on the CTL parameter EXEC are either enabled or disabled for the plan or package.

**(IMSBMP) IMSBMP(***imsid***, ...)**
Specifies the value of IMSID on the CTL parameter EXEC. The IMS BMP connection identified by *imsid* is either enabled or disabled for the plan or package.

**(IMSMPP)**
Specifies the IMS connection for the Message Processing Program (MPP) and IMS Fast Path (IFP) regions. All IMS MPP connections identified by the value of the IMSID on the CTL parameter EXEC. are either enabled or disabled for the plan or package.

**(IMSMPP) IMSMPP(***imsid***, ...)**
Specifies the value of IMSID on the CTL parameter EXEC. The IMS MPP connection identified by *imsid* is either enabled or disabled for the plan or package.

**(REMOTE)**
Indicates that all remote connections are either enabled or disabled for the plan or 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 OS/390 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 OS/390 subsystem. You must bracket a logical unit name with the less than (<) and the greater than (>) characters to differentiate it from a location name. |

**(RRSAF)**
# Indicates that the RRS attachment facility connection is either enabled or disabled for the plan or package.

***Performance hint:*** Whenever the plan or package is allocated, DB2 must check the connection type and connection name with the list of enabled or disabled connections. For best performance, keep the list short.

***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.

Product-sensitive Programming Interface

| | | |
|---|---|---|
| **EXPLAIN** | (<u>**NO**</u>)<br>(**YES**) | **On: BIND and REBIND PLAN and PACKAGE , REBIND TRIGGER PACKAGE** |

Obtains information about how SQL statements in the package, or in the member list of the plan, are to execute, and then inserts that information into the table *owner*.PLAN_TABLE, where *owner* is the authorization ID of the owner of the plan or package. This option does not obtain information for statements that access remote objects.

PLAN_TABLE must be a table; it cannot be a view, alias, or synonym. It should exist before the bind process begins.

| The EXPLAIN option also populates two optional tables, if they exist: DSN_STATEMNT_TABLE and DSN_FUNCTION_TABLE.

DSN_STATEMNT_TABLE contains DB2's estimate of the processing cost for an SQL statement. See Section 7 of *DB2 Application Programming and SQL Guide* for more information.

| DSN_FUNCTION_TABLE contains information about function resolution. See
| Section 4 of *DB2 Application Programming and SQL Guide* for more information.

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 a description of the tables populated by the EXPLAIN option, see information
| about the EXPLAIN statement in Chapter 6 of *DB2 SQL Reference*.

***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.

**(<u>NO</u>)** Provides no EXPLAIN information.

**(YES)** Inserts information in the tables populated by EXPLAIN. 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.

***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 EXPLAN of SYSPLAN.

End of Product-sensitive Programming Interface

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

Determines what messages to display.

**(<u>I</u>)** 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 |

| | | |
|---|---|---|
| **IMMEDWRITE** | (<u>**NO**</u>)<br>(**PH1**)<br>(**YES**) | **On: BIND and REBIND PLAN and PACKAGE** |

Tells whether immediate writes will be done for updates made to group buffer pool dependent pagesets or partitions. This option is only applicable for data sharing environments. Table 13 on page 74 shows the implied hierarchy of this option. The IMMEDWRITE option values are as follows:

**(<u>NO</u>)** Specifies that normal write activity is done. Updated pages that are group buffer pool dependent are written at or before phase two of commit or at the end of abort for transactions that have rolled back.

**(PH1)** Specifies that updated pages that are group buffer pool dependent are written at or before phase one of commit. If the transaction subsequently rolls back, the pages will be updated again during the rollback process, and they will be written again at the end of abort.

**(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 may impact performance.

# Table 13. The implied hierarchy of the IMMEDWRITE option

| # IMMEDWRITE bind option | IMMEDWRI subsystem parameter | Value at run time |
|---|---|---|
| # NO | NO | NO |
| # NO | PH1 | PH1 |
| # NO | YES | YES |
| # PH1 | NO | PH1 |
| # PH1 | PH1 | PH1 |
| # PH1 | YES | YES |
| # YES | NO | YES |
| # YES | PH1 | YES |
| # YES | YES | YES |

| ***Performance hints:*** You can use IMMEDWRITE(PH1) and IMMEDWRITE(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 IMMEDWRITE(PH1) to cause group buffer pool dependent pages to be written at or before phase 1 of commit.

Specify IMMEDWRITE(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. IMMEDWRITE(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 IMMEDWRITE(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 | NO |
| \| REBIND PLAN | Existing value |
| \| REBIND PACKAGE | Existing value |

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

| | | |
|---|---|---|
| **ISOLATION** | (<u>**RR**</u>)<br>(**RS**)<br>(**CS**)<br>(**UR**)<br>(**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. For more information on isolation levels, see Improving Concurrency in Section 5 (Volume 2) of DB2 Administration Guide.

**(<u>RR</u>)** *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.

**(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.

**(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 OS/390. DB2 for OS/390 does not support NC. If the server does not support this isolation level, it uses UR.

For more information about how the ISOLATION option affects locking and concurrency, including how DB2 resolves conflicts by using the most restrictive value when the values specified in the plan and package differ, see Section 5 of *DB2 Application Programming and SQL Guide*.

***Defaults:***

| **Process** | **Default value** |
|---|---|
| BIND PLAN | RR |
| BIND PACKAGE | Plan value |
| REBIND PLAN | Existing value |
| REBIND PACKAGE | Existing value |

The default for binding a package to a remote server is **RR**.

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** | (<u>**NO**</u>)<br>(**YES**) | **On: BIND and REBIND PLAN and PACKAGE** |

Determines whether DB2 keeps dynamic SQL statements after commit points.

**(<u>NO</u>)** 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), you must not specify REOPT(VARS). KEEPDYNAMIC(YES) and REOPT(VARS) are mutually exclusive.

***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 KEEPDYNAMIC(NO).

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

| | | |
|---|---|---|
| **LIBRARY** | (*dbrm-pds-name*)<br>(*dbrm-pds-name*, ...) (BIND PLAN only) | **On: BIND PLAN, BIND PACKAGE** |

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.

| | | |
|---|---|---|
| **MEMBER** | (*dbrm-member-name*)<br>(*dbrm-member-name*, ...) (BIND PLAN only) | **On: BIND PLAN, BIND PACKAGE** |

Determines what database request modules (DBRMs) to include in the plan or package.

*dbrm-member-name*

# Specifies the name of a library member that contains a DBRM. You can name
# the partitioned data set, of which a DBRM is a member, either in the LIBRARY
# option or in the JCL statement for DBRMLIB DD.

# For BIND PACKAGE only, the name becomes the package name. Names
# beginning with DSN are reserved; you receive a warning message if you use
# one.

**For BIND PACKAGE**, you can use only one member. If you do not use MEMBER, you must use COPY. You cannot use both options.

**For BIND PLAN**, you can list many members. DB2 sorts the member list in alphabetical order. If you do not use MEMBER, you must use PKLIST.

***Defaults:***

| Process | Default value |
|---|---|
| BIND PLAN | None |
| BIND PACKAGE | None |
| REBIND PLAN | N/A |
| REBIND PACKAGE | N/A |

***Catalog record:*** Column NAME of table SYSPACKAGE for BIND PACKAGE, or the table SYSDBRM for BIND PLAN.

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

Controls whether query optimization hints are used for static SQL.

**('***hint-id***')** A character string of up to eight 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 DSNTIP4.

For more information about using the OPTHINT option, see Section 5 (Volume 2) of *DB2 Administration Guide*.

***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(COMPOSITE)**
**OPTIONS(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 to be copied.

**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 options defaults are used at the server. You must use the OPTIONS(COMMAND) when copying to a downlevel server. A down-level server is any server that is not DB2 Version 6.

***Defaults:***

| **Process** | **Default value** |
|---|---|
| BIND PACKAGE COPY | OPTIONS(COMPOSITE) |

---

| **OWNER** | (*authorization-id*) | **On: BIND and REBIND PLAN and PACKAGE** |
|---|---|---|

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.

**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** | (*location-name.collection-id.package-id.(version-id)*)<br>(*****) (REBIND PACKAGE only) | **On: BIND and REBIND PACKAGE** |
|---|---|---|

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.

*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.

*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.

# DBRMs created in releases of DB2 before Version 2 Release 3 use a
# *version-id* of the empty string by default.

**(*) (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.

For more information about:

# • How to define a location name in SYSIBM.LOCATIONS, see Section 3 of *DB2*
# *Administration Guide*.

- Which packages are bound depending on how you specify collections, packages, and versions on the REBIND PACKAGE command, see Section 5 of *DB2 Application Programming and SQL Guide*.

| | | |
|---|---|---|
| \| **PATH** | (*schema-name*) (**USER**) (*schema-name*, **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 precom-
| pile 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 above. If you do
| not specify the SYSIBM, SYSFUN, and SYSPROC schemas, they are not
| included in the 254-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 you can specify is limited by the length of the
| resulting SQL path, which cannot exceed 254 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 254 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.

For more information about schema names, ordinary identifiers, and delimited identifiers, see Chapter 3 of *DB2 SQL Reference*.

***Defaults:***

| **Process** | **Default value** |
|---|---|
| BIND PLAN | "SYSIBM", "SYSFUN", "SYSPROC", *plan qualifier* [2] |
| BIND PACKAGE | "SYSIBM", "SYSFUN", "SYSPROC", *package qualifier* [2] |
| REBIND PLAN | Existing value |
| REBIND PACKAGE | Existing value |

**PATHDEFAULT** **On: REBIND PLAN and PACKAGE**

Resets the **PATH** for a package or plan to "SYSIBM", "SYSFUN", "SYSPROC", *plan-qualifier/package-qualifier*.

The **PATHDEFAULT** keyword is mutually exclusive with the **PATH** keyword. Do not specify both keywords in the same REBIND command.

***Defaults:***

| **Process** | **Default value** |
|---|---|
| REBIND PLAN | None |
| REBIND PACKAGE | None |

**PKLIST**
**NOPKLIST** (*location-name.collection-id.package-id*, ...) PKLIST only **On: BIND and REBIND PLAN**

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.

NOPKLIST is used with REBIND PLAN only. NOPKLIST determines that the plan rebinds without a package list. If a package list already exists, then NOPKLIST deletes it.

*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.

[2] Although this is the default value, it is not stored in the catalog. Instead, the catalog value is blank.

# 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 (*). There is no default.

If you use an asterisk, then DB2 checks the privileges to use the SQL statements embedded in the package 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, then 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 occurs.

*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.

For more information about:

- How the order of search for packages affects performance, see Section 6 of *DB2 Application Programming and SQL Guide*.
- # How to define a location name in SYSIBM.LOCATIONS, see Section 3 of *DB2*
  # *Administration Guide*.
- The TSO/E restriction that limits the maximum number of packages specified in the PKLIST, see OS/390 TSO/E Programming Services.

| | | |
|---|---|---|
| **PLAN** | (*plan-name*)<br>(*) (REBIND PLAN only) | **On: BIND and REBIND PLAN** |

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.

| | | |
|---|---|---|
| **QUALIFIER** | (*qualifier-name*) | **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 trans-
# lation 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.

| | | |
|---|---|---|
| **RELEASE** | (<u>**COMMIT**</u>)<br>\| (**DEALLOCATE**) | **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.

**(<u>COMMIT</u>)**
Releases resources at each commit point.

**(DEALLOCATE)**
Releases resources only when the program terminates.

The value has no effect on dynamic SQL statements, which always use RELEASE(COMMIT), with one exception: When you use RELEASE(DEALLOCATE) and KEEPDYNAMIC(YES), and your subsystem is installed with YES for field CACHE DYNAMIC SQL on installation panel DSNTIP4, the RELEASE(DEALLOCATE) option is honored for dynamic SELECT, INSERT, UPDATE and DELETE statements.

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.

For more information about how the RELEASE option affects locking and concurrency, see Section 5 (Volume 2) of *DB2 Administration Guide* or Section 6 of *DB2 Application Programming and SQL Guide*.

***Defaults:***

| **Process** | **Default value** |
|---|---|
| BIND PLAN | COMMIT |
| BIND PACKAGE | Plan value |
| REBIND PLAN | Existing value |
| REBIND PACKAGE | Existing value |

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

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

## <u>NOREOPT(VARS)</u> REOPT(VARS)

**On: BIND and REBIND PLAN and PACKAGE**

Specifies whether to have DB2 determine an access path at run time using values for host variables, parameter markers, and special registers.

| | |
|---|---|
| **<u>NOREOPT(VARS)</u>** | Does not determine an access path at run time. |
| **REOPT(VARS)** | Re-determines the access path at run time. |

***Usage notes:***

- You cannot use both REOPT(VARS) and NOREOPT(VARS).
- You cannot use both REOPT(VARS) and KEEPDYNAMIC(YES).

- You cannot use both REOPT(VARS) and NODEFER(PREPARE).

***Defaults:***

| **Process** | **Default value** |
|---|---|
| BIND PLAN | NOREOPT |
| BIND PACKAGE | NOREOPT |
| REBIND PLAN | Existing value |
| REBIND PACKAGE | Existing value |

The default for a package on a remote DB2 server is NOREOPT(VARS).

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

**SQLERROR** (**<u>NOPACKAGE</u>**) (**CONTINUE**) **On: BIND PACKAGE only**

Determines whether to create a package if SQL errors occur.

**(<u>NOPACKAGE</u>)**
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.

***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** (**<u>DB2</u>**) (**STD**) **On: BIND and REBIND PLAN**

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 or the rules for DB2 private protocol access.

**(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.

| | | |
|---|---|---|
| **VALIDATE** | (**RUN**)<br>(**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)** Indicated 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.

**(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.

***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.

## -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 an MVS 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, the privilege set of this process must include one of the following:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
►►──CANCEL──┬─THREAD(token)──────────────┬──┬──────┬──────────────────────────────────────►◄
            └─DDF THREAD(─┬─luwid─┬─)─┘  └─DUMP─┘
                          └─token─┘
```

## 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. It can be determined from the DB2 command DISPLAY THREAD or from 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.

**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.

## 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. See Chapter 5 of the DB2 Administration Guide, for an explanation of global transactions.

***Canceling active threads:*** 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 MVS Cancel command to terminate the application immediately.
- Hung up in VTAM. Use VTAM commands to cause VTAM to return processing to DB2, which will terminate the thread. See topic below for details.

***Using VTAM commands to cancel SNA distributed threads:*** 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 do this, 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.LUND0.9F6D9F459E92=123 ACCESSING DATA AT
 V446-SAN JOSE:LUND1
 V447--LOCATION         SESSID             A ST    TIME
 V448--SAN JOSE         00D3590EA1E89701     S1    9332108460302
 V448--SAN JOSE         00D3590EA1E89822   V R1    9332108460431
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

The **V** indicates the thread is processing in VTAM.

2. Record positions 3 through 16 of SESSID for the threads to be canceled. (In the DISPLAY THREAD output above, the values are D3590EA1E89701 and D3590EA1E89822.)

3. Issue the VTAM command DISPLAY NET to display the VTAM session IDs. The ones you want to cancel match the SESSIDs in positions 3 through 16 and the corresponding session IDs are in bold. The following is an output example of this command:

```
D NET,ID=LUND0,SCOPE=ACT

IST097I DISPLAY ACCEPTED
IST075I NAME = LUND0, TYPE = APPL
   IST486I STATUS= ACTIV, DESIRED STATE= ACTIV
   IST171I ACTIVE SESSIONS = 0000000005, SESSION REQUESTS = 0000000000
   IST206I SESSIONS:
   IST634I NAME     STATUS          SID          SEND   RECV  VR  TP  NETID
   IST635I LUND1    ACTIV-S     D24B171032B76E65  0051   0043   0   0  NET2
   IST635I LUND1    ACTIV-S     D24B171032B32545  0051   0043   0   0  NET2
   IST635I LUND1    ACTIV-R     D2D3590EA1E89701  0022   0031   0   0  NET2
   IST635I LUND1    ACTIV-R     D2D3590EA1E89802  0022   0031   0   0  NET2
   IST635I LUND1    ACTIV-R     D2D3590EA1E89822  0022   0031   0   0  NET2
   IST314I END
```

4. Issue the VTAM command VARY NET,TERM for each of the VTAM SIDs associated with the DB2 thread. In this case, it might be necessary to cancel only the session ID that DISPLAY THREAD shows to be processing in VTAM (D2D3590EA1E89822).

For more information about VTAM commands, see *VTAM for MVS/ESA Operation*.

## Examples

***Example 1:*** To cancel a non-distributed thread whose token you found through -DISPLAY THREAD and to produce a diagnostic dump, issue:

```
-CANCEL THREAD (123) DUMP
```

***Example 2:*** To cancel a distributed thread whose LUWID you found through -DISPLAY THREAD, issue:

```
-CANCEL DDF THREAD (LUDALLAS.DB2SQL1.3042512B6425)
```

Assume 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:

```
-CANCEL DDF THREAD (45162)
```

or

```
-CANCEL THREAD (45162)
```

As in the first example, specifying DUMP with any of the commands shown in this example causes a diagnostic dump to be produced.

## /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

## 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, as described in *IMS/ESA Administration Guide: System.*

## Syntax

```
►►─/CHANGE──┬─SUBSYS─subsystem-name──────────────────────────┬─RESET──────────────────────────────►◄
            ├─SUBSYS─ALL─────────────────────────────────────┤
            └─SUBSYS─subsystem-name─OASN─schedule-number─────┘
```

## Option descriptions

**SUBSYS**
Deletes IMS recovery elements from one or more subsystems.

One of the following subparameters must be coded:

*subsystem-name*, **...**
Specifies one or more subsystems from which recovery elements will be deleted.

**ALL**
Deletes IMS recovery elements from all subsystems.

*subsystem-name* **OASN** *schedule-number*, **...**
Deletes one or more origin application schedule numbers 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.

**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. For a complete description, see *IMS/ESA Operator's Reference*.

## Examples

***Example 1:*** Reset all indoubt recovery units for subsystem DB2.

```
/CHA SUBSYS DB2 RESET
```

***Example 2:*** Reset all indoubt recovery units for all subsystems.

```
/CHA SUBSYS ALL RESET
```

***Example 3:*** Reset indoubt recovery units identified by OASN numbers 99, 685, and 2920 for subsystem DB2.

```
/CHA SUBSYS DB2 OASN 99 685 2920 RESET
```

## 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 view named in the catalog.

For further information regarding the DCLGEN command and uses for its output, see Section 3 of *DB2 Application Programming and SQL Guide*.

## 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, the privilege set of the process must include one of the following:

- SELECT privilege on the table or view
- Ownership of the table or view
- DBADM authority on the database containing the table
- SYSADM authority
- SYSCTRL authority (catalog tables only)

## Syntax

```
►►──DCLGEN──TABLE(──┬─table-name─┬──)──┬──────────────────────┬──┬──────────────────────┬──────────►
                    └─view-name──┘     └─OWNER(owner-name)─┘  └─AT(location-name)─┘

►──LIBRARY(library name─┬──────────────────┬──┬─────────────┬──)──┬─────────────────────────────┬──►
                        └─(member-name)─┘  └─/password─┘        │            ┌─ADD─────┐      │
                                                                └─ACTION(──┴─REPLACE─┴──)─┘

►──┬─────────────────────────────┬──┬────────────────┬──┬──────────────────────────────┬──┬───────┬──►
   │             ┌─COBOL──┐      │  └─NAMES(prefix)─┘  └─STRUCTURE(structure-name)─┘  ├─APOST─┤
   └─LANGUAGE(──┼─PLI────┼──)──┘                                                       └─QUOTE─┘
                ├─C──────┤
                ├─COB2───┤
                ├─IBMCOB─┤
                └─CPP────┘

►──┬────────────────┬──┬─────────────────────┬──┬──────────────────────────┬──┬──────────────────────────┬──►
   │      ┌─NO──┐   │  └─DBCSSYMBOL(──┬─G─┬──)─┘  │           ┌─YES─┐        │  │           ┌─NO──┐        │
   └─LABEL──┴─YES─┴─┘                 └─N─┘       └─DBCSDELIM(──┴─NO──┴──)─┘  └─COLSUFFIX(──┴─YES─┴──)─┘

►──┬───────────────────────┬──────────────────────────────────────────────────────────────────────►◄
   │        ┌─NO──┐        │
   └─INDVAR(──┴─YES─┴──)─┘
```

## Option descriptions

**TABLE**

# Specifies what table or view for which to generate a declaration. *table-name* or *view-name* is the qualified or unqualified name of the table or view.

The name must follow the following 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 name | OWNER(*owner-name*) specified | OWNER(*owner-name*) not specified |
|---|---|---|
| TABLE(*table-name*) qualified | *table-name* qualifier | *table-name* qualifier |
| TABLE(*table-name*) unqualified | *owner-name* | 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.

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: COBOL, COB2, PLI, or C; |
| (*member-name*) | Optional; if not used, the output goes to a sequential data set. |

*password* is optional.

**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:

- (**COBOL**), for OS/VS COBOL
- (**COB2**), for other COBOL languages
- (**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. Special characters can be used, 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, as in Version 3.

**(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, then the field name is “NEWEMPNO.”

If NAMES is *not* specified, DCLGEN issues a warning message and uses the column names as the field names, as in Version 3.

**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).

## Usage notes

Parsing of the DCLGEN command conforms to standard TSO parsing conventions. For information about TSO command parsing, see the *OS/390 TSO/E Programming Services*.

***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.

***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 999,999,999, 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 1:*** This example shows the use of the DCLGEN. The statement

```
DCLGEN TABLE(VEMPL) -
       LIBRARY('prefix.SRCLIB.DATA(DSN8MPEM)') -
       LANGUAGE(PLI) -
       APOST
```

produces the following statements in *prefix*.SRCLIB.DATA(DSN8MPEM):

```
/*********************************************************************/
/* DCLGEN TABLE(VEMPL) -                                             */
/*        LIBRARY('prefix.SRCLIB.DATA(DSN8MPEM)') -                  */
/*        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 2:*** This example shows the use of NAMES and STRUCTURE. The statement

```
DCLGEN TABLE(VEMPL) -
       LIBRARY('prefix.SRCLIB.DATA(DSN8MPEM)') -
       LANGUAGE(PLI) -
       NAMES(FIELD) -
       STRUCTURE(EMPRECORD) -
       APOST
```

produces the following statements in *prefix*.SRCLIB.DATA(DSN8MPEM):

```
/*********************************************************************/
/* DCLGEN TABLE(VEMPL) -                                             */
/*        LIBRARY('prefix.SRCLIB.DATA(DSN8MPEM)') -                  */
/*        LANGUAGE(PLI) -                                            */
/*        NAMES(FIELD) -                                             */
/*        STRUCTURE(EMPRECORD) -                                     */
/*        APOST                                                      */
/* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS      */
/*********************************************************************/
EXEC SQL DECLARE VEMPL TABLE
          ( EMPNO                          CHAR(6) NOT NULL,
            FIRSTNME                       VARCHAR(12) NOT NULL,
            MIDINIT                        CHAR(1) NOT NULL,
            LASTNAME                       VARCHAR(15) NOT NULL,
            WORKDEPT                       CHAR(3) NOT NULL
          ) ;
/*********************************************************************/
/* PLI DECLARATION FOR TABLE VEMPL                                   */
/*********************************************************************/
DCL 1 EMPRECORD,
     5 FIELD1   CHAR(6),
     5 FIELD2   CHAR(12) VAR,
     5 FIELD3   CHAR(1),
     5 FIELD4   CHAR(15) VAR,
     5 FIELD5   CHAR(3);
/*********************************************************************/
/* THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 5          */
/*********************************************************************/
```

## /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 associated with the subsystem.

### 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, as described in the *IMS/ESA Administration Guide: System*.

### Syntax

```
                      ┌─,──────────────┐
►►──/DISPLAY──┬─SUBSYS──▼─subsystem-name─┴──┬──────────────────────────────────────────────►◄
              ├─SUBSYS──ALL─────────────────┤
              │              ┌─,──────────────┐ │
              ├─OASNSUBSYS──▼─subsystem-name─┴─┤
              └─OASN──SUBSYS──ALL──────────────┘
```

### Option descriptions

One of the following options is required:

**SUBSYS**
# Identifies the subsystems to display information about.

*subsystem-name*, **...**
Specifies one or more subsystems. See "Output" on page 103 for a description of possible subsystem status.

**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:

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.

The preceding description of the /DISPLAY command is a partial description only. For a complete description, see *IMS/ESA Operator's Reference*.

## Example

Display the status of connections between IMS and all subsystems.

```
/DISPLAY SUBSYSTEM ALL
```

```
SUBSYS    CRC    REGID    PROGRAM    LTERM    STATUS
SSTR      ?                                   CONN
                   1      DDLTLM17   PTERM01  CONN,ACTIVE
                   2      DDLTLM06   PTERM02  CONN
*85202/065933*
```

## -DISPLAY ARCHIVE (DB2)

The DB2 command DISPLAY ARCHIVE displays input archive log information.

**Abbreviation**: -DIS ARC

## Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following:

- ARCHIVE or DISPLAY system privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any 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 to each member.

## Example

Display tape unit information about input archive logs.

```
-DISPLAY ARCHIVE
```

```
DSNJ322I - DISPLAY ARCHIVE REPORT FOLLOWS-
               COUNT                TIME
             (TAPE UNITS)         (MIN,SEC)
DSNZPARM          2                   0,00
CURRENT           2                   5,30
===============================
ADDR STATUS CORR-ID  VOLSER DATASET_NAME
 290  AVAIL  *****   TAPE1  DSNCAT.ARCHLOG1.A0000033
 294  PREM   *****   TAPE3  DSNCAT.ARCHLOG1.A0000035
 293  BUSY  RECOVER2 TAPE2  DSNCAT.ARCHLOG1.A0000034
END OF DISPLAY ARCHIVE REPORT.
DSN9022I - DSNJC001 '-DISPLAY ARCHIVE' NORMAL COMPLETION
```

This example report shows:

- 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 dedicated tape units
- Volume and data set names associated with all busy tape units

## -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 an MVS 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, the privilege set of the process must include one of the following:

- DISPLAY system privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
                      ┌─(──ACTIVE──)──────┐
►►──DISPLAY BUFFERPOOL┴┬──────────────────┬┴──┬────────────────────────────┬──┬──────────────────────────┬──────►
                       ├─(──*──)──────────┤   │          ┌─INTERVAL─┐      │  │        ┌─ACTIVE─┐        │
                       │     ┌─,──────┐   │   └─DETAIL(──┴─*────────┴──)───┘  └─LIST(──┴─*──────┴──)─────┘
                       └─(───▼─bpname─┴──)┘

                  ┌─ACTIVE─┐
►──┬────────┬──(──┴─*──────┴──)──┬───────────────────────────────────────┬──────────────────────────────────►
   └─LSTATS─┘                    │          ┌─*─────────────────────┐    │
                                 │          │ ┌─,─────────────┐     │    │
                                 └─DBNAME(──┼─▼─database-name─┴─────┼──)─┘
                                            ├─name1:name2───────────┤
                                            └─name*─────────────────┘

►──┬──────────────────────────────────────┬──┬────────────────────────┬──┬──────────────────────────┬────────►◄
   │            ┌─*────────────────────┐  │  └─GBPDEP(──┬─YES─┬──)────┘  └─CASTOWNR(──┬─YES─┬──)────┘
   │            │ ┌─,──────────┐       │  │             └─NO──┘                       └─NO──┘
   └─SPACENAM(──┼─▼─space-name─┴───────┼──)┘
                ├─name1:name2──────────┤
                └─name*────────────────┘
```

## Option descriptions

**(<u>ACTIVE</u>)**
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, BP1, ..., BP49.
- 8-KB page buffer pools are named BP8K0, BP8K1, ..., BP8K9.
- 16-KB page buffer pools are named BP16K0, BP16K1, ..., BP16K9.
- 32-KB page buffer pools are named BP32K, BP32K1, ..., 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 there was no previous incremental display.

**(*)** 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 no-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, then 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 from one to eight characters. *name* represents a string of from 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 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:

- There is inter-DB2 R/W interest in it.
- There are changed pages from it in the group buffer pool that have not yet been written to DASD.

**(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

You can request a summary report or a detail report.

### Summary report

A summary report contains the following information, as seen in Example 1 on page 113:

***Identification***

| | |
|---|---|
| BUFFERPOOL NAME | Buffer pool external name (BP0, BP1, ..., BP49, or BP32K, BP32K1, ..., BP32K9). |
| BUFFERPOOL ID | Buffer pool internal identifier (0-49, 80-89, 100-109, 120-129). |
| USE COUNT | Number of open table spaces or index spaces that reference this buffer pool. (Inactive pools have a zero use count.) |
| VIRTUAL BUFFERPOOL SIZE | User-specified virtual buffer pool size |
| BUFFERS ALLOCATED | Number of allocated buffers in an active virtual buffer pool. |
| TO BE DELETED | Number of buffers to be deleted in an active virtual buffer pool (because of pool contraction). |
| IN-USE/UPDATED | Number of currently active (not stealable) buffers in the virtual buffer pool. |

***Hiperpool Values***

| | |
|---|---|
| HIPERPOOL SIZE | User-specified hiperpool size. |
| CASTOUT | Hiperpool CASTOUT attribute. |
| ALLOCATED | Number of allocated buffers in an active hiperpool. |
| TO BE DELETED | Number of buffers to be deleted in an active hiperpool (because of pool contraction). |
| BACKED BY ES | Number of hiperpool buffers backed by expanded storage. |

***Thresholds***

| | |
|---|---|
| VP SEQUENTIAL | Sequential steal threshold for the virtual buffer pool. |
| HP SEQUENTIAL | Sequential steal threshold for the hiperpool. |
| DEFERRED WRITE | Deferred write threshold for the virtual buffer pool. |
| VERTICAL DEFERRED WRT | Vertical deferred write threshold for the virtual buffer pool. |
| PARALLEL SEQUENTIAL | Parallel sequential threshold for the virtual buffer pool. |
| ASSISTING PARALLEL SEQT | Assisting parallel sequential threshold for the virtual buffer pool. |

***Names of hiperspaces allocated to the hiperpool:*** A hiperspace name consists of 3 parts, with a total length of 8 characters. A hiperspace name always starts with an indicator of '@' followed by:

1. The buffer pool's internal ID (two characters, using Hex)
2. A sequence number about hiperspace allocation (one character)
3. The DB2 subsystem name (four characters)

## Detail report

A detail report includes all summary report information and additional buffer pool related statistics. You can request cumulative statistics (accumulated since DB2 startup) or incremental statistics (accumulated since the last incremental display). A sample report appears in Example 2 on page 113. The statistics in a detail report are grouped in the following categories:

***Getpage information (message DSNB411I):***

| | |
|---|---|
| RANDOM GETPAGE | Non-sequential getpage requests |
| SYNC READ I/O(R) | Synchronous read I/O operations for non-sequential getpage |
| SEQ. GETPAGE | Sequential getpage requests |
| SYNC READ I/O(S) | Synchronous read I/O operations for sequential getpage |
| DMTH HIT | Number of times data management threshold reached |

***Sequential prefetch statistics (message DSNB412I):***

| | |
|---|---|
| REQUESTS | Sequential prefetch requests |
| PREFETCH I/O | Sequential prefetch read I/O operations |
| PAGES READ | Number of pages read with sequential prefetch |

***List prefetch statistics (message DSNB413I):***

| | |
|---|---|
| REQUESTS | List prefetch requests |
| PREFETCH I/O | List prefetch read I/O operations |
| PAGES READ | Number of pages read due to list prefetch |

***Dynamic prefetch statistics (message DSNB414I):***

| | |
|---|---|
| REQUESTS | Dynamic prefetch requests |
| PREFETCH I/O | Dynamic prefetch read I/O operations |
| PAGES READ | Number of pages read with dynamic prefetch |

***Disabled prefetch statistics (message DSNB415I):***

| | |
|---|---|
| NO BUFFER | Prefetch disabled - no buffer |
| NO READ ENGINE | Prefetch disabled - no read processor |

***Page update statistics (message DSNB420I):***

| | |
|---|---|
| SYS PAGE UPDATES | System page updates |

| | |
|---|---|
| SYS PAGES WRITTEN | System pages written |
| ASYNC WRITE I/O | Asynchronous write I/O operations |
| SYNC WRITE I/O | Synchronous write I/O operations |

***Page write statistics (message DSNB421I):***

| | |
|---|---|
| DWT HIT | Number of times deferred write threshold reached |
| VERTICAL DWT HIT | Number of times vertical deferred write threshold reached |
| NO WRITE ENGINE | Number of times write processor not available for I/O operations |

***Hiperpool activity (not using the Asynchronous Data Mover Facility): (message DSNB430I)***

| | |
|---|---|
| SYNC HP READS | Number of times that a requested page was found in hiperpool and synchronously moved to virtual buffer pool |
| SYNC HP WRITES | Number of pages synchronously moved from the virtual buffer pool to the hiperpool |
| ASYCN HP READS | Number of times that a requested page was found in hiperpool and asynchronously moved to virtual buffer pool without the use of the Asynchronous Data Mover Facility |
| ASYNC HP WRITES | Number of pages asynchronously moved from the virtual buffer pool to the hiperpool without the use of the Asynchronous Data Mover Facility |
| READ FAILURES | Number of page read failures (other than those that occurred using the Asynchronous Data Mover Facility) |
| WRITE FAILURES | Number of page write failures (other than those that occurred using the Asynchronous Data Mover Facility) |

***Hiperpool activity (using the Asynchronous Data Mover Facility) (message DSNB431I):***

| | |
|---|---|
| HP READS | Number of pages moved asynchronously from the hiperpool to the virtual pool using the Asynchronous Data Mover Facility |
| HP WRITES | Number of pages moved asynchronously from the virtual pool to the hiperpool using the Asynchronous Data Mover Facility |
| READ FAILURES | Number of page read failures that occurred using the Asynchronous Data Mover Facility |
| WRITE FAILURES | Number of page write failures using that occurred the Asynchronous Data Mover Facility |

***Parallel processing activity (message DSNB440I):***

| | |
|---|---|
| PARALLEL REQUEST | Number of negotiations for task streams for parallel processing activity |

DEGRADED PARALLEL
: Number of times negotiation resulted in a degraded mode of operation

## Examples

***Example 1:*** A summary report is the default report if the DETAIL option is not specified. The following is an example of a summary report which could be produced by the command:

```
-DIS BUFFERPOOL(BP0) LIST(*) DBNAME(DSN8*)
```

```
   DSNB401I - BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 20
   DSNB402I - VIRTUAL BUFFERPOOL SIZE = 500 BUFFERS 736
                ALLOCATED       =     500   TO BE DELETED   =        0
                IN-USE/UPDATED  =      0
   DSNB403I - HIPERPOOL SIZE = 10000 BUFFERS, CASTOUT = YES
                ALLOCATED       =   0   TO BE DELETED   =        0
                BACKED BY ES    =    0
   DSNB404I - THRESHOLDS - 739
                VP SEQUENTIAL         = 80   HP SEQUENTIAL         = 75
                DEFERRED WRITE        = 85   VERTICAL DEFERRED WRT = 80,0
                PARALLEL SEQUENTIAL   = 50 ASSISTING PARALLEL SEQT  = 0
   DSNB406I - VIRTUAL BUFFERPOOL TYPE = -737
                CURRENT             = PRIMARY
                PENDING               = PRIMARY
              PAGE STEALING METHOD = LRU
   DSNB460I -   740
```

***Example 2:*** A detail report can be generated that includes all summary report information and additional buffer pool related statistics. The following is an example of a detail report that could be produced by the command:

```
-DISPLAY BUFFERPOOL(BP0) DETAIL
```

```
DSNB401I - BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 10
DSNB402I - VIRTUAL BUFFERPOOL SIZE = 1000 BUFFERS
             ALLOCATED       =     1000   TO BE DELETED   =        0
             IN-USE/UPDATED  =      200
DSNB403I - HIPERPOOL SIZE = 600000 BUFFERS, CASTOUT = YES
             ALLOCATED       =   600000   TO BE DELETED   =        0
             BACKED BY ES    =   483651
DSNB404I - THRESHOLDS -
             VP SEQUENTIAL         = 80   HP SEQUENTIAL         = 80
             DEFERRED WRITE        = 50   VERTICAL DEFERRED WRT = 10
             PARALLEL SEQUENTIAL   = 50
DSNB405I - HIPERSPACE NAMES - @001SSOP @002SSOP

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

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
DSNB421I - DWT HIT           =        0 VERTICAL DWT HIT  =           0
           NO WRITE ENGINE   =        0

DSNB430I - HIPERPOOL ACTIVITY (NOT USING ASYNCHRONOUS
           DATA MOVER FACILITY) -
             SYNC HP READS   =      100   SYNC HP WRITES  =           0
             ASYNC HP READS  =        0   ASYNC HP WRITES =           0
             READ FAILURES   =        0   WRITE FAILURES  =           0

DSNB431I - HIPERPOOL ACTIVITY (USING ASYNCHRONOUS
           DATA MOVER FACILITY) -
             HP READS        =      244   HP WRITES       =           3
             READ FAILURES   =        0   WRITE FAILURES  =           0

DSNB440I - PARALLEL ACTIVITY -
             PARALL REQUEST  =        0   DEGRADED PARALL =       0

DSN9022I - DSNB1CMD '-DISPLAY BUFFERPOOL' NORMAL COMPLETION
```

***Example 3:*** 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
```

```
DSNB401I - BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 3
DSNB402I - VIRTUAL BUFFERPOOL SIZE = 1000 BUFFERS
             ALLOCATED       =     1000   TO BE DELETED   =        0
             IN-USE/UPDATED  =      200
DSNB403I - HIPERPOOL SIZE = 100000 BUFFERS, CASTOUT = YES
             ALLOCATED       =   100000   TO BE DELETED   =        0
             BACKED BY ES    =    89152
DSNB404I - THRESHOLDS -
             VP SEQUENTIAL         = 80   HP SEQUENTIAL         = 80
             DEFERRED WRITE        = 50   VERTICAL DEFERRED WRT = 10
             PARALLEL SEQUENTIAL   = 50
DSNB405I - HIPERSPACE NAMES - @001SSOP
DSNB455I -   SYNCHRONOUS I/O DELAYS -
               AVERAGE DELAY  =     22   MAXIMUM DELAY      =    35
               TOTAL PAGES    =     23
DSN9022I - DSNB1CMD '-DISPLAY BUFFERPOOL' NORMAL COMPLETION
```

## -DISPLAY DATABASE (DB2)

The DB2 command DISPLAY DATABASE displays information about the status of DB2 databases, table spaces, tables in segmented table spaces, LOB table spaces, index spaces within a database, indexes on auxiliary tables, and partitions of partitioned table spaces and 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 or auxiliary warning advisory status.

In a data sharing environment, the command can be issued from any DB2 in the group that has access to the database.

**Abbreviation**: -DIS DB

### Environment

This command can be issued from an MVS 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

No special privilege is required to issue -DISPLAY DATABASE.

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:

- DISPLAYDB privilege
- DISPLAY privilege
- DBMAINT, DBCTRL, or DBADM authority
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
                         ┌─,─────────────┐
►►──DISPLAY DATABASE──(──▼─database-name─┴──────┬──)──────────────────────────────────────────────►
                        ├─*─────────────────────┤
                        ├─dbname1:dbname2───────┤
                        ├─dbname*───────────────┤
                        ├─*dbname───────────────┤
                        ├─*dbname*──────────────┤
                        └─*dbstring1*dbstring2*─┘

►──┬──────────────────────────────────────────────────────────────────────────────────┬───────────►
   │ ├─USE──────┤                                                                        │
   │ ├─CLAIMERS─┤                                                                        │
   │ ├─LOCKS────┤                                                                        │
   │ ├─LPL──────┤                                                                        │
   │ └─WEPR─────┘                                                                        │
   │                  ┌─,──────────┐                                                     │
   ├─SPACENAM──(──┬──▼─space-name─┴─────────────┬──)──┬─────────────────────────┬───────┤
   │              ├─*───────────────────────────┤      │ ┌──────────┐ ┌──────┐ │       │
   │              ├─spacename1:spacename2───────┤      └─┼─USE──────┼─┴─ONLY─┘─┘       │
   │              ├─spacename*──────────────────┤        ├─CLAIMERS─┤                    │
   │              ├─*spacename──────────────────┤        ├─LOCKS────┤                    │
   │              ├─*spacename*─────────────────┤        ├─LPL──────┤                    │
   │              └─*spacestring1*spacestring2*─┘        └─WEPR─────┘                    │
   └─ONLY─(1)──────────────────────────────────────────────────────────────────────────┘

►──┬───────────────────────────────────────────┬──┬─────────────────────────────┬────────────────►
   │        ┌─,───────┐                        │  │          ┌─50──────┐        │
   └─PART(──┬─▼─integer─┴───────────┬──)──────┘  └─LIMIT(──┼─integer─┼──)────┘
            └─integer1:integer2─────┘                        └─*───────┘

►──┬───────┬──┬────────┬──┬──────────┬──┬──────────┬──┬───────┬───────────────────────────────►◄
   └─AFTER─┘  └─ACTIVE─┘  └─RESTRICT─┘  └─ADVISORY─┘  └─AREST─┘
```

**Note:**

1 If you specify the ONLY option without the SPACENAM() keyword, only the LIMIT, AFTER, and RESTRICT keywords apply.

## 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 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... |
|---|---|
| dbname1:dbname2 | All databases whose names 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**

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[3]

**(*)** Displays information about all table spaces and index spaces of the specified database.

*spacename* and *spacestring* can have any of the forms 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... |
|---|---|
| spacename1::spacename2 | All table spaces or index spaces whose names 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* |

**USE**

Displays information about the following:

- The applications and subsystems of the database or space that has internal DB2 resources allocated
- The applications and subsystems of the database or space on whose behalf locks for the space are held or waited upon
- 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

[3] Consecutive pattern-matching characters (*) are not allowed, and you cannot specify two pattern-matching characters in the middle of a keyword string.

**CLAIMERS**
Displays information about the following:

- 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 the claims associated with them

CLAIMERS overrides both LOCKS and USE. If you specify CLAIMERS, any references to LOCKS or USE are ignored.

**LOCKS**
Displays information about the following:

- The applications and subsystems on whose behalf locks are held, waited upon, 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 associated with them
- The retained locks for a resource
- The page set or partition physical locks (P-locks) for a resource

LOCKS overrides USE. If both LOCKS and USE are specified, USE is ignored.

For a description of DB2 locking, see Section 5 (Volume 2) of *DB2 Administration Guide*.

**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 either:

- A list of one or more partitions, or
- A range of all partition numbers that collate greater than or equal to *integer1* and less than or equal to *integer2*

Both a list and a range cannot be specified.

**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.

**Abbreviation**: RES

Use of a database is restricted if the database is in any one 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:

- It is in one of the three situations above.
- 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).

The DISPLAY DATABASE RESTRICT command does not display the informational COPY pending advisory status; use the DISPLAY DATABASE ADVISORY command instead.

**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.

For information about resetting an advisory status, see Section 2 of *DB2 Utility Guide and Reference*.

**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 state. If backout activity against the object is not already underway, it should be initiated either by issuing the RECOVER POSTPONED command, or by recycling the system with the system parameter LBACKOUT=AUTO.

## 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. See Appendix D of *DB2 SQL Reference* for a list of table space names and assigned tables.

If you issue DISPLAY DATABASE LOCKS on the catalog (DSNDB06), you may see a lock held on SYSDBASE with the correlation ID 020.DBCMD_05 or 020.DBCMD_06. This simply 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. For information about reset-
| ting a restrictive or advisory status, see Section 2 of *DB2 Utility Guide and Refer-*
| *ence*.

***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, then 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.

# ***Displaying databases for declared temporary tables:*** DISPLAY DATABASE dis-
# plays information about databases that are created with the AS TEMP option and
# the associated table spaces, but does not display information for declared tempo-
# rary tables or index spaces that the database contains.

## Output

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

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

| | |
|---|---|
| GRECP | The object is GBP-dependent and a group buffer pool RECOVER is pending. |
| ICOPY | The index space is in the informational COPY-pending advisory status. |
| LPL | The object has entries in the logical page list. |
| LSTOP | The logical partition of a nonpartitioning index is stopped. |
| PSRBD | The entire nonpartitioning index space is in a page set REBUILD pending status. |
| RBDP | The physical or logical index partition is in the REBUILD pending status. |
| RBDP* | 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. |
| RECP | The object (a table space, table space partition, index space, index partition, or logical index partition) is in the RECOVER-pending status. |
| REORP | The data partition is in the REORG-pending status. |
| REST | The page set or partition is being restarted. |
| RESTP | The table space or index space is in the restart-pending status. |
| RO | The database, table space, table space partition, index space, or index space partition is started for read-only activity. |
| RW | The database, table space, table space partition, index space, or index space partition is started for read and write activity. |
| STOP | The database, table space, table space partition, index space, or index space partition is stopped. |
| STOPE | 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. |
| STOPP | A stop is pending for the database, table space, table space partition, index space, or index space partition. |
| UT | The database, table space, table space partition, index space, or index space partition is started for utility processing only. |
| UTRO | 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. |

## Examples

***Example 1:*** Display information about table space TBS33 in database CB3. USE causes *connection-name*(CONNID), *correlation-id*(CORRID), and *authorization ID* (USERID) information to be displayed.

```
-DISPLAY DATABASE(CB3) SPACENAM(TBS33) USE
```

The following output is generated:

```
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    01  RW                LSS001   DSN2SQL      SYSADM
TBS33    TS    02  RW                LSS001   DSN2SQL      SYSADM
TBS33    TS    03  RW                LSS001   DSN2SQL      SYSADM
TBS33    TS    04  RW                LSS001   DSN2SQL      SYSADM
******* DISPLAY OF DATABASE CB3      ENDED      *********************
DSN9022I . DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
```

***Example 2:*** Display information about table space TBS33 in database CB3 when the table space is defined with LOCKPART YES. LOCKS 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 following output is generated:

```
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    01  RW
TBS33    TS    02  RW
TBS33    TS    03  RW
TBS33    TS    04  RW                LSS004   DSN2SQL      H(IS,S,C)
TBS33    TS    04  RW                LSS005   DSN2SQL      H(IS,S,C)
******* DISPLAY OF DATABASE CB3      ENDED      *********************
```

***Example 3:*** Display information about table space TBS33 in database CB3. CLAIMERS 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 following output is generated:

```
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    01  RW
TBS33    TS    02  RW
TBS33    TS    03  RW
TBS33    TS    04  RW                 LSS001   DSN2SQL      (RR,C)
TBS33    TS    04  RW                 LSS001   DSN2SQL      (WR,C)
******* DISPLAY OF DATABASE CB3      ENDED      *********************
```

***Example 4:*** In a data sharing environment, display information about locks held when the table space is defined with LOCKPART YES. The application 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(DSN8D51A) SPACENAM(TSPART) LOCKS
```

# Output similar to the following output is generated:

```
NAME     TYPE PART STATUS             CONNID   CORRID       LOCKINFO
-------- ---- ---- ------------------ -------- ------------ ---------
TSPART   TS    01  RO                 LSS001   DSN2SQL      H-IS,P,C
    -                  MEMBER NAME DB1G
TSPART   TS    01  RO                                       H-S,PP,I
    -                  MEMBER NAME DB1G
TSPART   TS    01  RO                 LSS002   DSN2SQL      H-IS,P,C
    -                  MEMBER NAME DB2G
TSPART   TS    01  RO                                       H-S,PP,I
    -                  MEMBER NAME DB2G
TSPART   TS    02  RW                 LSS001   DSN2SQL      H-IS,P,C
    -                  MEMBER NAME DB1G
TSPART   TS    02  RW                                       H-S,PP,I
    -                  MEMBER NAME DB1G
TSPART   TS    03  RW                 LSS002   DSN2SQL      H-IS,P,C
    -                  MEMBER NAME DB2G
TSPART   TS    03  RW                                       H-S,PP,I
    -                  MEMBER NAME DB2G
TSPART   TS    04  RW
```

If the table space is defined with LOCKPART NO, the display looks like this. 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    01  RO                                      H-S,PP,I
    -               MEMBER NAME DB1G
TSPART   TS    02  RW                                      H-S,PP,I
    -               MEMBER NAME DB2G
TSPART   TS    03  RW                                      H-S,PP,I
    -               MEMBER NAME DB2G
TSPART   TS    04  RW
```

***Example 5:*** Display information about page sets in database DSNDB01 that have entries in the logical page list. Limit the number of messages displayed to the space available.

```
-DB1G DISPLAY DATABASE(DSNDB01) SPACENAM(*) LIMIT(*) LPL
```

# Output similar to the following output is generated:

```
***********************************************************
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
----                                 000039-00003C
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 6:*** Suppose that table space TSPART, which is in database DSN8D61A, is defined with the keyword LOCKPART NO, which means that DB2 does not do selective partition locking on TSPART. When you specify this command:

```
-DB1G DISPLAY DATABASE(DSN8D61A) SPACE(TSPART) PART(1,4) LOCKS
```

two applications are accessing TSPART, and the partitions have different statuses. In the output, DB2 displays the locks as table space locks, as shown here:

```
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    01  RO
TSPART   TS    04  RW
```

***Example 7:*** 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 specify this command:

```
-DB1G DISPLAY DATABASE(DSN8D61A) SPACE(TSPART) PART(1:4) LOCKS
```

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. In the output, DB2 displays the locks as partition locks, as shown here:

```
NAME     TYPE PART STATUS            CONNID   CORRID       LOCKINFO
-------- ---- ---- ----------------- -------- ------------ ---------
TSPART   TS    01  RO                LSS001   DSN2SQL      H-IS,P,C
TSPART   TS    01  RO                LSS002   DSN2SQL      H-IS,P,C
TSPART   TS    02  RW                LSS001   DSN2SQL      H-IS,P,C
TSPART   TS    03  RW                LSS002   DSN2SQL      H-IS,P,C
TSPART   TS    04  RW
```

| ***Example 8:*** Display information about all table spaces and index spaces in the range of databases from DBKD0101 to DBKD0106 that are in a restrictive status. Limit the number of messages that are displayed to the available space.

```
-DISPLAY DATABASE(DBKD0101,DBKD0103) SPACENAM(*) RESTRICT LIMIT(*)
```

| Output similar to the following output is generated:

```
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 9:*** Display information about all table spaces that are in the auxiliary warning advisory status (AUXW), and all index spaces that are in informational COPY pending status (ICOPY) in database DBIQUQ01. Limit the number of messages that are displayed to the available space.

```
-DISPLAY DATABASE(DBIQUQ01) SPACENAM(*) LIMIT(*) ADVISORY
```

Output similar to the following output is generated:

```
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    001 RW,AUXW
TPIQUQ01 TS    002 RW,AUXW
TPIQUQ01 TS    003 RW,AUXW
TPIQUQ01 TS    004 RW,AUXW
IAIQUQ01 IX        RW,ICOPY
IAIQUQ02 IX        RW,ICOPY
IAIQUQ03 IX        RW,ICOPY
IAIQUQ04 IX        RW,ICOPY
IPIQUQ01 IX    001 RW,ICOPY
IPIQUQ01 IX    002 RW,ICOPY
IPIQUQ01 IX    003 RW,ICOPY
IPIQUQ01 IX    004 RW,ICOPY
IUIQUQ03 IX        RW,ICOPY
IXIQUQ02 IX        RW,ICOPY
******* DISPLAY OF DATABASE DBIQUQ01 ENDED      **********************
DSN9022I - DSNTDDIS 'DISPLAY DATABASE' NORMAL COMPLETION
```

## -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 an MVS 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, the privilege set of the process must include one of the following authorities for each function:

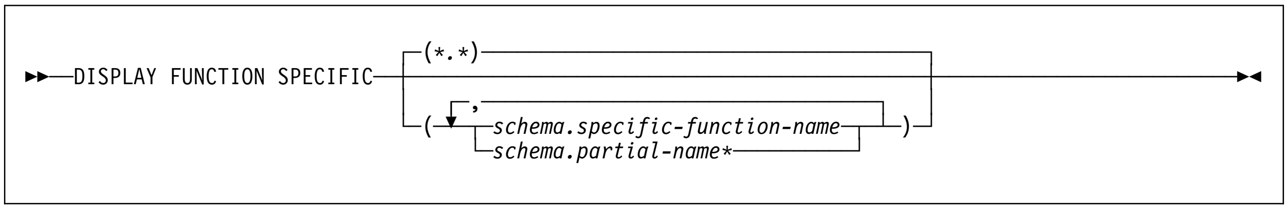
- DISPLAY privilege
- Ownership of the function
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

If -DISPLAY FUNCTION SPECIFIC *.* or schema.partial-name* is specified, the privilege set of the process must include one of the following authorities:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## 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 have been accessed by DB2 applica-
| tions since the DB2 subsystem was started.

## | Usage notes

| 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.

| This command does not apply to built-in functions or user-defined functions that are
| sourced on another function.

## | Output

| This command displays one output line 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 fol-
| lowing output:

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

------- SCHEMA=PAYROLL
FUNCTION           STATUS ACTIVE QUEUED MAXQUE TIMEOUT   WLM_ENV
APPL1              STARTED     1      0      0       0   PAYROLL
APPL2              STARTED     1      0      0       0   PAYROLL
APPL3              STARTED     0      1      2       0   PAYROLL
APPL5              STOPREJ     0      0      0       0   SANDBOX
APPL6              STOPABN     0      0      0       0   PAYROLL
FUNC1              STOPQUE     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. |
| QUEUED | The number of threads that are waiting for the function to be scheduled. |
| MAXQUE | 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. |
| WLM_ENV | The WLM environment where the function executes. |

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 "-STOP FUNCTION SPECIFIC" on page 322 for more information.

## Examples

***Example 1:*** Display information about functions in the PAYROLL schema and the HRPROD schema.

```
-DISPLAY FUNCTION SPECIFIC(PAYROLL.*, HRPROD.*)
```

This command produces output similar to the following output:

```
DSNX975I csect - DISPLAY FUNCTION SPECIFIC REPORT FOLLOWS-

------ SCHEMA=PAYROLL
FUNCTION          STATUS   ACTIVE  QUEUED  MAXQUE  TIMEOUT   WLM_ENV
PAYRFNC1          STARTED       0       0       1        0    PAYROLL
PAYRFNC2          STOPQUE       0       5       5        3    PAYROLL
PAYRFNC3          STARTED       2       0       6        0    PAYROLL
USERFNC4          STOPREJ       0       0       1        0    SANDBOX

------ SCHEMA=HRPROD
FUNCTION          STATUS   ACTIVE  QUEUED  MAXQUE  TIMEOUT   WLM_ENV
HRFNC1            STARTED       0       0       1        0    HRFUNCS
HRFNC2            STOPREJ       0       0       1        0    HRFUNCS
DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY FUNC' NORMAL COMPLETION
```

***Example 2:*** Display information about specific functions in the PAYROLL schema.

```
-DISPLAY FUNCTION SPECIFIC(PAYROLL.USERFNC2,PAYROLL.USERFNC4)
```

This command produces output similar to the following output:

```
DSNX975I csect - DISPLAY FUNCTION SPECIFIC REPORT FOLLOWS-

------ SCHEMA=PAYROLL
FUNCTION           STATUS   ACTIVE  QUEUED  MAXQUE  TIMEOUT   WLM_ENV
USERFNC2           STOPQUE       0       5       5       3    SANDBOX
USERFNC4           STOPREJ       0       0       1       0    SANDBOX
DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY FUNC' NORMAL COMPLETION
```

***Example 3:*** Display information about all functions in the PAYROLL schema that DB2 applications have accessed. This example assumes that the STOP FUNCTION SPECIFIC(PAYROLL.*) ACTION(QUEUE) command is in effect at the time you issue the DISPLAY FUNCTION SPECIFIC command:

```
-DISPLAY FUNCTION SPECIFIC(PAYROLL.*)
```

This command produces output similar to the following output:

```
DSNX975I csect - DISPLAY FUNCTION SPECIFIC REPORT FOLLOWS-

------ SCHEMA=PAYROLL
FUNCTION           STATUS   ACTIVE  QUEUED  MAXQUE  TIMEOUT   WLM_ENV
USERFNC2           STOPQUE       0       5       5       3    SANDBOX
USERFNC4           STOPQUE       0       0       1       0    SANDBOX
FUNCTIONS USERFNC2           - USERFNC29999999999 STOP QUEUE
FUNCTIONS USERFNC4           - USERFNC49999999999 STOP QUEUE
DSNX9DIS DISPLAY FUNCTION SPECIFIC REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY FUNC' NORMAL COMPLETION
```

## -DISPLAY GROUP (DB2)

The DB2 command DISPLAY GROUP displays information about the data sharing group to which a DB2 subsystem belongs.

**Abbreviation**: -DIS GROUP

## Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following:

- DISPLAY privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
►►──DISPLAY GROUP──┬────────┬──────────────────────────────────────────────►◄
                   └─DETAIL─┘
```

## Option descriptions

**DETAIL**
Displays information about the parallelism coordinator and parallelism assistant.

## 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, or FAILED.

ACTIVE indicates that the DB2 subsystem is active, and FAILED indicates that it is failed. A QUIESCED status results from a STOP DB2 command and consists of several subcategories:

**QUIESCED** This is 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 retained locks are held.

**C** There was a castout error. The last updater of the page set or partition could not write from the coupling facility to DASD.

Make sure there are no connectivity problems between the coupling facility and the processor before restarting DB2.

**R** There is retained information needed for DB2 to perform resynchronization with one or more remote locations.

When DB2 is restarted, this resynchronization occurs.

| | |
|---|---|
| **ACTIVE** | This is a normal active state without conditions. |
| **A** | A (active) can be paired with the following letter: |

**I** Indoubt or postponed abort units of recovery (URs) are outstanding. This means 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.

## Output

***DISPLAY GROUP command output:*** The DISPLAY GROUP command displays the following output:

| | |
|---|---|
| *** BEGIN | The name of the DB2 group |
| DB2 MEMBER | The names of its members |
| ID | The IDs of its members |
| SUBSYS | The subsystem names of its members |
| CMDPREF | The command prefix for each member |
| STATUS | The status of each member (ACTIVE, QUIESCED with or without additional conditions, or FAILED) |
| SYSTEM NAME | The names of the MVS system where the member is running, or was last running in cases when the member status is QUIESCED or FAILED |
| IRLM SUBSYS | The name of the IRLM subsystem to which the DB2 member is connected |
| IRLMPROC | The procedure names of the connected IRLM |
| SYSTEM NAME | The MVS system name where the data sharing member runs on. |
| LVL | A string of three numeric characters that list as follows: • DB2 version • DB2 release • DB2 modification level |
| SCA | The SCA structure size in KB and the percentage currently in use |
| LOCK1 | The LOCK1 structure size in KB. |

The display also shows the following:

- The maximum number of lock entries possible for the lock table and how many of those lock entries are currently in use. This number is an approximate value.

- The maximum number of modify lock list entries and how many of those list entries are currently in use.

For more information about the lock table and the list of modify locks, see Chapter 7 of *DB2 Data Sharing: Planning and Administration*.

PARALLELISM COORDINATOR
Indicates whether this DB2 member can coordinate parallel processing.

PARALLELISM ASSISTANT
Indicates whether this DB2 member can assist with parallel processing.

# If the output indicates that either the lock structure or SCA are 0% in use, that does
# not necessarily mean that the structure is empty. It could mean that the structures
# are very large and that the number of locks held or the number of records in the
# SCA is less than 1%.

***Description of message DSN7101I:***

GROUP The name of the data sharing group

GROUP LEVEL
A string of three numeric characters that denotes:

- DB2 version
- DB2 release
- DB2 modification level

This is the highest release with which any DB2 in the data sharing group have been started.

## Examples

***Example 1:*** The following sample output for a data sharing group can be generated by the command:

```
-DB1A DIS GROUP
```

```
DSN7100I -DB1A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DSNDB10 ) GROUP LEVEL(610)
                                     GROUP ATTACH NAME(DB10)
--------------------------------------------------------------------
DB2                                    DB2 SYSTEM    IRLM
MEMBER   ID  SUBSYS CMDPREF   STATUS   LVL NAME      SUBSYS IRLMPROC
-------- --- ----   --------  -------- --- --------  ----   --------
DB1A       1 DB1A   -DB1A     ACTIVE   610 MVSA      DJ1A   DB1AIRLM
DB1B       2 DB1B   -DB1B     ACTIVE   610 MVSB      DJ1B   DB1BIRLM
DB1C       3 DB1C   -DB1C     ACTIVE   610 MVSC      DJ1C   DB1CIRLM
DB1D       4 DB1D   -DB1D     FAILED   610 MVSD      DJ1D   DB1DIRLM
DB1E       5 DB1E   -DB1E     QUIESCED 610 MVSE      DJ1E   DB1EIRLM
DB1F       6 DB1F   -DB1F     ACTIVE   610 MVSF      DJ1F   DB1FIRLM
DB1G       7 DB1G   -DB1G     ACTIVE   610 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 2:*** In a non-data-sharing environment, the following sample output is generated by the command:

```
-DB1A DISPLAY GROUP
```

```
DSN7100I -DB1A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(.......) GROUP LEVEL(...)
                                    GROUP ATTACH NAME(....)
--------------------------------------------------------------------
DB2                                    DB2 SYSTEM    IRLM
MEMBER   ID  SUBSYS CMDPREF   STATUS   LVL NAME      SUBSYS IRLMPROC
-------- --- ----   --------  -------- --- --------  ----   --------
........   0 DB1A   -DB1A     ACTIVE   610 MVSA      DJ1A   DB1AIRLM
--------------------------------------------------------------------
*** END DISPLAY OF GROUP(DSNDB10)
DSN9022I -DB1A DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION
```

***Example 3:*** Using the DETAIL option, you can obtain more information about the data sharing group as shown in the following example using the command:

```
-DB1A DIS GROUP DETAIL
```

```
DSN7100I -DB1A DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DSNCAT ) GROUPLEVEL(610)
--------------------------------------------------------------------
DB2                                     SYSTEM         IRLM
MEMBER   ID  SUBSYS CMDPREF   STATUS   NAME     LVL  SUBSYS IRLMPROC
-------- --- ----   --------  -------- -------- ---  ----   --------
DB1A       1 DB1A   -DB1A     ACTIVE   MVSA     610  AR21   ARLM21
DB1B       2 DB1B   -DB1B     ACTIVE   MVSB     610  BR21   BRLM21
DB1C       3 DB1C   -DB1C     ACTIVE   MVSC     510  CRLM   CRLM21
DB2D       4 DB2D   -DB2D     FAILED   MVSD     610  DR21   DRLM21
DB2E       5 DB2E   -DB2E     QUIESCED MVSE     610  ER21   ERLM21
DB2F       6 DB2F   -DB2F     ACTIVE   MVSF     610  FR21   FRLM21
DB2G       7 DB2G   -DB2G     ACTIVE   MVSG     610  GR21   GRLM21
--------------------------------------------------------------------
DB2      PARALLEL    PARALLEL
MEMBER   COORDINATOR ASSISTANT
-------- ----------- ---------
DB2A             YES        NO
DB2B             YES       YES
DB2B             YES       YES
DB1C            ****      ****
DB2D            ****      ****
DB2E            ****      ****
DB2F              NO       YES
DB2G              NO        NO
--------------------------------------------------------------------
SCA   STRUCTURE SIZE:     1024 KB, STATUS= AC,   SCA IN USE:    11 %
LOCK1 STRUCTURE SIZE:     1536 KB,            LOCK1 IN USE: <   1 %
NUMBER  LOCK ENTRIES:      262144, LOCK ENTRIES  IN USE:         33
NUMBER  LIST ENTRIES:        7353, LIST ENTRIES  IN USE:          0
*** END DISPLAY OF GROUP(DSNCAT )
DSN9022I -DB1A DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION
```

## -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 an MVS 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, the privilege set of the process must include one of the following:

- DISPLAY privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
►►──DISPLAY GROUPBUFFERPOOL──┬──────────────────────────────────┬──┬───────────────────────────────┬──────────►
                             │ ┌─(*)───────────────────────────┐ │  └─TYPE──(──┬─*───────┬──)─┘
                             └─┴─(─▼─┬─gbpname────────┬──)───┴─┘             ├─GCONN───┤
                                  ┌─,                                        ├─MCONN───┤
                                     └─structure-name─┘                      └─NOCACHE─┘

►──┬──────────────────────────────────┬──┬──────────────────────────────────┬────────────────────────────────►
   └─MDETAIL──┬───────────────────┬──┘  └─GDETAIL──┬───────────────────┬──┘
              │   ┌─INTERVAL─┐    │                 │   ┌─INTERVAL─┐    │
              └─(─┴─*────────┴─)─┘                 └─(─┴─*────────┴─)─┘

►──┬────────────────────────────┬────────────────────────────────────────────────────────────────────────────►◄
   │                 ┌─NO──┐    │
   └─CONNLIST──(─────┴─YES─┴──)─┘
```

### Option descriptions

(*) 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, GBP1, ..., GBP49.
- 8-KB group buffer pools are named GBP8K0, GBP8K1, ... , GBP8K9.
- 16-KB group buffer pools are named GBP16K0, GBP16K1, ... , GBP16K9.
- 32-KB group buffer pools are named GBP32K, GBP32K1, ... , 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 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 re-allocated.

**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 three report types are:

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

These reports are described here.

### Summary report

You can display summary information about group buffer pools. The report indicates whether this DB2 is actively connected to the group buffer pools you requested information for. 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 non-volatile storage; however, it can allocate in a volatile structure.
- The actual number of directory entries, data pages, and connection 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 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: 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, CFLEVEL
: Indicates the name and 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 name and level associated with the primary group buffer pool.

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. See Chapter 7 of *DB2 Data Sharing: Planning and Administration* for more information about using this information. The report includes the same information as in the summary report with the addition of:

**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 may not be exactly the same, due to timing periods for gathering the counter information for display or previous transaction failures that may 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.

**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 DASD, 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 DASD 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 4 pages written per castout write I/O on average, then 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 it knows that there is no data 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 pagesets 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 usually 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 may not be exactly the same, due to timing periods for gathering the counter information for display or previous transaction failures that may 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 from the primary group buffer pool.

**READ CASTOUT STATISTICS**
Indicates the number of READ CASTOUT STATITICS 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 1:*** This is an example of a summary report that can be produced by the following command:

```
-DISPLAY GROUPBUFFERPOOL(GBP29)
```

Message DSNB799I is displayed if the group buffer pool is duplexed and the secondary group buffer pool is currently allocated. If a secondary group buffer pool is not allocated, message DSNB799I is not included in the output.

```
DSNB750I - DISPLAY FOR GROUP BUFFER POOL GBP29 FOLLOWS
DSNB755I - DB2 GROUP BUFFER POOL STATUS
             CONNECTED                                 = YES
             CURRENT DIRECTORY TO DATA RATIO           = 5
             PENDING DIRECTORY TO DATA RATIO           = 5
             CURRENT GBPCACHE ATTRIBUTE                = YES
             PENDING GBPCACHE ATTRIBUTE                = YES
DSNB756I -   CLASS CASTOUT THRESHOLD                   = 10%
             GROUP BUFFER POOL CASTOUT THRESHOLD       = 50%
             GROUP BUFFER POOL CHECKPOINT INTERVAL     = 8 MINUTES
             RECOVERY STATUS                           = NORMAL
             AUTOMATIC RECOVERY                        = Y
DSNB757I - MVS CFRM POLICY STATUS FOR DSNCAT_GBP29     = NORMAL
             MAX SIZE INDICATED IN POLICY              = 2048 KB
             DUPLEX INDICATOR IN POLICY                = ENABLED
             CURRENT DUPLEXING MODE                    = DUPLEX
             ALLOCATED                                 = YES
DSNB758I -     ALLOCATED SIZE                          = 2048 KB
               VOLATILITY STATUS                       = VOLATILE
               REBUILD STATUS                          = DUPLEXED
               CFNAME                                  = CACHE01
               CFLEVEL                                 = 5
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, 1997
             GBP CHECKPOINT RECOVERY LRSN              = AF6BBAEF3307
             STRUCTURE OWNER                           = V61B
DSNB799I - SECONDARY GBP ATTRIBUTES
             ALLOCATED SIZE                            = 2048 KB
             VOLATILITY STATUS                         = VOLATILE
             CFNAME                                    = LF01
             CFLEVEL                                   = 5
             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 2:*** Assume you want a summary report about group buffer pool twenty-nine, including all connections to that group buffer pool. Enter the following command:

```
-DISPLAY GROUPBUFFERPOOL(GBP29) CONNLIST(YES)
```

This command produces output similar to the following output:

```
DSNB750I - DISPLAY FOR GROUP BUFFER POOL GBP29 FOLLOWS
DSNB755I - DB2 GROUP BUFFER POOL STATUS
             CONNECTED                                 = YES
             CURRENT DIRECTORY TO DATA RATIO           = 5
             PENDING DIRECTORY TO DATA RATIO           = 5
             CURRENT GBPCACHE ATTRIBUTE                = YES
             PENDING GBPCACHE ATTRIBUTE                = YES
DSNB756I -   CLASS CASTOUT THRESHOLD                   = 10%
             GROUP BUFFER POOL CASTOUT THRESHOLD       = 50%
             GROUP BUFFER POOL CHECKPOINT INTERVAL     = 8 MINUTES
             RECOVERY STATUS                           = NORMAL
             AUTOMATIC RECOVERY                        = Y
DSNB757I - MVS CFRM POLICY STATUS FOR DSNCAT_GBP29     = NORMAL
             MAX SIZE INDICATED IN POLICY              = 2048 KB
             DUPLEX INDICATOR IN POLICY                = ENABLED
             CURRENT DUPLEXING MODE                    = SIMPLEX
             ALLOCATED                                 = YES
DSNB758I -     ALLOCATED SIZE                          = 2048 KB
               VOLATILITY STATUS                       = VOLATILE
               REBUILD STATUS                          = DUPLEXED
               CFNAME                                  = CACHE01
               CFLEVEL                                 = 5
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, 1997
             GBP CHECKPOINT RECOVERY LRSN              = AF6BBAEF3307
             STRUCTURE OWNER                           = V61B
DSNB799I - SECONDARY GBP ATTRIBUTES
             ALLOCATED SIZE                            = 2048 KB
             VOLATILITY STATUS                         = VOLATILE
             CFNAME                                    = LF01
             CFLEVEL                                   = 5
             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 3:*** This example shows a group detail report that is produced by the command:

```
-DISPLAY GROUPBUFFERPOOL(GBP29) GDETAIL(*)
```

Message DSNB762I is displayed in the output only if the secondary group buffer pool is allocated.

```
DSNB750I - DISPLAY FOR GROUP BUFFER POOL GBP29 FOLLOWS
DSNB755I - DB2 GROUP BUFFER POOL STATUS
             CONNECTED                                 = YES
             CURRENT DIRECTORY TO DATA RATIO           = 5
             PENDING DIRECTORY TO DATA RATIO           = 5
             CURRENT GBPCACHE ATTRIBUTE                = YES
             PENDING GBPCACHE ATTRIBUTE                = YES
DSNB756I -   CLASS CASTOUT THRESHOLD                   = 10%
             GROUP BUFFER POOL CASTOUT THRESHOLD       = 50%
             GROUP BUFFER POOL CHECKPOINT INTERVAL     = 8 MINUTES
             RECOVERY STATUS                           = NORMAL
             AUTOMATIC RECOVERY                        = Y
DSNB757I - MVS CFRM POLICY STATUS FOR DSNCAT_GBP29     = NORMAL
             MAX SIZE INDICATED IN POLICY              = 2048 KB
             DUPLEX INDICATOR IN POLICY                = ENABLED
             CURRENT DUPLEXING MODE                    = DUPLEX
             ALLOCATED                                 = YES
DSNB758I -     ALLOCATED SIZE                          = 2048 KB
               VOLATILITY STATUS                       = VOLATILE
               REBUILD STATUS                          = DUPLEXED
               CFNAME                                  = CACHE01
               CFLEVEL                                 = 5
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, 1997
             GBP CHECKPOINT RECOVERY LRSN              = AF6BBAEF3307
             STRUCTURE OWNER                           = V61B
DSNB799I - SECONDARY GBP ATTRIBUTES
             ALLOCATED SIZE                            = 2048 KB
             VOLATILITY STATUS                         = VOLATILE
             CFNAME                                    = LF01
             CFLEVEL                                   = 5
             NUMBER OF DIRECTORY ENTRIES               = 1950
             NUMBER OF DATA PAGES                      = 389
DSNB783I - CUMULATIVE GROUP DETAIL STATISTICS SINCE 17:08:35 OCT 16,
1997
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 4:*** This example shows the member detail section from the report that is produced by the command:

```
-DISPLAY GROUPBUFFERPOOL(GBP29) MDETAIL(*)
```

Messages DSNB764I and DSNB793I are displayed in the output only if the secondary group buffer pool is allocated.

```
DSNB750I - DISPLAY FOR GROUP BUFFER POOL GBP29 FOLLOWS
DSNB755I - DB2 GROUP BUFFER POOL STATUS
             CONNECTED                                 = YES
             CURRENT DIRECTORY TO DATA RATIO           = 5
             PENDING DIRECTORY TO DATA RATIO           = 5
             CURRENT GBPCACHE ATTRIBUTE                = YES
             PENDING GBPCACHE ATTRIBUTE                = YES
DSNB756I -   CLASS CASTOUT THRESHOLD                   = 10%
             GROUP BUFFER POOL CASTOUT THRESHOLD       = 50%
             GROUP BUFFER POOL CHECKPOINT INTERVAL     = 8 MINUTES
             RECOVERY STATUS                           = NORMAL
             AUTOMATIC RECOVERY                        = Y
DSNB757I - MVS CFRM POLICY STATUS FOR DSNCAT_GBP29     = NORMAL
             MAX SIZE INDICATED IN POLICY              = 2048 KB
             DUPLEX INDICATOR IN POLICY                = ENABLED
             CURRENT DUPLEXING MODE                    = DUPLEX
             ALLOCATED                                 = YES
DSNB758I -     ALLOCATED SIZE                          = 2048 KB
               VOLATILITY STATUS                       = VOLATILE
               REBUILD STATUS                          = DUPLEXED
               CFNAME                                  = CACHE01
               CFLEVEL                                 = 5
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, 1997
             GBP CHECKPOINT RECOVERY LRSN              = AF6BBAEF3307
             STRUCTURE OWNER                           = V61B
DSNB799I - SECONDARY GBP ATTRIBUTES
             ALLOCATED SIZE                            = 2048 KB
             VOLATILITY STATUS                         = VOLATILE
             CFNAME                                    = LF01
             CFLEVEL                                   = 5
             NUMBER OF DIRECTORY ENTRIES               = 1950
             NUMBER OF DATA PAGES                      = 389
DSNB772I - CUMULATIVE MEMBER DETAIL STATISTICS SINCE 17:08:41 OCT 16,
1997
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
               REGISTER PAGE LIST NOT AVAILABLE
                 DATA RETURNED                         = 0
DSNB789I -    REGISTER PAGE LIST                       = 0
               RETRIEVE CHANGED PAGES                  = 0
               RETRIEVE CLEAN PAGES                    = 0
          FAILED READS DUE TO LACK OF STORAGE          = 0
DSNB776I -   SYNCHRONOUS WRITES
               CHANGED PAGES                           = 5
               CLEAN PAGES                             = 0
DSNB777I -   ASYNCHRONOUS WRITES
               CHANGED PAGES                           = 0
               CLEAN PAGES                             = 0
             FAILED WRITES DUE TO LACK OF STORAGE      = 0
```

```
    DSNB778I - CASTOUT THRESHOLDS DETECTED
                 FOR CLASSES                              = 0
                 FOR GROUP BUFFER POOL                    = 0
               GBP CHECKPOINTS TRIGGERED                  = 0
               PARTICIPATION IN REBUILD                   = 1
    DSNB796I - CASTOUTS
                 PAGES CASTOUT                            = 0
                 UNLOCK CASTOUT                           = 0
                 READ CASTOUT CLASS                       = 0
                 READ CASTOUT STATISTICS                  = 0
                 READ DIRECTORY INFO                      = 0
    DSNB779I -   ENGINES NOT AVAILABLE
                   FOR CASTOUT                            = 0
                   FOR WRITING                            = 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
                   CHANGED PAGES                          = 5
                   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
```

## -DISPLAY LOCATION (DB2)

If you specify the DETAIL option, which is optional, each line could be followed by information regarding conversations owned by DB2 system threads that are communicating with the 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 an MVS 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, the privilege set of the process must include one of the following:

- DISPLAY privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
►►──DISPLAY LOCATION──┬─(*)────────────────────────────────┬──┬────────┬──────────────────────────►◄
                      │                                    │  └─DETAIL─┘
                      │    ┌─,──────────────────────┐      │
                      └─(──▼─┬─location-name───────┴────)─┘
                             ├─partial-location*─┤
                             ├─<luname>──────────┤
                             └─ipaddr────────────┘
```

## Option descriptions

# **(*)** Displays information for all remote locations.

**(*location-name*)**
Lists one or more location names, separated by commas. If *location-name* is not specified, information for all remote locations is displayed.

Because DB2 does not receive a location name from requesters that are not DB2 for OS/390 subsystems, you can enter the LUNAME or IP address of such a requester. Enclose the LUNAME by the less-than (<) and greater-than (>) symbols. Enter the IP address in the form *nnn.nnn.nnn.nnn*. For example, -DISPLAY LOCATION(<LULA>) displays information about a remote location (that is not DB2 for OS/390) with the LUNAME of LULA.

-DISPLAY LOCATION(124.63.51.17) displays information about clients at the remote TCP/IP host whose dotted decimal IP address is 124.63.51.17.

**(***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'.

You can use an asterisk (*) when specifying a LUNAME in the same manner as described above for location names that are not DB2 for OS/390 subsystems. For example, -DISPLAY LOCATION(<LULA*) selects all remote locations (that are not DB2 for OS/390) with an LUNAME that begins with the string 'LULA'.

*<luname>*
Requests information about the remote clients that are connected to DDF through the remote SNA LU that is specified.

**(***ipaddr***)**
Requests information about the clients that are connected to DDF through the remote TCP/IP host. *nnn.nnn.nnn.nnn* is the dotted decimal IP address.

**DETAIL**
Displays additional information about conversation activity for DB2 system threads, as shown in Example 2 on page 153.

## Output

The DISPLAY LOCATION command displays the following output:

| | |
|---|---|
| LOCATION | The LOCATION of the remote system. |
| PRDID | The product identifier (PRDID) of the remote system. The PRDID is displayed in the form *nnnvvrrm*, where: |

| | |
|---|---|
| *nnn* | The database product |
| *vv* | The product version |
| *rr* | The product release |
| *m* | The product modification level |

| | |
|---|---|
| LINKNAME | The address (LU name or IP address) of the remote system. |
| REQUESTERS | The number of active threads from the local subsystem that are accessing the remote system. |
| SERVERS | The number of threads from the remote system that are accessing the local subsystem. |
| CONVERSATIONS | The total number of conversations or sockets related to the partner system. |

## Examples

***Example 1:*** Display information about threads and conversations with specific remote locations, using the following command:

```
-DISPLAY LOCATION(SAN_JOSE,SAN_FRANCISCO)

DSNL200I - DISPLAY LOCATION REPORT FOLLOWS-
LOCATION       PRDID       LINKNAME          REQUESTERS SERVERS CONVS
SAN_JOSE       DSN05010    LUND1                 1         0      1
SAN_FRANCISCO  DSN05010    LUND3                 1         0      1
DISPLAY LOCATION REPORT COMPLETE
```

***Example 2:*** Display information about threads and conversations with all remote locations. Additionally, display detail conversation information about DB2 system threads that communicate with other locations. This is an example of the output generated by the following command:

```
-DISPLAY LOCATION DETAIL

DSNL200I - DISPLAY LOCATION REPORT FOLLOWS-
LOCATION          PRDID     LINKNAME          REQUESTERS SERVERS CONVS
SAN_JOSE          DSN05010  LUND1                 1         0      3
 -SYSTASK   SESSID           A ST TIME
 -SYSCON-O  00D359691359EE80   S  9128009214880
 -SYSCON-I  00D359691359EE81 W R  9128009214881
MENLO_PARK        DSN05010  LUND2                 1         0      4
 -SYSTASK   SESSID           A ST TIME
 -SYSCON-O  00D359691359EE82   S  9128009214882
 -SYSCON-I  00D359691359EE83 W R  9128009214883
 -RESYNC    00D359691359EE84 V R  9128009214884
SAN_FRANCISCO     DSN05010  LUND3                 1         0      6
 -SYSTASK   SESSID           A ST TIME
 -SYSCON-O  0000000000000000   C  9128009214885
 -SYSCON-I  00D359691359EE86 W R  9128009214886
 -RESYNC    00D359691359EE87 W R  9128009214887
 -RESYNC    00D359691359EE88 W R  9128009214888
 -RESYNC    00D359691359EE89 W R  9128009214889
DISPLAY LOCATION REPORT COMPLETE
```

***Example 3:*** Display information for a DB2 that is connected to the following DRDA partners:

- A non-MVS server named DRDALOC via TCP/IP.
- Several TCP/IP clients from the same TCP/IP host as the DRDALOC server.
- A DB2 for MVS server named DB2SERV via SNA.

```
DISPLAY LOCATION(*)

DSNL200I - DISPLAY LOCATION REPORT FOLLOWS -
 LOCATION      PRDID     LINKNAME        REQUESTERS  SERVERS   CONVS
 DRDALOC       SQL03030  124.63.51.17    3           0         3
 124.63.51.17  SQL03030  124.63.51.17    0           15        15
 DB2SERV       DSN05010  LULA            1           0         1
 DISPLAY LOCATION REPORT COMPLETE
```

***Example 4:*** The following example assumes DB2 is connected to the following DRDA partners:

- DB2A is connected to this DB2, using TCP/IP for DRDA connections and SNA for DB2 private protocol connections.
- DB2SERV is connected to this DB2 using only SNA.

```
DISPLAY LOCATION(*)

DSNL200I - DISPLAY LOCATION REPORT FOLLOWS -
 LOCATION      PRDID     LINKNAME        REQUESTERS  SERVERS   CONVS
 DB2A          DSN05010  LUDB2A          3           4         9
 DB2A          DSN05010  124.38.54.16    2           1         3
 DB2SERV       DSN04010  LULA            1           1         3
 DISPLAY LOCATION REPORT COMPLETE
```

## -DISPLAY LOG (DB2)

The DB2 command DISPLAY LOG displays log information and the status of the offload task.

**Abbreviation**: DIS LOG

## Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following authorities:

- DISPLAY privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands that are issued from an MVS console are not associated with any 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 LOGLOAD setting, including 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). For more information about the Print Log Map utility, see Section 3 of *DB2 Utility Guide and Reference*.

## Examples

***Example 1:*** Display log information and status of the offload task.

```
DISPLAY LOG
```

This command produces output similar to the following output:

```
DSNJ370I - DSNJC00A LOG DISPLAY
CURRENT COPY1 LOG = DSNC610.LOGCOPY1.DS03 IS 22% FULL
CURRENT COPY2 LOG = DSNC610.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, 1998
RESTART RBA 0000039A8000
DSN9002I - DSNJC001 'DIS LOG' NORMAL COMPLETION
```

This example shows 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, 1998, and began logging at RBA 0000039A8000.

## -DISPLAY PROCEDURE (DB2)

The DB2 command DISPLAY PROCEDURE displays statistics about stored procedures accessed by DB2 applications. This command displays one output line for each stored procedure that has been accessed by a DB2 application. 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 an MVS 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, the privilege set of the process must include one of the following:

- DISPLAY privilege
- SYSOPR, SYSCTRL, or SYSADM authority
- Ownership of the stored procedure

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 issued from an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
                       ┌─(*.*)─────────────────────────────────┐
►►──DISPLAY PROCEDURE──┼───────────────────────────────────────┼──────────────────────────►◄
                       │      ┌─,──────────────────────┐       │
                       └─(──▼─┬─schema.procedure-name─┬─┴─)───┘
                              ├─schema.partial-name*──┤
                              ├─procedure-name────────┤
                              └─partial-name*─────────┘
```

## Option descriptions

| **(*.*)**
| Displays information for all stored procedures in all schemas that have been accessed by DB2 applications 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 have been
| accessed by DB2 applications since DB2 was started. The names of all proce-
| dures 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 have been accessed by DB2 applications 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.

## Output

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

```
DSNX940I - DSNX9DIS DISPLAY PROCEDURE REPORT FOLLOWS -

PROCEDURE          STATUS ACTIVE  QUEUED MAXQUE TIMEOUT  WLM_ENV
APPL1              STARTED     1       0      0       0
APPL2              STARTED     1       0      0       0
APPL2              STARTED     0       1      2       0
APPL5              STOPREJ     0       0      0       0
APPL6              STOPABN     0       0      0       0
PROC1              STOPQUE     0       0      0       0

DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
```

***Description of output:*** Each output line displays:

| | |
|---|---|
| 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. |
| QUEUED | The number of threads that are waiting for the procedure to be scheduled. |
| MAXQUE | 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 execute 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 execute the START PROCEDURE command. |

Message DSNX943I lists a range of procedures that are stopped because a STOP PROCEDURE command included a partial name with a pattern-matching character (*) such as:

```
-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.

## Examples

***Example 1:*** Display information about all stored procedures that have been accessed by DB2 applications.

```
-DISPLAY PROCEDURE
```

This command produces output similar to the following output:

```
DSNX940I DSNX9DIS - DISPLAY PROCEDURE REPORT FOLLOWS-
PROCEDURE         STATUS    ACTIVE    QUEUED  MAXQUE  TIMEOUT
USERPRC1          STARTED       0         0       1        0
USERPRC2          STOPQUE       0         5       5        3
USERPRC3          STARTED       2         0       6        0
USERPRC4          STOPREJ       0         0       1        0
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

***Example 2:*** Display information about specific stored procedures in the SYSPROC schema.

```
-DISPLAY PROCEDURE(SYSPROC.USERPRC2,USERPRC4)
```

This command produces output similar to the following output:

```
DSNX940I DSNX9DIS - DISPLAY PROCEDURE REPORT FOLLOWS-

------ SCHEMA=SYSPROC
PROCEDURE         STATUS    ACTIVE    QUEUED  MAXQUE  TIMEOUT  WLM_ENV
USERPRC2          STOPQUE       0         5       5        3  SANDBOX
USERPRC4          STOPREJ       0         0       1        0  SANDBOX
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

***Example 3:*** Display information about stored procedures in the PAYROLL and HRPROD schemas.

```
-DISPLAY PROCEDURE(PAYROLL.*,HRPROD.*)
```

This command produces output similar to the following output:

```
DSNX940I DSNX9DIS - DISPLAY PROCEDURE REPORT FOLLOWS-

------ SCHEMA=PAYROLL
PROCEDURE          STATUS    ACTIVE    QUEUED  MAXQUE  TIMEOUT  WLM_ENV
PAYPRC1            STARTED        0         0       1        0  PAYROLL
PAYPRC2            STOPQUE        0         5       5        3  PAYROLL
PAYPRC3            STARTED        2         0       6        0  PAYROLL
USERPRC4           STOPREJ        0         0       1        0  SANDBOX

------ SCHEMA=HRPROD
PROCEDURE          STATUS    ACTIVE    QUEUED  MAXQUE  TIMEOUT  WLM_ENV
HRPRC1             STARTED        0         0       1        0  HRPROCS
HRPRC2             STOPREJ        0         0       1        0  HRPROCS
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

***Example 4:*** Display information about all stored procedures in the SYSPROC schema that have been accessed by DB2 applications. Assume the -STOP PROCEDURE(SYSPROC.*) ACTION(QUEUE) command is in effect at the time this command is issued.

```
-DISPLAY PROCEDURE(SYSPROC.*)
```

This command produces output similar to the following output:

```
DSNX940I DSNX9DIS - DISPLAY PROCEDURE REPORT FOLLOWS-

------ SCHEMA=SYSPROC
PROCEDURE          STATUS    ACTIVE    QUEUED  MAXQUE  TIMEOUT  WLM_ENV
USERPRC2           STOPQUE        0         5       5        3  SANDBOX
USERPRC4           STOPQUE        0         0       1        0  SANDBOX
PROCEDURES USERFNC2           - USERFNC29999999999 STOP QUEUE
PROCEDURES USERFNC4           - USERFNC49999999999 STOP QUEUE
DSNX9DIS DISPLAY PROCEDURE REPORT COMPLETE
DSN9022I - DSNX9COM '-DISPLAY PROC' NORMAL COMPLETION
```

## -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 being used.

**Abbreviation**: -DIS RLIM

## Environment

This command can be issued from an MVS 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, the privilege set of the process must include the following:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
►►──DISPLAY RLIMIT──────────────────────────────────────────────────────────►◄
```

## Example

Display the current status of the resource limit facility.

```
-DISPLAY RLIMIT
```

If the resource limit facility (RLF) is inactive, the following output is generated:

```
DSNT701I - RESOURCE LIMIT FACILITY IS INACTIVE
DSN9022I - DSNTCDIS 'DISPLAY RLIMIT' NORMAL COMPLETION
```

If the RLF is active, the value of field RESOURCE AUTHID on panel DSNTIPP is SYSADM, and the resource limit specification table with RLST NAME SUFFIX = 03 was started, the following output is generated:

```
DSNT700I = SYSADM.DSNRLST03 IS THE ACTIVE RESOURCE LIMIT
SPECIFICATION TABLE
DSN9022I = DSNTCDIS 'DISPLAY RLIMIT' NORMAL COMPLETION
```

# -DISPLAY THREAD (DB2)

The DB2 command DISPLAY THREAD displays current status information about DB2 threads. A DB2 thread is either 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 nondistributed; 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, or the set of active, indoubt, and postponed threads (see discussion under the TYPE option for more information)
- The allied threads associated with the address spaces whose connection names are specified
- Allied threads
- Distributed threads
- Distributed threads associated with a specific remote location
- Detailed information about connections with remote locations
- A specific logical unit of work ID (LUWID)

The information 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 an MVS 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, the privilege set of the process must include one of the following:

- DISPLAY privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
►►──DISPLAY THREAD──┬──────────────────────────────────────────┬──┬─────────────────────────┬──────────────►
                    │      ┌─,──────────────────────┐          │  │         ┌─ACTIVE────┐   │
                    ├─(────▼─┬─connection-name──────┬┴───)─────┤  └─TYPE(──┼─INDOUBT───┼─)─┘
                    │        └─partial-connection*──┘          │           ├─*─────────┤
                    └─(──*──)──────────────────────────────────┘           ├─INACTIVE──┤
                                                                            └─POSTPONED─┘

►──┬──────────────────────────────────────────────────────────┬──┬────────┬──────────────────────────────►◄
   │                   ┌─,─────────────────────┐              │  └─DETAIL─┘
   ├──LOCATION(──┬─────▼─┬─location-name───────┬┴──┬──)───────┤
   │             │       └─partial-location*───┘   │          │
   │             └─*───────────────────────────────┘          │
   │           ┌─,──────────────────┐                         │
   └──LUWID(───▼─┬─luwid────────────┬┴───)────────────────────┘
                 ├─partial-luwid*───┤
                 └─token────────────┘
```

## Option descriptions

Only under certain conditions, as described below, are any of the following options required.

If neither (*connection-name*) nor (*) is specified, then 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, then the connection name is inherited from the associated address space.
- If the command is not issued from one of the above environments, then the following rules apply:
  - If neither LOCATION nor LUWID is specified, then processing terminates with a DSNV413I message.
  - If LOCATION or LUWID is specified, then only distributed threads of the type selected by the TYPE option are displayed.
  - When *location-name* is explicitly specified, then 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 LOCATION(*) is specified, then only distributed threads of the type specified in the TYPE option are displayed.
- When *location-name* is explicitly specified, then only distributed threads of the specified type that either have (active or inactive threads) or had (indoubt threads) 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 described above 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 LOCATION(*) is specified, then only distributed threads are displayed.
- When *location-name* is explicitly specified, then 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 associated with the transaction manager from which the command was entered.

**TYPE**
Tells what 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 via VTAM 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 produced by ACTIVE can be useful for debugging purposes, especially messages DSNV403I and DSNV404I; the contents of those messages are described in Section 3 of *DB2 Messages and Codes*.

**(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 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 OS/390 RRS for threads that use RRSAF. The commit coordinator for a database access thread is a requester at a remote system.

Indoubt threads hold locks on all resources that were updated.

**Abbreviation**: I

| **(*)** Displays active, indoubt, and postponed threads.

**(INACTIVE)**
Displays only inactive threads. An inactive thread is a database access thread that is connected via VTAM to another system and is idle, waiting 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 could 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 on page 260 to complete backout processing for the postponed units of work.

**LOCATION(***location-name***, ...)**
Limits the display to distributed threads as described below.

**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 OS/390 subsystems. To display information about a requester that is not a DB2 for OS/390 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, the following command displays information about a remote location (that is not DB2 for OS/390) with the LUNAME of LULA:

```
-DISPLAY THREAD (*) LOCATION (<LULA>)
```

The following command displays information about a remote location (that is not DB2 for OS/390) with an IP address of 123.34.101.98:

```
-DISPLAY THREAD (*) LOCATION (123.34.101.98)
```

DB2 uses the <LUNAME> 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 a LUNAME in the same manner as described above for other location names that are not DB2 for OS/390 subsystems. For example, LOCATION(<LULA*) selects all remote locations (that are not DB2 for OS/390) with an LUNAME 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 1 to 8 character network ID, a period, and a 1 to 8 character network LU name. The LUW instance number consists of 12 hex 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 there are no 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.

**DETAIL**
Displays additional information about conversation or socket activity when distributed information is displayed for active or inactive threads. DETAIL has no effect on the display of indoubt threads.

## 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. See these messages in Section 3 of *DB2 Messages and Codes* for an explanation of the formatted report.

***Threads using private protocol and DRDA access:*** It is possible for a database access thread that is connected to an application requester using DRDA access to also be connected to a database access thread at another DB2 location using DB2 private protocol access. In this case, a DSNV445I message is issued for the application requester, and a DSNV444I message and 0 or more DSNV446I messages are issued for the remote connections that are using DB2 private protocol access to other DB2 subsystems.

***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 MVS 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, or a combination of these, as appropriate, to reduce the output.

***Showing parallel tasks:*** The DB2 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. This shows that these parallel tasks are related to the originating task. If the parallel task is running on a DB2 that is different from the DB2 that runs the originating task, then the connection name is shown and the entry is followed by message DSNV443I.

## Output

Table 16 on page 168 shows sample -DISPLAY THREAD commands and the types of output they generate. The DETAIL keyword is not included because it affects only the amount of information displayed about a distributed thread.

# *Table 16. Sample DISPLAY THREAD commands. The following output is generated when*
# *commands are issued from different environments with different TYPE specifications. (Speci-*
# *fying TYPE(*) displays the equivalent output of both TYPE(ACTIVE) and TYPE(INDOUBT) in*
# *one report.)*

| | ACTIVE | INDOUBT | INACTIVE |
|---|---|---|---|
| **Command issued from a DSN session under TSO, DB2I, IMS or CICS, where the connection name is inherited** | | | |
| -DIS THD | 1 | 1 | 2 |
| -DIS THD LOC(*) | 3 | 3 | 2 |
| -DIS THD LOC(*location-name*) | 4 | 4 | 2 |
| **Command issued from MVS console** | | | |
| -DIS THD | 6 | 6 | 6 |
| -DIS THD LOC(*) | 9 | 9 | 8 |
| -DIS THD LOC(*location-name*) | 10 | 10 | 11 |
| **Command issued from any source** | | | |
| -DIS THD(*connection-name*) | 1,12 | 1,12,15 | 12 |
| -DIS THD(*connection-name*) LOC(*) | 3,12 | 3,12,15 | 12 |
| -DIS THD(*connection-name*) LOC(*location-name*) | 4,13 | 4,13,15 | 13 |
| -DIS THD(*) | 7 | 7,15 | 8 |
| -DIS THD(*) LOC(*) | 9 | 9,15 | 8 |
| -DIS THD(*) LOC(*location-name*) | 10 | 10,15 | 11 |
| -DIS THD(*) LUWID(*luwid* or *token*) | 5 | 5,15 | 5 |
| -DIS THD(*connection-name*) LUWID(*luwid* or *token*) | 14 | 14,15 | 14 |
| -DIS THD LUWID(*luwid* or *token*) | 5 | 5,15 | 5 |

**Description of display generated:**

1. Allied threads of the specified TYPE with the connection name.
2. No threads (inactive threads are database access threads and have no inherited connection name).
3. Distributed allied threads of the specified TYPE with the connection name.
4. Distributed allied threads of the specified TYPE with the connection name and a distributed connection = *location-name*.
5. The threads of the specified TYPE that have LUWID = *luwid* or *token*.
6. Message DSNV413I is displayed to indicate an error.
7. All threads (both allied and database access) of the specified TYPE.
8. All inactive database access threads.
9. All distributed threads (both allied and database access) of the specified TYPE.
10. All distributed threads (both allied and database access threads) of the specified TYPE with a distributed connection = *location-name*.
11. All inactive database access threads with a distributed connection = *location-name*.
12. Database access threads of the specified TYPE with the connection name.
13. Database access threads of the specified TYPE with the connection name and a distributed connection = *location-name*.
14. A thread of the specified TYPE with the connection name and LUWID = *luwid* or *token*.
15. Messages DSNV407 and DSNV408 also display coordinator's TCP/IP resync port number; message DSNV446 also displays the participant's TCP/IP resync port number.

If the DETAIL option is specified, then the following additional information is displayed:

| | |
|---|---|
| LOCATION | The location name of the remote system. |
| SESSID | For a VTAM connection, the VTAM defined session instance identifier of the session on which the conversation is executing. For a TCP/IP connection, the local and remote TCP/IP port numbers, in the form *local:remote*. *local* is the port number for the local DB2 subsystem. *remote* is the port number for the remote partner. |
| A | If VTAM or TCP/IP has control of the conversation (if DB2 transferred control of the thread to VTAM or TCP/IP for that conversation), there is a V in the A (Active) column. A W indicates that DB2 has suspended processing on this conversation until VTAM notifies DB2 that the VTAM event is complete. The column is otherwise blank. |
| STATUS | This 2 byte column indicates the status of the conversation or socket. The possible values for STATUS are: |

| **Value** | **Status** |
|---|---|
| S*x* | Send |
| R*x* | Receive |
| A*x* | Allocation |
| D*x* | Deallocation |
| C*x* | Change number of sessions (CNOS) processing |
| X*x* | Exchange Log name processing |
| blank | Not in one of the above mentioned states |

*x* can be one of the following values:

1 = private protocol conversation with single-phase commit
2 = DRDA conversation with single-phase commit
3 = private protocol conversation with two-phase commit
4 = DRDA conversation with two-phase commit.

## Examples

***Example 1:*** 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 command CANCEL THREAD.

```
-DIS THD(*) 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  *     5 BKH2C           SYSADM    BKH2      000D   123
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

***Example 2:*** This example shows information about conversation activity when distribution information is displayed for active threads. DB2 returns the following message, indicating that the local site application is waiting for a conversation to be allocated in DB2, and a DB2 server that is accessed by a DRDA client using TCP/IP.

```
-DIS THD(*) LOCATION(*) DETAIL
```

This command produces output similar to the following output:

```
-DIS THD(*) LOC(*) DET
  DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
  DSNV402I - ACTIVE THREADS -
     NAME     ST A   REQ ID           AUTHID   PLAN     ASID   TOKEN
     TSO      TR *     3 SYSADM       SYSADM   DSNESPRR 002E       2
      V436-PGM=DSNESPRR.DSNESM68, SEC=1, STMNT=116
      V444-DB2NET.LUND0.A238216C2FAE=2 ACCESSING DATA AT
      V446-USIBMSTODB22:LUND1
      V447--LOCATION        SESSID        A ST    TIME
      V448--USIBMSTODB22    0000000000000000 V A1 9015816504776
     TSO      RA *    11 SYSADM       SYSADM   DSNESPRR 001A      15
      V445-STLDRIV.SSLU.A23555366A29=15 ACCESSING DATA FOR 123.34.101.98
      V447--LOCATION        SESSID           A ST    TIME
      V448--123.34.101.98   446:3171        S2    9015611253108
     DISPLAY ACTIVE REPORT COMPLETE
  DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

***Example 3:*** In this example, a system at STL has a TSO application and an IMS application. The system at STL fails after DB2 commits the TSO application, but before the commit decision has been communicated to the participant subsystems at SJ and LA. The failure occurs before IMS has communicated the commit or rollback decision to STL's DB2. The DISPLAY THREAD commands that are issued after the STL DB2 restarts but before reconnect with IMS. DISPLAY THREAD commands that are issued at each location show output similar to the following output:

At STL:

```
-DIS THD(*) TYPE(INDOUBT)
```

This command produces output similar to the following output:

```
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV406I - INDOUBT THREADS -
 COORDINATOR              STATUS     RESET URID         AUTHID
 STLIMS01                 INDOUBT          0F201050A010 SM09H
  V467-HAS LUWID IBM.STLDB21.15A86A876789.0010=1
  V449-HAS NID=A5 AND ID=STLIMS01
  V450-HAS PARTICIPANT INDOUBT AT
  V446--IBMSJ0DB20001:STLDB22
 IBMSTLDB20001            COMMITTED        0F20105B0000 JO78S
  V467-HAS LUWID IBM.STLDB21.16B57B954427.0003=2
  V450-HAS PARTICIPANT INDOUBT AT
  V446--IBMSJ0DB20001:STLDB22  IBMLA0DB20001:STLDB23
 DISPLAY INDOUBT REPORT COMPLETE -
 DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

At San Jose:

```
-DIS THD(*) TYPE(INDOUBT)
```

This command produces output similar to the following output:

```
 DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
 DSNV406I - INDOUBT THREADS -
 COORDINATOR              STATUS     RESET URID         AUTHID
 IBMSTLDB20001:STLDB21    INDOUBT          03201050A010 HEU4443
  V467-HAS LUWID IBM.STLDB21.15A86A876789.0010=8
  V466-THREAD HAS BEEN INDOUBT FOR 00:05:20
 IBMSTLDB20001:STDB21     INDOUBT          03201050B000 PP433MM
  V467-HAS LUWID IBM.STLDB21.16B57B954427.0003=6
 DISPLAY INDOUBT REPORT COMPLETE
 DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

At Los Angeles (both ACTIVE and INDOUBT threads are displayed):

```
-DIS THD(*) TYPE(*) 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
 SERVER   RA *     0 RUW2STAT     JONES    DISTSERV 0005     4
  V465-THREAD HAS BEEN PREPARED FOR 00:05:20
  V445-IBM.STLDB21.15A86A876789=4 ACCESSING DATA FOR
       IBMSJ0DB20001:STLDB21
  V447--LOCATION        SESSID           A ST TIME
  V448--IBMSJ0DB20001    0000000400000004 W R4 9034817015032
 DISPLAY ACTIVE REPORT COMPLETE
 DSNV406I - INDOUBT THREADS -
 COORDINATOR              STATUS     RESET URID         AUTHID
 IBMSTLDB20001:STLDB21    INDOUBT          03201050B000 SM43YY33
  V467-HAS LUWID IBM.STLDB21.16B57B954427.0003=5
  V466-THREAD HAS BEEN INDOUBT FOR 00:05:20
 DISPLAY INDOUBT REPORT COMPLETE
 DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

***Example 4:*** 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:

```
-DIS THD(*)
```

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:

```
-DIS THD(*)
```

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 5:*** This example shows an allied, nondistributed 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. 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  DSNTEP3  0025    30
         PT *   641 PUPPYDML     USER001  DSNTEP3  002A    40
         PT *    72 PUPPYDML     USER001  DSNTEP3  002A    39
         PT *   549 PUPPYDML     USER001  DSNTEP3  002A    38
         PT *   892 PUPPYDML     USER001  DSNTEP3  002A    37
         PT *    47 PUPPYDML     USER001  DSNTEP3  002A    36
         PT *   612 PUPPYDML     USER001  DSNTEP3  002A    35
         PT *   545 PUPPYDML     USER001  DSNTEP3  002A    34
         PT *   432 PUPPYDML     USER001  DSNTEP3  002A    33
         PT *   443 PUPPYDML     USER001  DSNTEP3  002A    32
         PT *   252 PUPPYDML     USER001  DSNTEP3  002A    31
DISPLAY ACTIVE REPORT COMPLETE
16.32.58 STC00090  DSN9022I DB1G DSNVDT '-DISPLAY THREAD' NORMAL
 COMPLETION
```

***Example 6:*** This example shows the detail report for a DB2 client that uses TCP/IP to access a remote DRDA server.

```
DISPLAY THREAD(*) LOCATION(*)
```

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 *     6 BKH2C        SYSADM YW1019C  0009      2
    V444-STLDRIV.SSLU.A23555366A29=2 ACCESSING DATA AT
    V446-USIBMSTODB22:123.34.101.98:446
    V447--LOCATION         SESSID           A ST TIME
    V448--USIBMSTODB22     4019:446         V R2 9015611253116
   DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

***Example 7:*** This example shows the detail report for a DB2 server that is accessed by a DRDA client using TCP/IP.

```
DISPLAY THREAD(*) LOCATION(*)
```

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    RA *     5 BKH2C        SYSADM DISTSERV 0008      2
    V445-STLDRIV.SSLU.A23555366A29=2 ACCESSING DATA FOR 123.34.101.98
    V447--LOCATION         SESSID           A ST TIME
    V448--123.34.101.98    446:3171           S2 9015611253108
   DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

| ***Example 8:*** 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 9:*** This example shows the token for a thread that is executing 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  DSNTEP3  0025    18
|  V436-PGM=CLIP74C1.UFIP74C1, SEC=0, STMNT=0
|  V429 CALLING FUNCTION =SCIP7401.SP_UFIP74C1       ,
|      PROC=V61AWLM3, ASID=0030, WLM_ENV=WLMENV3
| DISPLAY ACTIVE REPORT COMPLETE
| DSN9022I - DSNVDT '-DISPLAY THREAD' NORMAL COMPLETION
```

## -DISPLAY TRACE (DB2)

The DB2 command DISPLAY TRACE displays a list of active traces. For more information about this trace facility, see Section 4 (Volume 1) of *DB2 Administration Guide*.

There is an additional option to this command and values for a few options that are not described here. They are intended for service and use under the direction of IBM support personnel. For details, see *DB2 Diagnosis Guide and Reference*.

**Abbreviation**: -DIS TRACE

### Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following:

- DISPLAY privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
                     ┌─(──*──)────────┐
►►──DISPLAY TRACE────┼────────────────┼──┬───────────────────┬──┬──────────────────┬──────────────►
                     ├─(──PERFM──)────┤  └─destination block─┘  └─constraint block─┘
                     ├─(──ACCTG──)────┤
                     ├─(──STAT──)─────┤
                     ├─(──AUDIT──)────┤
                     └─(──MONITOR──)──┘

►──┬──────────────────────┬──┬──────────────────┬────────────────────────────────────────────────►◄
   └─DETAIL(output-type)──┘  └─COMMENT(string)──┘

destination block:

         ┌─,────────┐
►──DEST(─▼─┬─GTF─┬──┴─)──────────────────────────────────────────────────────────────────────────►
           ├─SMF─┤
           ├─SRV─┤
           └─OPn─┘

constraint block:

        ┌─*───────────────┐           ┌─*─────────────┐          ┌─*──────────────┐
        │  ┌─,──────────┐ │           │  ┌─,────────┐ │          │  ┌─,─────────┐ │
►──PLAN(┴──▼─plan-name──┴─┴─)──AUTHID(┴──▼─auth-id──┴─┴─)──CLASS(┴──▼─integer──┴─┴─)──────────────►

       ┌─*────────────┐               ┌─*────────────────────┐
       │  ┌─,───────┐ │               │  ┌─,───────────────┐ │
►──TNO(┴──▼─integer─┴─┴─)──LOCATION(──┼──▼─location-name───┴─┼─)──────────────────────────────────►
                                      ├─<luname>─────────────┤
                                      └─nnn.nnn.nnn.nnn──────┘
```

## 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 command

```
-DISPLAY TRACE (PERFM) CLASS (1,2)
```

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).

# **( *)**

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. For further descriptions of each type, see "-START TRACE (DB2)" on page 300.

| **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, PLAN, LOCATION |
| 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.

**DEST**

Limits the list to traces started for particular destinations. More than one value can be specified, but do not use the same value twice. If you do not specify a value for DEST, DB2 does not use the destination of where trace output is recorded to limit the list of traces displayed.

**Abbreviation**: D

Possible values and their meanings are:

| **Value** | **Trace destination** |
|---|---|
| GTF | The generalized trace facility |
| SMF | The system management facility |
| SRV | An exit to a user-written routine |
| OP*n* | A specific destination. *n* can be a value from 1 to 8. |

See “-START TRACE (DB2)” on page 300 for a list of allowable destinations for each trace type.

**PLAN(*plan-name*, ...)**

Limits the list to traces started for particular application plans. Up to eight plan names can be used. If more than one name is used, only one value can be used for AUTHID, TNO, and LOCATION. Do not use this option with STAT.

The **default** is **PLAN**(_*_), which does not limit the list.

**AUTHID(***authorization-id***, ...)**
Limits the list to traces started for particular authorization identifiers. Up to eight identifiers can be used. If more than one identifier is used, only one value can be used for PLAN, TNO, and LOCATION. Do not use this option with STAT.

The **default** is **AUTHID(*)**, which does not limit the list.

**CLASS(***integer***, ...)**
Limits the list to traces started for particular classes. For descriptions of the allowable classes, see "-START TRACE (DB2)" on page 300.

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.

**LOCATION(***location-name***, ...)**
Limits the list to traces started for threads that have a distributed relationship with the specified location.

**(***location-name***)**
The location names that you supply are the 1 to 16 character identifiers assigned to the DB2 subsystem whose traces you want to display. Supplying an * as the location name indicates that the trace display must include all traces started with any location name qualifier.

You can specify up to eight location names. If you specify more than one location name, you can only specify one value each for PLAN, AUTHID, and TNO.

LOCATION cannot be specified when you choose a statistics trace.

**Requesters other than DB2 for OS/390**: DB2 does not receive a location name from requesters that are not DB2. To display information about a requester that is not a DB2 for OS/390 subsystem, enter its LUNAME, enclosed by the less-than (<) and greater-than (>) symbols. For example, the following command displays information about a remote location with the LUNAME of LULA:

```
-DISPLAY TRACE (*) LOCATION (<LULA>)
```

DB2 uses the < LUNAME> notation in messages displaying information about requesters that are not DB2 for OS/390.

The **default** is **LOCATION(*)**, which does not limit the list.

<*luname*>
Activates the DB2 trace for the remote clients that are connected to DDF through the remote SNA LU that you specified in *luname*.

*nnn.nnn.nnn.nnn*
Activates the DB2 trace for the remote clients that are connected to DDF through the remote TCP/IP host whose IP address is specified by *nnn.nnn.nnn.nnn*.

## Examples

***Example 1:*** List all traces that have the generalized trace facility as their only destination.

```
-DISPLAY TRACE (*) DEST (GTF)
```

***Example 2:*** List the trace started for Example 2 of -START TRACE.

```
-DISPLAY TRACE (ACCTG) PLAN (DSN8BC61)
  COMMENT ('ACCTG TRACE FOR DSN8BC61')
```

***Example 3:*** List all active performance traces.

```
-DISPLAY TRACE=P
```

***Example 4:*** List all active audit traces for threads that are connected to the DB2 subsystem with location name USIBMSTODB23.

```
-DISPLAY TRACE (AUDIT) LOCATION (USIBMSTODB23)
```

***Example 5:*** Output from -DISPLAY TRACE is a set of messages that look like this:

```
- 10.26.34           -DISPLAY TRACE
- 10.26.34 STC   21  DSNW127I - CURRENT TRACE ACTIVITY IS -
- TNO TYPE   CLASS        DEST QUAL
- 01  STAT   01           SMF  NO
- 02  ACCTG  01           SMF  YES
- 03  PERFM  01,02,03     GTF  YES
- 04  AUDIT  01,02,03,04, SMF  YES
- 04         06,07
- 05  MON    01,02,03     OP1  NO
- *********END OF DISPLAY TRACE SUMMARY DATA*********
- 10.26.34 STC   21  DSN9022I - DSNWVCM1 '-DISPLAY TRACE' NORMAL COMPLETION

- 10.28.47           -DISPLAY TRACE DETAIL(*)
- 10.28.47 STC   21  DSNW127I - CURRENT TRACE ACTIVITY IS -
- TNO TYPE   CLASS        DEST QUAL
- 01  STAT   01           SMF  NO
- 02  ACCTG  01           SMF  YES
- 03  PERFM  01,02,03     GTF  YES
- 04  AUDIT  01,02,03,04, SMF  YES
- 04         06,07
- 05  MON    01,02,03     OP1  NO
- *********END OF DISPLAY TRACE SUMMARY DATA*********
- 10.28.47 STC   21  DSNW143I - CURRENT TRACE QUALIFICATIONS ARE -
- TNO AUTHID   PLAN     RMID         LOCATION
- 01  *        *        *
- 02  *        *        *
- 03  USER01   *        *
- 04  *        *        14,16,18,26  DENVER
- 05  *        PROG1    *
- 06  *        *        *
- ******END OF DISPLAY TRACE QUALIFICATION DATA******
- 10.28.47 STC   21  DSN9022I - DSNWVCM1 '-DISPLAY TRACE' NORMAL COMPLETION
```

## -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 an MVS 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

```
►►──DISPLAY UTILITY──(─┬─utility-id──────────┬─)─┬─────────────────────────────┬──────────────►◄
                       ├─partial-utility-id*─┤   │          ┌─,───────────┐    │
                       └─*───────────────────┘   └─MEMBER(──▼─member-name─┴─)─┘
```

## Option descriptions

Use at least one of the following options but do not use the same one 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*. For a list of values for *control-file-name*, see the description of the UID parameter for the DSNU command procedure (CLIST) in *DB2 Utility Guide and Reference*.

  If *utility-id* was omitted, *utility-id* has the form *userid.jobname*.

**(***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 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 -DISPLAY UTILITY 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 as follows as shown in Example 4 on page 182

| | |
|---|---|
| DEADLINE | Indicates a timestamp according to the most recently specified value of DEADLINE. |
| MAXRO | Indicates the number of seconds, according to the most recently specified value of MAXRO. |
| LONGLOG | Indicates either CONTINUE, TERM, or DRAIN according to the most recently specified value of LONGLOG. |
| DELAY | Indicates the number of seconds according to the most recently specified value of DELAY. |

***Output during LOG phase of REORG with SHRLEVEL CHANGE:*** During the LOG phase of REORG with SHRLEVEL CHANGE, the output of -DISPLAY UTILITY now includes the additional information found in message DSNU383I as shown in Example 4 on page 182 as follows:

| | |
|---|---|
| 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 values are 0 if the current iteration number is 1 or 2. |
| PREVIOUS ITERATION | Indicates the ELAPSED TIME and the NUMBER OF LOG RECORDS PROCESSED for the previous iteration. Their values are 0 if the current iteration number is 1. |
| 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. |

***Progress of utility processing:*** The DISPLAY UTILITY command provides the user an estimate of how much processing the utility has completed. The output displays information from message DSNU105I as seen in Example 2 on page 181 and includes:

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 below, ignore this field.

- For CHECK INDEX, LOAD, RECOVER INDEX and REORG, COUNT represents the number of records processed.
- For COPY, MERGE COPY, RECOVER (restore phase), and RUNSTATS, COUNT represents the number of pages processed.
- For STOSPACE, COUNT represents the number of table spaces or indexes processed.

For more information, see *DB2 Utility Guide and Reference*.

## Examples

***Example 1:*** Display status information for all utility jobs currently known to DB2.

```
-DISPLAY UTILITY (*)
```

***Example 2:*** Display the status of utilities on all members of the data sharing group.

```
-DB1G DISPLAY UTILITY (*)
```

The following output, which shows utility jobs on members DB1G and DB2G, is generated:

```
 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 3:*** In a data sharing environment, display the status of utilities on member DB1G.

```
-DB1G DISPLAY UTILITY (*) MEMBER (DB1G)
```

***Example 4:*** This shows output from the command DISPLAY UTILITY:

```
-DB1G DISPLAY UTILITY(*)
```

```
DSNU105I -DB1G DSNUGDIS - USERID = SYSADM 973
                MEMBER =
                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 DSNUGCCC '-DIS UTIL' NORMAL COMPLETION
```

## DSN (TSO)

The TSO command DSN enables you to issue DSN subcommands, namely:

| | | | |
|---|---|---|---|
| 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, execute it with the DSN command; for example:

```
TEST 'prefix.SDSNLOAD(DSN)' CP
```

The ABEND subcommand listed above is used for diagnostic purposes only, and is intended to be used only under the direction of your IBM Support Center. 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

```
►►─DSN──┬──────────────────────────────────────────┬──┬───────────────────────────┬──┬──────────────────┬─────────►◄
        └─SYSTEM(─┬─DSN────────────────────┬─)─┘  └─RETRY(─┬─0───────┬─)─┘  └─TEST(integer)─┘
                  ├─subsystem-name─────────┤                └─integer─┘
                  └─group-attachment-name──┘
```

## 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.

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 sub-
# system 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.

## Usage notes

***Beginning a DSN session:*** Issue the DSN command to begin a DSN session, which allows you to enter DSN subcommands. These 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 placed there by any DSN subcommand used in the session or by any program run by the RUN subcommand. Your run-time environment might format that value as a return code. The value does not, however, originate in DSN.

## Examples

***Example 1:*** Start a DSN session. If the attempt to connect to DB2 fails, five retries (at 30 second intervals) are to be made.

```
DSN SYSTEM (DB2) RETRY (5)
```

***Example 2:*** Start a DSN session, run a program, and then end the session and return to TSO.

```
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
```

## 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, as described in the appropriate *CICS for MVS/ESA CICS-RACF Security Guide* or *CICS/MVS Operations Guide.*

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

```
►►──DSNC──┬─────────────┬──db2-command──────────────────────────────────────────────►◄
          └─destination─┘
```

## 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 SKR*xxxx* can be used to support the scrolling of DSNC DB2 commands from your terminal. For further information regarding the SIT keywords and parameters, see *CICS for MVS/ESA System Definition Guide.*

## Example

***Example:*** Issue the DB2 command -DISPLAY THREAD from a CICS terminal.

```
DSNC -DISPLAY THREAD
```

## 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, as described in the appropriate *CICS for MVS/ESA CICS-RACF Security Guide.*

## 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 command CANCEL THREAD. For details, see "-CANCEL THREAD (DB2)" on page 88.

You can stop the transactions associated with a particular plan ID in CICS with the MAXIMUM setting for TCLASS or MAXACTIVE setting for TRANCLASS. This prevents new instances of the transaction from causing a re-creation of a thread.

***Alternative for protected threads:*** You may 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.

## Example

Disconnect active threads for PLAN1.

```
DSNC DISC PLAN1
```

## 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 entries in the resource control table (RCT).

**Abbreviation**: DSNC 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, as described in the *CICS for MVS/ESA CICS-RACF Security Guide* or *CICS/MVS Operations Guide.*

### Syntax

```
►►──DSNC DISPLAY──┬─PLAN──┬───────────┬───────┬──┬─────────────┬──────────────────────────►◄
                  │       └─plan-name─┘       │  └─destination─┘
                  ├─TRANSACTION──┬────────────────┬┤
                  │              └─transaction-id─┘│
                  └─STATISTICS─────────────────────┘
```

### 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 plan names listed in the resource control table.

**TRANSACTION** *transaction-id*
: Displays information about transactions by transaction ID.

  **Abbreviation**: TRAN

  *transaction-id* is a valid transaction ID for which information is displayed. For a group transaction entry in the resource control table, you can enter an identifier for any transaction associated with the group.

  **Default**: If you do not specify a transaction ID, information is displayed for all active transactions listed in the resource control table.

**STATISTICS**
: Displays the statistical counters associated with each entry in the resource control table. The counters concern 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, a statistics display is produced with no obvious indication that the subsystem is not operational. Message DSNC037A appears in the CICS message log to indicate that the attachment facility is waiting for DB2 to start.

For a description of the output produced by this parameter, see Section 4 (Volume 1) of *DB2 Administration Guide*.

*destination*
Is the identifier of another terminal 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 desired information at the desired terminal.

***Acknowledging display information sent to an alternate destination:*** When an alternate destination is specified to receive the requested display information, the following message is sent to the requesting terminal:

```
DSNC020I  THE DISPLAY COMMAND IS COMPLETE
```

## Output

The following is an example of the output for the DSNC DISPLAY (PLAN or TRANSACTION) command. For each created thread, the output shows the plan or transaction name. An 'A' in field A/I indicates that the thread is within a unit of work. An 'I' indicates that the thread is waiting for a unit of work authorization ID for the plan being used.

```
DSNC013I DISPLAY REPORT FOLLOWS
  NAME    A/I   AUTH-ID
XP05       A    SYSADM
DSNC020I THE DISPLAY COMMAND IS COMPLETE
```

For each entry in the RCT, the output of a DSNC DISPLAY STATISTICS command as seen in Example 4 on page 192 displays the following information:

| | |
|---|---|
| TRAN | Transaction name. For group entries, this is the name of the first transaction defined in the group. DSNC shows the statistics for the TYPE=COMD RCT entry. POOL shows statistics for the TYPE=POOL entry, unless the TYPE=POOL entry contains the parameter TXID=x. |
| PLAN | The plan name associated with this entry. Eight asterisks in this field indicates that this transaction is using dynamic plan allocation. The command processor transaction DSNC does not have a plan associated with it because it uses a command processor. |

| | |
|---|---|
| CALLS | The total number of SQL statements issued by transactions 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 TWAIT for the entry.<br><br>If TWAIT was set to POOL in the RCT, W/P indicates the number of times the transaction overflowed to the pool. An overflow to the pool shows up in the transaction statistics only and is not reflected in the pool statistics.<br><br>If TWAIT was set to YES, this reflects the number of times that the thread both had to wait, and could not attach a new subtask (number of started tasks has reached THRDA).<br><br>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 there are enough threads defined for the entry. |
| HIGH | The maximum number of threads required by transactions associated with this entry at any time since the connection was started. This number includes the transactions that were forced to wait or diverted to the pool. It provides a basis for setting the maximum number of threads for the entry. |
| ABORTS | The total number of units of recovery which were rolled back. It includes both abends and SYNCPOINT ROLLBACKS, including SYNCPOINT ROLLBACKS generated by -911 SQL codes. |
| COMMITS | One of the following two fields is incremented each time a DB2 transaction 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 associated with this entry. This total does not include any two-phase commits (see the explanation for 2-PHASE below). This total does include read-only commits as well as single phase commits for units of recovery which 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 associated with this entry. This number does not include one-phase commit transactions. |

## Examples

***Example 1:*** Display information on all active plan IDs listed in the resource control table. The display information is to be sent to another terminal designated as MTO2.

```
DSNC DISP PLAN * MTO2
```

***Example 2:*** Display information about all active transactions listed in the resource control table.

```
DSNC DISP TRANSACTION
```

***Example 3:*** Display statistical counters associated with each entry in the resource control table.

```
DSNC DISP STAT
```

***Example 4:*** This is an example of the output for the DSNC DISPLAY STATISTICS command:

```
DSNC014I   STATISTICS REPORT FOR 'DSNCRCTC' FOLLOWS
                                                             -----COMMITS-----
TRAN  PLAN         CALLS   AUTHS     W/P HIGH   ABORTS   1-PHASE   2-PHASE
DSNC                   1       1       1    1        0         0         0
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  DSNTP03          1       1       0    1        0         1         0
XA20  DSNTA20          1       1       0    1        0         0         1
XA88  ********         0       0       0    0        0         0         0
DSNC020I   THE DISPLAY COMMAND IS COMPLETE
```

## DSNC MODIFY (CICS attachment facility)

The CICS attachment facility command DSNC MODIFY modifies the ERRDEST entry in the resource control table (RCT), or the maximum active thread value associated with a given transaction or group name.

**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, as described in the appropriate *CICS for MVS/ESA CICS-RACF Security Guide* or *CICS/MVS Operations Guide.*

## Syntax

```
►►──DSNC MODIFY──┬─DESTINATION──old──new──────────────────┬──────────────────────────────────────────►◄
                 └─TRANSACTION──transaction-id──integer──┘
```

## Option descriptions

**DESTINATION**
Specifies that the ERRDEST parameter of the resource control table is to be changed, replacing the “old” destination ID with the “new” destination ID.

**Abbreviation**: DEST

*old*
Is any destination ID currently active in the ERRDEST of the resource control table.

*new*
Is a new destination identifier. The new destination is verified to ensure that it is an existing transient data entry in the destination control table.

**TRANSACTION**
Specifies that the maximum active thread value associated with the given transaction or group is to be modified.

**Abbreviation**: TRAN

*transaction-id*
Is a valid transaction identifier. If the change is for a group transaction entry in the RCT, any transaction ID associated with the group can be entered to identify the entry.

*integer*
Is a new maximum value.

## Usage notes

***Protected threads:*** If you increase the active thread value using the command DSNC MODIFY TRANSACTION, the attributes of the TYPE=ENTRY macro are used. If the TYPE=ENTRY definition does not include a setting for an attribute, then the TYPE=INIT value is used.

Issuing DSNC MODIFY TRANSACTION to increase the total threads permitted allows creation of unprotected threads. For example, assume THRDS=2, THRDA=2 and THRDM=6. If the total number of threads permitted is increased, the additional threads are unprotected.

The command DSNC MODIFY TRANSACTION can also allow creation of protected threads. If THRDS=2, THRDA=2 and THRDM=6 and the value of THRDA is modified to 1, one of the protected threads is eliminated. If the value of THRDA is then modified back to 2, the thread that is re-created is protected.

***TRANSACTION thread limit:*** The *integer* specified in the command DSNC MODIFY TRANSACTION cannot be larger than the value specified for the THRDM parameter of the DSNCRCT TYPE=ENTRY macro. The lowest possible value is zero. The value specified for the THRDM parameter is an upper limit (provided during initialization) on the number of threads to be connected for a transaction group. For more information about defining the thread limit, see Section 2 of *DB2 Installation Guide*.

## Example

Change the specification of the ERRDEST parameter in the resource control table from MTO1 to MTO2.

```
DSNC MODI DEST MTO1 MTO2
```

## 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, as described in the appropriate *CICS for MVS/ESA CICS-RACF Security Guide* or *CICS/MVS Operations Guide.*

## Syntax

```
             ┌─QUIESCE─┐
►►──DSNC STOP──┴─FORCE───┴──────────────────────────────────────────────────────────►◄
```

## Option descriptions

**QUIESCE**
Specifies that the CICS attachment facility is to be stopped after CICS transactions 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 both the requesting terminal, and to the transient data queue for error messages defined in the DSNCRCT TYPE=INITIAL macro.

## Example

Stop the CICS attachment facility.

```
DSNC STOP FORCE
```

## 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, as described in the appropriate *CICS for MVS/ESA CICS-RACF Security Guide* or *CICS/MVS Operations Guide.*

### Syntax

```
►►──DSNC STRT──┬────────────┬──┬───────┬──────────────────────────────────────────►◄
               └─rct-suffix─┘  └─,ssid─┘
```

### Option descriptions

*rct-suffix*
: Specifies the resource control table to be loaded when the CICS attachment facility starts. The *rct-suffix*, which can be one or two bytes, is appended to DSN2CT to create the name of the resource control table. (For example, if you specify 33 as the *rct-suffix*, module DSN2CT33 is loaded.)

  The **default** is the suffix specified in the CICS INITPARM parameter, a feature of the new CICS attachment facility. If a suffix was not specified in INITPARM, the default is 00, and module DSN2CT00 is loaded.

*ssid*
: Specifies the subsystem ID (SSID) field of the resource control table, which allows the attachment facility to connect to any DB2 subsystem using just one resource control table. No blanks are allowed between the suffix and the SSID.

  The **default** is the SSID specified in the CICS INITPARM parameter. If an SSID was not specified in INITPARM, the default is the SSID specified in the resource control table.

### Usage note

***Output destinations:*** Output from the DSNC START command is sent to both the requesting terminal, and to the transient data queue for error messages defined in the DSNCRCT TYPE=INIT macro.

## Examples

***Example 1:*** Start the CICS Version 4 Release 1 attachment facility. Use DSN2CT33 to connect to SSID DB2P.

```
DSNC STRT 33,DB2P
```

***Example 2:*** Start the CICS Version 4 Release 1 attachment facility. Use the default resource control table with SSID DBA1.

```
DSNC STRT ,DBA1
```

***Example 3:*** Start the CICS Version 4 Release 1 attachment facility. Use DSN2CTA to connect to the SSID that is specified in DSN2CTA.

```
DSNC STRT A
```

## DSNH (TSO CLIST)

The DSNH command procedure (a TSO CLIST) is a powerful yet easy method of preparing an application program for execution. 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 following actions referred to by the two-letter step name:

| For invoking the... | Use step name |
|---|---|
| PL/I macro processor | MP |
| DB2 precompiler | PC |
| CICS command language translator | TR |
| DSN BIND PLAN subcommand for binding a plan | BI |
| DSN BIND PACKAGE subcommand for binding a package | BP |
| Compiler or assembler for your program | CO |
| A C compiler prelink utility for including compile-time parameters | PL |
| Link-editor to produce an executable load module | LE |
| DSN RUN subcommand to execute the program | RU |

**Note:** The step names are used in the heading of Table 18 on page 200.

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 been previously completed successfully by DSNH. For guidance in preparing an application program for execution, see *DB2 Application Programming and SQL Guide*. See Table 1 on page 21 for a description of the DSN BIND subcommands.

## 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

See “BIND PACKAGE (DSN)” on page 47 for a description of the privileges necessary to bind a package.

See “BIND PLAN (DSN)” on page 52 for a description of the privileges necessary to bind a plan.

See “RUN (DSN)” on page 267 for a description of the privileges necessary to run a plan.

## Syntax

```
►►──DSNH──INPUT(data-set-name)──┬──────────────────────┬──────────────────────►◄
                                │ ┌──────────────────┐ │
                                └─▼─clist-parameter──┴─┘
```

## 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 200. The figure 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.

In Table 18 on page 200, a prefix is separated from the DB2 parameter name by a slash (/). Table 17 shows the variations possible for a single parameter name.

*Table 17. DSNH CLIST prefixing rules* #

| Parameter value | Function or subcommand | Example |
|---|---|---|
| *parameter* | If no prefix is specified, the parameter applies to a single function or subcommand. | `DBRMLIB` |
| B/*parameter* | The prefix B is used to indicate that this variation of the parameter applies only to the BIND PLAN step. | `BDBRMLIB` |
| P/*parameter* | The prefix P is used to indicate that this variation of the parameter applies only to the BIND PACKAGE step. | `PDBRMLIB` |

*Table 18 (Page 1 of 2). 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 | - | - |
| 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 | - |
| 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 | - | - |
| CPPPMSGS | - | - | - | - | - | - | 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 | - | - | - |
| GRAPHIC | - | Y | - | - | - | - | - | - | - |
| HOST | Y | Y | Y | * | - | Y | - | Y | Y |
| P/IMSBMP | - | - | - | Y | Y | - | - | - | - |
| P/IMSMPP | - | - | - | Y | Y | - | - | - | - |
| IMSPRE | - | - | - | - | - | - | - | Y | - |
| 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 | - | - | - | - | - | - |
| 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 | - | - | - |
| 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 | - | - | - | - |
| REMOTE | - | - | - | - | Y | - | - | - | - |
| P/REOPT | - | - | - | Y | Y | - | - | - | - |
| REPLVER | - | - | - | - | Y | - | - | - | - |
| RETAIN | - | - | - | Y | - | - | - | - | - |
| RUN | - | Y | Y | - | - | - | - | Y | Y |
| RUNIN | - | - | - | - | - | - | - | - | Y |
| RUNOUT | - | - | - | - | - | - | - | - | Y |
| SOMDLLI | - | - | - | - | - | Y | Y | - | - |
| SOURCE | Y | Y | Y | - | - | Y | - | - | - |
| SPACEUN | Y | Y | Y | - | - | Y | - | - | Y |
| SQL | - | Y | - | - | - | - | - | - | - |
| SQLDELIM | - | Y | - | - | - | - | - | - | - |

*Table 18 (Page 2 of 2). Summary of DSNH CLIST parameters*

| OPTIONS | MP | PC | TR | BI | BP | CO | PL | LE | RU |
|---|---|---|---|---|---|---|---|---|---|
| SQLERROR | - | - | - | - | Y | - | - | - | - |
| SQLFLAG | - | 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. For a summary of:

- BIND PLAN parameters, see Table 20 on page 213
- BIND PACKAGE parameters, see Table 21 on page 216.

Also see "DSNH/DSN subcommand summary" on page 213 for a description of conventions used in those tables.

The only parameter required on the DSNH statement is INPUT; the others are optional. In the table of general parameters that follow:

- Parameter values must be enclosed between parentheses.
- Parameter values need not be enclosed between apostrophes. However,
  - If the value is a list of tokens with separators, it must be enclosed between apostrophes.
  - If the value is a data set name, your user identifier is prefixed to it. To avoid the prefix, enclose the 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 (Page 1 of 13). General DSNH CLIST parameters*

| DSNH CLIST | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| ASMLIB | *dsname* | Specifies a data set to be used as the standard MACLIB for High Level Assembler/MVS.<br><br>The **default** is '''SYS1.MACLIB'''. |

*Table 19 (Page 2 of 13). General DSNH CLIST parameters*

| DSNH CLIST | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| ASMLOAD | *dsname* | Specifies a data set that contains the High Level Assembler/MVS load module.<br><br>*dsname* can include a member name.<br><br>The **default** is '‘SYS1.LINKLIB(ASMA90)’'. |
| CCLINK | *dsname* | Specifies a data set that contains the C compiler prelink utility invocation load module.<br><br>*dsname* can include a member name.<br><br>The **default** for **HOST(C)** is '‘CEE.V1R3M0.SCEERUN(EDCPRLK)’'.<br><br>The **default** for **HOST(CPP)** is '‘CEE.V1R4M0.SCEERUN(EDCPRLK)’'. |
| CCLLIB | *dsname* | Specifies the data set that contains the linkage editor include modules for the C compiler routines.<br><br>The **default** for **HOST(C)** is '‘CEE.V1R3M0.SCEELKED’'.<br><br>The **default** for **HOST(CPP)** is '‘CEE.V1R4M0.SCEELKED’'. |
| CCLOAD | *dsname* | Specifies a data set that contains the C compiler invocation load module.<br><br>*dsname* can include a member name.<br><br>The **default** for **HOST(C)** is '‘EDC.V1R2M0.SEDCDCMP(EDCDC120)’'.<br><br>The **default** for **HOST(CPP)** is '‘CBC.V3R1M0.SCBC3CMP(CBC310)’'. |
| CCMSGS | *dsname* | Specifies a data set that contains the C compiler messages.<br><br>*dsname* can include a member name.<br><br>The **default** for **HOST(C)** is '‘EDC.V1R2M0.SEDCDMSG(EDCMSGE)’'.<br><br>The **default** for **HOST(CPP)** is '‘CBC.V3R1M0.SCBC3MSG(EDCMSGE)’'. |
| CCOLIB | <u>NONE</u><br>*dsname* | Specifies the data set that contains C object modules to be included during the execution of the prelink utility step. |
| CCPLIB | <u>NONE</u><br>*dsname* | Specifies the data set containing include modules for PLI routines. This parameter is used only for IBM C/370 Version 2 or earlier. |
| CCPMSGS | *dsname* | Specifies the data set containing prelink utility error messages.<br><br>The **default** for **HOST(C)** is '‘CEE.V1R3M0.SCEEMSGP(EDCPMSGE)’'.<br><br>The **default** for **HOST(CPP)** is '‘CEE.V1R4M0.SCEEMSGP(EDCPMSGE)’'. |
| CCSLIB | *dsname* | Specifies the data set that contains the C compiler headers.<br><br>The **default** for **HOST(C)** is '‘EDC.V1R2M0.SEDCDHDR’'.<br><br>The **default** for **HOST(CPP)** is '‘CBC.V1R4M0.SCEEH.H’'. |
| CICSOPT | <u>NONE</u><br>*option-list* | Gives a list of additional CICS translator options. See the appropriate CICS application programming reference for information about translator options.<br><br>NONE gives no additional options. |
| CICSPRE | *prefix* | Gives the prefix for the CICS libraries. The library names are:<br><br>*prefix*.LOADLIB for translators<br>*prefix*.PL1LIBn for PL/I include<br>*prefix*.COBLIB for COBOL include<br><br>Leave this parameter blank to use CICSLLIB, CICSPLIB, CICSCOB.<br><br>The **default** is blank. |

*Table 19 (Page 3 of 13). General DSNH CLIST parameters*

| **DSNH CLIST** | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| CICSLLIB | *dsname* | Specifies the CICS load library. To use this library, leave the CICSPRE parameter blank.<br><br>The **default** is set on install panel DSNTIP3. |
| CICSPLIB | *dsname* | Specifies the CICS PL/I library. To use this library, leave the CICSPRE parameter blank.<br><br>The **default** is set on install panel DSNTIP3. |
| CICSCOB | *dsname* | Specifies the CICS COBOL library. To use this library, leave the CICSPRE parameter blank.<br><br>The **default** is set on install panel DSNTIP3. |
| CICSVER | 21<br>31<br><u>33</u><br>41 | Specifies the CICS release, as follows:<br><br>**Value** **CICS Release**<br>**21** Using CICS/MVS 2.1.2.<br>**31** Using CICS/ESA 3.1.1.<br>**33** Using CICS/ESA 3.2.1 or 3.3.0.<br>**41** Using CICS/ESA Version 4<br><br>The **default** is 33. |
| CICSXLAT | NO<br>YES | Tells whether to execute the CICS command translator. The option is effective only if you use RUN(CICS). You cannot use the option with the MARGINS option of the translator.<br><br>The **default** is YES. The DB2I panel **default** is NO. |
| CLIB<br>C<br>*n*LIB | <u>NONE</u><br>*dsname* | Specifies a data set that contains host language source statements to be included by the compiler or assembler. The parameters C*n*LIB (where *n* can be 2, 3, or 4) are extensions of CLIB, used to simplify passing a list of data set names.<br><br>Use NONE to specify no data set. |
| COBICOMP | *dsname* | Specifies the IBM COBOL data set required for compilation.<br><br>The **default** is '"IGY.V1R2M0.SIGYCOMP"'. |
| COBILINK | *dsname* | Specifies the IBM COBOL data set required for link edit.<br><br>The **default** is '"CEE.V1R5M0.SCEELKED"'. |
| COBIPLNK | *dsname* | Specifies the IBM COBOL data set required for prelink routines.<br><br>The **default** is '"CEE.V1R5M0.SCEERUN"'. |
| COBIPMSG | *dsname* | Specifies the IBM COBOL data set required for prelink messages.<br><br>The **default** is '"CEE.V1R5M0.SCEEMSGP(EDCPMSGE)"'. |
| COBLIB | *dsname* | Specifies the linkage editor include library to be used for OS/VS COBOL routines.<br><br>The **default** is '"SYS1.COBLIB"'. |
| COBLOAD | *dsname* | Specifies a data set that contains the OS/VS COBOL compiler load module.<br><br>*dsname* can include a member name.<br><br>The **default** is '"SYS1.LINKLIB(IKFCBL00)"'. |
| COBSOM | *dsname* | Specifies the IBM System Object Model (SOM) data set required for access to SOM objects.<br><br>The **default** is '"SOMMVS.V1R1M0.SGOSPLKD"'. |

Table 19 (Page 4 of 13). General DSNH CLIST parameters

| DSNH CLIST | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| COB2CICS | *dsname* | Specifies the linkage editor include library to be used for VS COBOL II CICS routines.<br>The **default** is '''SYS1.COB2CICS'''. |
| COB2LIB | *dsname* | Specifies the linkage editor include library to be used for the VS COBOL II or COBOL/370 routines.<br>The **default** is '''SYS1.V1R3.COB2LIB'''. |
| COB2LOAD | *dsname* | Specifies a data set that contains the VS COBOL II or COBOL/370 compiler load module.<br>*dsname* can include a member name.<br>The **default** is '''SYS1.V1R3.COB2COMP(IGYCRCTL)'''. |
| COMPILE | <u>YES</u><br>NO | Tells whether to execute the compiler or assembler if the precompile step is successful. |
| CONNECT | (1)<br>(<u>2</u>) | Specifies whether a CONNECT SQL statement should be processed as a type 1 CONNECT or a type 2 CONNECT statement. The DSNH(TSO CLIST) command does not accept the CT(1) and CT(2) abbreviations for this precompiler option.<br>The **default** is CONNECT(2). |
| CONTROL | <u>NONE</u><br>CONLIST<br>LIST<br>SYMLIST | CONTROL helps you trace the allocation of non-existent data sets. Use CONTROL if you have a problem without an obvious cause.<br>CONLIST displays CLIST commands after substitution for symbols and before command execution.<br>LIST displays TSO commands after substitution for symbols and before command execution.<br>SYMLIST displays all executable statements (TSO commands and CLIST statements) before substitution for symbols. |
| COPTION | <u>NONE</u><br>*string* | Gives a list of compiler or assembler options. For more information, see 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 on page 219.<br>NONE gives no options. |
| CPPCLASS | *dsname* | Specifies the data set containing C$^{++}$ class libraries.<br>The **default** is '''CBC.V3R1M0.SCLB3CPP'''. |
| CPPCLINK | *dsname* | Specifies the data set containing prelink utility modules used by the C$^{++}$ compiler.<br>The **default** is '''CEE.V1R4M0.SCEERUN(EDCPRLK)'''. |
| CPPCLLIB | *dsname* | Specifies the data set for C linkage editor automatic call library used by the C$^{++}$ compiler.<br>The **default** is '''CEE.V1R4M0.SCEELKED'''. |
| CPPCSLIB | *dsname* | Specifies data set containing the C compiler headers used by the C$^{++}$ compiler.<br>The **default** is '''CBC.V1R4M0.SCEEH.H'''. |
| CPPLLIB | *dsname* | Specifies the data set containing the C$^{++}$ prelink automatic call library.<br>The **default** is '''CEE.V1R4M0.SCEECPP'''. |

*Table 19 (Page 5 of 13). General DSNH CLIST parameters*

| DSNH CLIST | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| CPPPMSGS | *dsname* | Specifies the data set containing prelink utility error messages used by the C$^{++}$ compiler.<br>The **default** is '''CEE.V1R4M0.SCEEMSGP(EDCPMSGE)'''. |
| CPPSLIB | *dsname* | Specifies the data set containing C$^{++}$ header files for class libraries.<br>The **default** is '''CBC.V3R1M0.SCLB3H.H'''. |
| CPPUTIL | *dsname* | Specifies the data set containing procedures to set up and execute the C$^{++}$ compiler.<br>The **default** is '''CBC.V3R1M1.SCBC3UTL'''. |
| DATE | ISO<br>JIS<br>USA<br>EUR<br>LOCAL | Specifies the format of date values that are to be returned overriding the format specified as the location default.<br>The **default** is the value supplied when DB2 was installed, and is written in the data-only load module, DSNHDECP. |
| DBRMLIB | <u>DEFAULT</u><br>*dsname(member)*<br>NONE | Specifies the partitioned data set, and optionally a member name, that contains the DBRM library and member name used during the DB2 precompile step. Because you can specify an individual DBRM member and library names during each individual phase, you must use the DBRMLIB parameter and associated prefixes to identify a specific phase. DBRMLIB specifies the DBRM library and member defined on the DBRMLIB DD statement during DB2 precompiler processing.<br>DEFAULT indicates that the same DBRM library data set defined for the DB2 precompiler process (DBRMLIB(*parameter*)) is also used on the LIBRARY(*dsname*) subcommand keyword. If the precompiler DBRMLIB was not specified, then the default generated DBRMLIB library based upon the INPUT data set name is used.<br>*dsname* is generated using the DSNH OUTNAME parameter value, or its default TEMP, with the constant DBRM appended to the prefix; for example, *outname*.DBRM or TEMP.DBRM.<br>*member* is obtained from the data set member name specified on the DSNH INPUT parameter, or from the data set name as follows:<br>Given INPUT(*outname*.DBRM(*dbrmmem*)):<br>• *outname*.DBRM(*dbrmmem*) - If member name is specified<br>• *outname*.DBRM(*dbrm*) - If no member name is specified<br>NONE indicates that no LIBRARY(*dsname*) subcommand keyword is specified on invocation. |
| DECARTH | <u>DEFAULT</u><br>15<br>31 | Sets the maximum precision of decimal numbers.<br>DEFAULT designates the value chosen, during installation, for the DECIMAL ARITHMETIC field on the APPLICATION PROGRAMMING DEFAULTS panel.<br>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.<br>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.<br>DECARTH is ignored for Fortran. |

*Table 19 (Page 6 of 13). General DSNH CLIST parameters*

| DSNH CLIST | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| DECIMAL | COMMA<br>PERIOD | Gives the decimal point indicator for decimal and floating point literals. DECIMAL is valid only for COBOL programs; PERIOD is forced for all other programs.<br><br>COMMA makes the indicator a comma.<br><br>PERIOD makes the indicator a period.<br><br>The **default** is the value of the DECIMAL POINT field, set on the DB2 APPLICATION PROGRAMMING DEFAULTS panel during installation. |
| DELIMIT | <u>DEFAULT</u><br>APOST<br>QUOTE | Sets the APOST or QUOTE precompiler option to indicate the string delimiter used within host language statements. DELIMIT is effective only for COBOL programs; APOST is forced for all other programs.<br><br>DEFAULT designates the value chosen, during installation, for the STRING DELIMITER field on the APPLICATION PROGRAMMING DEFAULTS panel.<br><br>APOST specifies the apostrophe as the string delimiter for host language statements.<br><br>QUOTE specifies a quotation mark as the string delimiter for host language statements. |
| ENTRY | *entry-name* | Specifies the entry point assigned by the linkage editor.<br><br>The **default** depends on the host language and the value of RUN.<br><br>• For the PL/I language, the ENTRY value default is:<br>– NONE if the RUN value is CICS<br>– PLISTART for any other RUN value.<br>• For the ASM language, the ENTRY value default is DLITASM if the RUN value is IMS.<br>• For COBOL, the ENTRY value default is DLITCBL if the RUN value is IMS.<br>• For any other language, the ENTRY value default is NONE (no specified entry point) for any RUN value. |
| FLAG | <u>I</u><br>C<br>E<br>W | Tells what messages you want to see. Use one of the values listed to show messages of the corresponding types:<br><br>• I All informational, warning, error, and completion messages<br>• W Only warning, error, and completion messages<br>• E Only error and completion messages<br>• C Only completion messages |
| FORTLIB | *dsname* | Specifies the linkage editor include library to be used for Fortran routines.<br><br>The **default** is '''SYS1.VSF2FORT'''. |
| FORTLOAD | *dsname* | Specifies a data set that contains the VS Fortran compiler load module. *dsname* can include a member name.<br><br>The **default** is '''SYS1.VSF2VCOMP(FORTVS2)'''. |
| GRAPHIC | <u>NONE</u><br>NO<br>YES | Specifies the value of the DSNHDECP MIXED option for the precompiler.<br><br>NONE indicates that the default specified during install is used.<br><br>NO indicates that the data is **not** mixed DBCS.<br><br>YES indicates that all character data can be mixed DBCS.<br><br>GRAPHIC is ignored for C. |

*Table 19 (Page 7 of 13). General DSNH CLIST parameters*

| DSNH CLIST | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| HOST | ASM<br>C<br>CPP<br>COBOL<br>COB2<br>IBMCOB<br>FORTRAN<br>PLI | Defines the host language within which SQL statements are embedded.<br>If your program fits one of the following descriptions, you cannot use DB2I to prepare it:<br>• An IBM COBOL for MVS & VM program that uses object-oriented extensions<br>• A C$^{++}$ program that uses object-oriented extensions and consists of more than one compilation unit<br>The **default** is the value of the LANGUAGE DEFAULT field, set on the DB2 APPLICATION PROGRAMMING DEFAULTS panel during installation. |
| IMSPRE | *prefix* | Sets the prefix for RESLIB, used for routines to be included by the linkage editor for IMS.<br>The **default** is IMSVS. |
| INPUT | *dsname* | Specifies the data set that contains the host language source and SQL statements.<br>*dsname* can include a member name. |
| LINECOUNT | *integer* | Tells how many lines, including headings, are to be printed on each page of printed output.<br>The **default** is 60. |
| LINK | <u>YES</u><br>NO | Tells whether to execute the linkage editor upon successful completion of compilation or assembly.<br>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 specified by the RUN parameter.<br>NO indicates that linkage editor processing is to be bypassed. |
| LLIB<br>L*n*LIB | <u>NONE</u><br>*dsname* | Specifies a data set that contains object or load modules to be included by the linkage editor. The parameters L*n*LIB (where *n* can be 2, 3, or 4) are extensions of LLIB, used to simplify passing a list of data set names.<br>The LLIB and L*n*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.<br>Use NONE to specify no data set. |
| LOAD | *dsname* | Specifies a data set that is to contain the output from the linkage editor (the load module).<br>*dsname* can include a member name.<br>The **default** is RUNLIB.LOAD. |
| LOPTION | <u>NONE</u><br>*string* | Gives a list of linkage editor options. For information on options you can use, see the appropriate MVS/ESA publication.<br>Use NONE to give no options. |
| MACRO | <u>YES</u><br>NO | Tells 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. |

*Table 19 (Page 8 of 13). General DSNH CLIST parameters*

| DSNH CLIST | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| NOFOR | NO (default)<br>YES | Allows elimination of all FOR UPDATE OF clauses in static SQL.<br>When NOFOR is in effect, the FOR UPDATE OF clause is optional. Positioned updates can be made to any columns that the user has authority to update.<br>When NOFOR is not in effect, any query appearing 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.<br>The option is implied when the STDSQL(YES) option is in effect. |
| OPTIONS | NO (default)<br>YES | Tells whether to print the options used when executing the precompiler or the CICS command translator with the output listing. |
| OUTNAME | TEMP (default)<br>*string* | Gives a prefix used to form intermediate data set names.<br>*string* must not be enclosed between apostrophes and must not have the same initial character as the *dsname* for INPUT. It cannot contain special characters. |
| PARMS | NONE (default)<br>*string* | Gives a parameter string to be passed to the compiled program during its execution; the run time execution environment requested is TSO. If CAF is specified as the run time execution environment, this parameter is ignored.<br>Use NONE to pass no parameter string. |
| PASS | ONE or 1<br>TWO or 2 | Tells 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.<br>The **default** is ONE or 1 for PL/I and C.<br>The **default** is TWO or 2 for assembler. |
| PCLOAD | *dsname* | Specifies the precompiler load module.<br>*dsname* can include a member name.<br>The **default** is '''*prefix*.SDSNLOAD(DSNHPC)'''. |
| PLAN | *plan-name* | Specifies the application plan created by the bind process.<br>The **default** plan name is the first of the following available choices defined in the INPUT data set:<br>• DBRM member name<br>• Leftmost qualifier<br>*plan-name* must not be DEFAULT.<br>If no name is found, a plan is not created. |
| PLIB<br>P*n*LIB | NONE (default)<br>*dsname* | Specifies the data set that contains host language source or SQL statements included by the SQL INCLUDE statement during precompilation. The parameters P*n*LIB (where *n* can be 2, 3, or 4) are extensions of PLIB, used to simplify passing a list of data set names.<br>Use NONE to specify no data set. |
| PLI2LIB | *dsname* | Specifies the linkage editor common library used for PL/I routines.<br>The **default** is '''SYS1.SIBMBASE'''. |

*Table 19 (Page 9 of 13). General DSNH CLIST parameters*

| DSNH CLIST | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| PLILIB | *dsname* | Specifies the linkage editor base library used for PL/I routines.<br><br>The **default** is '"SYS1.PLIBASE"'. |
| PLILOAD | *dsname* | Specifies a data set that contains the PL/I optimizing compiler load module.<br><br>*dsname* can include a member name.<br><br>The **default** is '"SYS1.LINKLIB(IEL0AA)"'. |
| POPTION | <u>NONE</u><br>*string* | Gives a list of the C compiler language prelink utility options. For information on the options provided, refer to the *IBM SAA AD/Cycle C/370 User's Guide*.<br><br>Use NONE to give no options. |
| PRECOMP | <u>YES</u><br>NO | Tells whether to precompile. |
| PRELINK | <u>YES</u><br>NO | Tells 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.<br><br>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.<br><br>Descriptions of the prelink process for C and IBMCOB are presented in their respective language publications.<br><br>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. |
| PRINT | <u>NONE</u><br>*dsname*<br>LEAVE<br>TERM | Tells where to send printed output, including the lists of options, source, cross-reference, error, and summary information.<br><br>NONE omits printed output.<br><br>*dsname* specifies a data set to be used for the output. Do not enclose *dsname* between apostrophes. The current user profile is prefixed to *dsname*. The following suffixes are also added:<br>• SYSCPRT.LIST for PL/I macro listings (these listings are overwritten by the compiler listings)<br>• PCLIST for precompiler listings<br>• CXLIST for CICS command translator listings<br>• LIST for compiler listings<br>The PRINT parameter is ignored for the compile step when HOST(CPP) is specified.<br>• SYSOUT.PRELLIST for C prelink utility listings<br>• LINKLIST for link-edit listings<br><br>LEAVE sends output to the specified print data set. You can allocate the print data set:<br>• Dynamically<br>• In the JCL used to run the DSNH CLIST (if in batch mode)<br>• With the TSO ALLOCATE command (before running DSNH)<br><br>TERM sends output to the terminal. |

*Table 19 (Page 10 of 13). General DSNH CLIST parameters*

| DSNH CLIST | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| PSECSPAC | *integer* | Tells the amount of secondary space to allocate for print data sets, in the units given by SPACEUN.<br><br>The **default** is 20. |
| PSPACE | *integer* | Tells the primary size of the print data sets in the units given by SPACEUN.<br><br>The **default** is 20. |
| RCTERM | *integer* | Gives the least value of the return code from the precompile step that prevents execution of later steps.<br><br>The **default** is 8. |
| RUN | <u>TSO or YES</u><br>BATCH or NO<br>CAF<br>CICS<br>IMS<br>RRSAF | Tells 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.<br><br>TSO or YES indicate that the application program is to be scheduled for execution in the TSO environment, and execute the compiled program.<br><br>BATCH or NO indicate that the application program is not to be scheduled for execution, and default to TSO as the execution environment.<br><br>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)<br><br>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.)<br><br>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.)<br><br>RRSAF indicates that the application program is not to be scheduled for execution, and identifies RRSAF as the RUN time execution environment. (RRSAF applications cannot run in TSO.) |
| RUNIN | <u>TERM</u><br>*dsname*<br>LEAVE<br>NONE | Tells where to get input for the RUN step.<br><br>TERM gets input from the terminal.<br><br>*dsname* specifies a data set to be used for the input.<br><br>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 for any other cases.<br><br>NONE allocates no input file. |
| RUNOUT | <u>TERM</u><br>*dsname*<br>LEAVE<br>NONE | Tells where to send output from the RUN step.<br><br>TERM sends output to the terminal.<br><br>*dsname* specifies a data set to receive output.<br><br>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 for any other cases.<br><br>NONE allocates no output file for the RUN step. |

*Table 19 (Page 11 of 13). General DSNH CLIST parameters*

| **DSNH CLIST** | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| SOMDLLI | *dsname* | Tells the name to be passed of the SOM/MVS DLL import library.<br><br>The **default** is '"SOM.SGOSIMP"'. |
| SOURCE | NO<br>YES | Tells whether the source code and diagnostics are to be printed with output from the precompiler, CICS command translator, and compiler. |
| SPACEUN | TRACK<br>CYLINDER | Specifies the unit of space for PSPACE and WSPACE.<br><br>TRACK makes the space unit one track.<br><br>CYLINDER makes the space unit one cylinder. |
| SQL | DB2<br>ALL | Interprets SQL statements and checks syntax for use by either DB2 for OS/390 or other database management systems.<br><br>DB2 indicates that SQL statements are to be interpreted and syntax is to be checked for use by DB2. SQL(DB2) is the recommended mode for DRDA access when the server is a DB2 subsystem.<br><br>ALL indicates that SQL statements are to be interpreted for use by database management systems that are not DB2 for OS/390. 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 OS/390.<br><br>The **default** is SQL(DB2). |
| SQLDELIM | DEFAULT<br>APOSTSQL<br>QUOTESQL | Sets the APOSTSQL or QUOTESQL precompiler option, to specify 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.<br><br>This parameter is effective only for COBOL. For PL/I, Fortran, and assembler language programs, the precompiler forces the APOSTSQL option.<br><br>DEFAULT designates the value chosen, during installation, for the SQL STRING DELIMITER field on the APPLICATION PROGRAMMING DEFAULTS panel.<br><br>APOSTSQL specifies that the string delimiter is the apostrophe (') and the escape character is the quotation mark (").<br><br>QUOTESQL specifies that the string delimiter is the quotation mark (") and the escape character is the apostrophe ('). |
| SQLFLAG | IBM or SAA<br>STD or 86<br>*ssname*<br>*qualifier* | Specifies the standard to be used to check the syntax of SQL statements. Deviations from the standard are flagged by informational messages written to the precompiler output listing.<br><br>IBM or SAA requests the use of the IBM SQL Version 2 syntax.<br><br>STD or 86 requests the use of the SQL92 Entry Level syntax.<br><br>*ssname* requests full semantics checking for catalog access using the specified DB2 subsystem name. If *ssname* is not specified, only syntax checking is performed.<br><br>*qualifier* specifies the qualifier to be used for unqualified object names. If *qualifier* is specified, *ssname* must always be specified first. If *qualifier* is not specified, the **default** is the authorization ID of the process that executed the precompiler. |

*Table 19 (Page 12 of 13). General DSNH CLIST parameters*

| **DSNH CLIST** | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| STDSQL | <u>NO</u><br>YES or 86 | Interprets SQL using a subset of ANSI rules.<br><br>NO specifies that DB2 rules are used.<br><br>YES or 86 automatically implies that the NOFOR option is used. |
| SUFFIX | <u>YES</u><br>NO | Tells 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 *userid.name*.COBOL, *userid.name*.PLI, *userid.name*.Fortran, or *userid.name*.ASM. |
| SYSTEM | *subsystem-name* | Gives the DB2 subsystem name as it is known to MVS.<br><br>The **default** is the installation-defined subsystem name (often DSN). |
| TERM | <u>TERM</u><br>*dsname*<br>LEAVE<br>NONE | Tells where to send terminal output, including error information, error statements, and summary information.<br><br>TERM sends output to the terminal.<br><br>*dsname* specifies a data set to be used for terminal output. Do not enclose *dsname* between apostrophes. The following suffixes are added to *dsname*:<br>• PCTERM for precompiler output<br>• LIST for compiler output<br><br>LEAVE sends the output to the current allocation for SYSTERM.<br><br>NONE omits terminal output. |
| TIME | ISO<br>JIS<br>USA<br>EUR<br>LOCAL | Specifies the format for time values that are to be returned, overriding the format specified as the location default.<br><br>There is **no default**, because this option overrides the default previously specified. |
| VERSION | *version-id*<br>AUTO | Specifies the name of the version ID for the program and associated DBRM during the DB2 precompile.<br><br>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.<br><br>The **default** is no version ID if specified at precompiler invocation. |
| WORKUNIT | *unit* | Tells what device to use for print and work data sets.<br><br>*unit* can be a unit name or a device type.<br><br>The **default** in batch mode is any eligible device.<br><br>The **default** in any other mode is the UADS unit name for the current TSO user. |
| WSECSPAC | *integer* | Tells the amount of secondary space to allocate for work data sets, in the units given by SPACEUN.<br><br>The **default** is 20. |
| WSPACE | *integer* | Tells the primary size of the work data sets in the units given by SPACEUN.<br><br>The **default** is 20. |

*Table 19 (Page 13 of 13). General DSNH CLIST parameters*

| DSNH CLIST | | |
|---|---|---|
| **Parameter** | **Value** | **Comments** |
| XLIB | *dsname* | Specifies the linkage editor include library to be used for DB2 routines.<br><br>The **default** is '''*prefix*.SDSNLOAD'''. |
| XREF | <u>NO</u><br>YES | Tells whether a sorted cross-reference listing of symbolic names used in source statements is to be printed with output from the precompiler. |

## DSNH/DSN subcommand summary

The following tables 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 (Page 1 of 4). DSNH CLIST/ BIND PLAN subcommand summary*

| DSNH CLIST | | BIND PLAN subcommand | | |
|---|---|---|---|---|
| **Parameter** | **Value** | **Keyword** | **Value** | **Comments** |
| ACQUIRE | <u>USE</u><br>ALLOCATE | ACQUIRE | <u>USE</u><br>ALLOCATE | |
| ACTION | <u>REPLACE</u><br>ADD | ACTION | <u>REPLACE</u><br>ADD | |
| BDMEM | <u>DEFAULT</u>[1]<br>*dbrm-member-name*<br>NONE[2] | MEMBER | *dbrm-member-name* | [1] DBRM member name obtained from one of the following sources, in the order listed:<br>• BDBRMLIB member name<br>• DBRMLIB member name<br>• INPUT member name, or generated using *dsname*.<br><br>[2] Keyword is not specified on subcommand invocation. |
| BIND | <u>YES</u>[1]<br>NO[2] | *(command-verb)* | | [1] Execute BIND PLAN subcommand.<br><br>[2] Do not execute BIND PLAN subcommand. |
| BLIB | <u>NONE</u>[1]<br>*dsname* | LIBRARY | *dbrm-pds-name* | [1] Keyword is not specified on subcommand invocation. |

Table 20 (Page 2 of 4). DSNH CLIST/ BIND PLAN subcommand summary

| DSNH CLIST | | BIND PLAN subcommand | | |
|---|---|---|---|---|
| **Parameter** | **Value** | **Keyword** | **Value** | **Comments** |
| B*n*LIB[1] | <u>NONE</u>[2]<br>*dsname* | LIBRARY | *list of dbrm-pds-names* | [1] *n* can be 2, 3, 4, 5, 6, 7, or 8. Specify the first data set name using the BLIB parameter and any additional data set names using this parameter.<br>[2] No additional data set names. |
| BMEM[1] | <u>NONE</u>[2]<br>*list of dbrm-member -names* | MEMBER | *list of dbrm-member-names* | [1] Specify the first DBRM member name using the BDMEM parameter and any additional member names individually using this parameter.<br>[2] No additional DBRM member names. |
| CACHESIZE | <u>NONE</u>[1]<br>*decimal-value*[2] | CACHESIZE | *decimal-value*[2] | [1] The size is provided by the subsystem.<br>[2] Specify a size from 0 to 4096 bytes. |
| CICS | <u>NONE</u>[1]<br>*application-ids* | CICS | *application-ids* | [1] Keyword is not specified on subcommand invocation. |
| CURRENTDATA | <u>YES</u><br>NO<br>NONE | CURRENTDATA | <u>YES</u><br>NO | |
| CURRENTSERVER | <u>NONE</u>[1]<br>*location-name* | CURRENTSERVER | *location-name* | |
| DBPROTOCOL | <u>NONE</u><br>DRDA<br>PRIVATE | DBPROTOCOL | <u>DRDA</u><br>PRIVATE | |
| BDBRMLIB | <u>DEFAULT</u>[1]<br>*dsname(member)*<br>NONE[2] | LIBRARY | *dbrm-pds-name* | [1] The precompiler DBRMLIB data set is used. If the precompiler DBRMLIB is not specified, then the default-generated DBRMLIB library based upon the INPUT data set is used.<br>[2]Keyword is not specified on subcommand invocation. |
| DEFER | <u>NONE</u>[1]<br>PREPARE | DEFER | PREPARE | [1] Keyword is not specified on subcommand invocation. |
| DEGREE | <u>1</u><br>ANY | DEGREE | <u>1</u><br>ANY | |

*Table 20 (Page 3 of 4). DSNH CLIST/ BIND PLAN subcommand summary*

| DSNH CLIST | | BIND PLAN subcommand | | |
|---|---|---|---|---|
| **Parameter** | **Value** | **Keyword** | **Value** | **Comments** |
| DISABLE | NONE | DISABLE | NONE | |
| | BATCH | | BATCH | |
| | CICS | | CICS | |
| | DB2CALL | | DB2CALL | |
| | IMS | | IMS | |
| | DLIBATCH | | DLIBATCH | |
| | IMSBMP | | IMSBMP | |
| | IMSMPP | | IMSMPP | |
| | RRSAF | | RRSAF | |
| DISCONNECT | EXPLICIT | DISCONNECT | EXPLICIT | |
| | AUTOMATIC | | AUTOMATIC | |
| | CONDITIONAL | | CONDITIONAL | |
| DLIBATCH | NONE[1] | DLIBATCH | *connection-name* | [1]Keyword is not specified on subcommand invocation. |
| | *list of connection-ids* | | | |
| DYNAMICRULES | RUN | DYNAMICRULES | RUN | |
| | BIND | | BIND | |
| ENABLE | NONE | ENABLE | NONE | |
| | * | | * | |
| | BATCH | | BATCH | |
| | CICS | | CICS | |
| | DB2CALL | | DB2CALL | |
| | IMS | | IMS | |
| | DLIBATCH | | DLIBATCH | |
| | IMSBMP | | IMSBMP | |
| | IMSMPP | | IMSMPP | |
| | RRSAF | | RRSAF | |
| EXPLAIN | NO | EXPLAIN | NO | |
| | YES | | YES | |
| FLAG | I | FLAG | I | |
| | C | | C | |
| | E | | E | |
| | W | | W | |
| IMSBMP | NONE[1] | IMSBMP | *imsid* | [1]Keyword is not specified on subcommand invocation. |
| | *imsid* | | | |
| IMSMPP | NONE[1] | IMSMPP | *imsid* | [1]Keyword is not specified on subcommand invocation. |
| | *imsid* | | | |
| ISOLATION | RR | ISOLATION | RR | |
| | RS | | RS | |
| | CS | | CS | |
| | UR | | UR | |
| KEEPDYNAMIC | NO | KEEPDYNAMIC | NO | |
| | YES | | YES | |
| NODEFER | NONE[1] | NODEFER | PREPARE | [1]Keyword is not specified on subcommand invocation. |
| | PREPARE | | | |

Table 20 (Page 4 of 4). DSNH CLIST/ BIND PLAN subcommand summary

| DSNH CLIST | | BIND PLAN subcommand | | |
|---|---|---|---|---|
| **Parameter** | **Value** | **Keyword** | **Value** | **Comments** |
| OPTHINT | **(' ')**<br>('*hint-id*') | OPTHINT | **(' ')**<br>('*hint-id*') | |
| OWNER | NONE[1]<br>*authorization-id* | OWNER | *authorization-id* | [1]Keyword is not specified on subcommand invocation. |
| PATH | *(schema-name)*<br>(**USER**)<br>*(schema-name,*<br>**USER***...)* | PATH | *(schema-name)*<br>(**USER**)<br>*(schema-name,*<br>**USER***...)* | |
| PKLIST | NONE[1]<br>*list of collection-ids and package- names* | PKLIST | *list of collection-ids and package- names* | [1] The package names are not specified on subcommand invocation. |
| PLAN | *plan-name*[1] | PLAN<br>*(primary-keyword)* | *plan-name* | [1] *plan-name* must not be DEFAULT. The default *plan-name* is the first of the following available choices defined in the INPUT data set:<br>• DBRM member name<br>• Leftmost qualifier<br>If no name is found, a plan is not created. |
| QUALIFIER | NONE[1]<br>*implicit-qualifier* | QUALIFIER | *qualifier-name* | [1]Keyword is not specified on subcommand invocation. |
| RELEASE | COMMIT<br>DEALLOCATE | RELEASE | COMMIT<br>DEALLOCATE | |
| REOPT | NONE[1]<br>VARS | NOREOPT<br>REOPT | VARS<br>VARS | [1]Keyword is not specified on subcommand invocation. |
| RETAIN | NO[1]<br>YES[2] | RETAIN | | [1]Keyword is not specified on subcommand invocation.<br>[2] Keyword is specified on subcommand invocation. |
| SQLRULES | DB2<br>STD | SQLRULES | DB2<br>STD | |
| VALIDATE | RUN<br>BIND | VALIDATE | RUN<br>BIND | |

## DSNH CLIST/BIND PACKAGE subcommand comparison

Table 21 (Page 1 of 4). DSNH CLIST/ BIND PACKAGE subcommand summary

| DSNH CLIST | | BIND PACKAGE subcommand | | |
|---|---|---|---|---|
| **Parameter** | **Value** | **Keyword** | **Value** | **Comments** |
| PACTION | REPLACE<br>ADD | ACTION | REPLACE<br>ADD | |

*Table 21 (Page 2 of 4). DSNH CLIST/ BIND PACKAGE subcommand summary*

| DSNH CLIST | | BIND PACKAGE subcommand | | |
|---|---|---|---|---|
| **Parameter** | **Value** | **Keyword** | **Value** | **Comments** |
| PBIND | NO[1]<br>YES[2] | *(command-verb)* | | [1] Do not execute BIND PACKAGE subcommand.<br>[2] Execute BIND PACKAGE subcommand. |
| PCICS | NONE[1]<br>*application-ids* | CICS | *application-ids* | [1]Keyword is not specified on subcommand invocation. |
| COPY | NONE[1]<br>*collection-id.*<br>*package-id* | COPY | *collection-id.*<br>*package-id* | [1]Keyword is not specified on subcommand invocation. |
| COPYVER | *version-id* | COPYVER | *version-id* | |
| PCURRENTDATA | NO<br>YES<br>NONE | CURRENTDATA | YES<br>NO | |
| PDBPROTOCOL | NONE<br>DRDA<br>PRIVATE | DBPROTOCOL | DRDA<br>PRIVATE | |
| PDBRMLIB | DEFAULT[1]<br>*dsname(member)*<br>NONE[2] | LIBRARY | *dbrm-pds-name* | [1] The precompiler DBRMLIB data set is used. If the pre-compiler DBRMLIB is not specified, then the default-generated DBRMLIB library based upon the INPUT data set is used.<br>[2]Keyword is not specified on subcommand invocation. |
| PDEFER | NONE[1]<br>PREPARE | DEFER | PREPARE | [1]Keyword is not specified on subcommand invocation. |
| PDEGREE | 1<br>ANY | DEGREE | 1<br>ANY | |
| PDISABLE | NONE<br>BATCH<br>CICS<br>DB2CALL<br>IMS<br>DLIBATCH<br>IMSBMP<br>IMSMPP<br>REMOTE<br>RRSAF | DISABLE | NONE<br>BATCH<br>CICS<br>DB2CALL<br>IMS<br>DLIBATCH<br>IMSBMP<br>IMSMPP<br>REMOTE<br>RRSAF | |
| PDLIBATCH | NONE[1]<br>*list of*<br>*connection-ids* | DLIBATCH | *connection-name* | [1]Keyword is not specified on subcommand invocation. |

Table 21 (Page 3 of 4). DSNH CLIST/ BIND PACKAGE subcommand summary

| DSNH CLIST | | BIND PACKAGE subcommand | | |
|---|---|---|---|---|
| **Parameter** | **Value** | **Keyword** | **Value** | **Comments** |
| PDMEM | DEFAULT[1]<br>*dbrm-member-name*<br>NONE[2] | MEMBER | *dbrm-member-name* | [1] DBRM member name obtained from one of the following sources, in the order listed:<br>• PDBRMLIB member name<br>• DBRMLIB member name<br>• INPUT member name, or generated using *dsname*<br>[2]Keyword is not specified on subcommand invocation. |
| PDYNAMICRULES | NONE<br>RUN<br>BIND<br>DEFINE<br>INVOKE | DYNAMICRULES | RUN<br>BIND<br>DEFINE<br>INVOKE | |
| PENABLE | NONE<br>*<br>BATCH<br>CICS<br>DB2CALL<br>IMS<br>DLIBATCH<br>IMSBMP<br>IMSMPP<br>REMOTE<br>RRSAF | ENABLE | NONE<br>*<br>BATCH<br>CICS<br>DB2CALL<br>IMS<br>DLIBATCH<br>IMSBMP<br>IMSMPP<br>REMOTE<br>RRSAF | |
| EXPLAIN | NO<br>YES | EXPLAIN | NO<br>YES | |
| PFLAG | I<br>C<br>E<br>W | FLAG | I<br>C<br>E<br>W | |
| PIMSBMP | NONE[1]<br>*imsid* | IMSBMP | *imsid* | [1]Keyword is not specified on subcommand invocation. |
| PIMSMPP | NONE[1]<br>*imsid* | IMSMPP | *imsid* | [1]Keyword is not specified on subcommand invocation. |
| PISOLATION | NONE[1]<br>RR<br>RS<br>CS<br>UR<br>NC | ISOLATION[1] | RR<br>RS<br>CS<br>UR<br>NC | [1] For local packages, the default value is the same as that of the plan appended at execution time. For remote packages, the default value is RR. |
| PKEEPDYNAMIC | NONE<br>NO<br>YES | KEEPDYNAMIC | NO<br>YES | |
| PNODEFER | NONE[1]<br>PREPARE | NODEFER | PREPARE | [1]Keyword is not specified on subcommand invocation. |

*Table 21 (Page 4 of 4). DSNH CLIST/ BIND PACKAGE subcommand summary*

| **DSNH CLIST** | | **BIND PACKAGE subcommand** | | |
|---|---|---|---|---|
| **Parameter** | **Value** | **Keyword** | **Value** | **Comments** |
| POPTHINT | **(' ')**<br>('*hint-id*') | OPTHINT | **(' ')**<br>('*hint-id*') | |
| POWNER | NONE1<br>*authorization-id* | OWNER | *authorization-id* | 1Keyword is not specified on subcommand invocation. |
| PACKAGE | DEFAULT1<br>*location-name. collection-id* | PACKAGE | *location-name. collection-id* | 1 Member name defined in the INPUT parameter data set, or the data set name if no member name was specified. |
| PPATH | *(schema-name)*<br>(**USER**)<br>*(schema-name,* **USER***, ...)* | PATH | *(schema-name)*<br>(**USER**)<br>*(schema-name,* **USER***, ...)* | |
| PQUALIFIER | NONE1<br>*implicit-qualifier* | QUALIFIER | *qualifier-name* | 1Keyword is not specified on subcommand invocation. |
| PRELEASE | NONE1<br>COMMIT<br>DEALLOCATE | RELEASE1 | COMMIT<br>DEALLOCATE | 1 For local packages, the default value is the same as that of the plan appended at execution time. For remote packages, the default value is NONE. |
| REOPT | NONE1<br>VARS | NOREOPT<br>REOPT | VARS<br>VARS | 1Keyword is not specified on subcommand invocation. |
| REMOTE | NONE1<br>*location-name, <luname>* | REMOTE | *network-name* | 1Keyword is not specified on subcommand invocation. |
| REPLVER | NONE1<br>*version-id* | REPLVER | *version-id* | 1 *version-id* is not specified on subcommand invocation. |
| SQLERROR | NOPACKAGE<br>CONTINUE | SQLERROR | NOPACKAGE<br>CONTINUE | |
| PVALIDATE | RUN<br>BIND | VALIDATE | RUN<br>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 (Language Environment for MVS & VM) 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 Table 22 on page 220.

*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 have 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:*** There can be at most eight bind libraries, four precompile libraries, four compile libraries, and four link-edit libraries.

***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:*** SQL host variables must be explicitly defined.

***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 insures that the temporary and intermediate data sets are allocated to the correct devices.

## Examples

***Example 1:*** Precompile, bind, compile, link-edit, and run 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 be put into the data set *prefix*.RUNLIB.LOAD(DSN8BC4).
- The plan name is DSN8BC61 for the bind and run.
- DCLGEN data from *prefix*.SRCLIB.DATA is required for the precompile.

This example assumes that 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(DSN8BC61) -
  PLIB('prefix.SRCLIB.DATA'')
```

***Example 2:*** Precompile, bind, compile, and link-edit the program in data set *prefix*.SDSNSAMP.PLI(DSN8BP4).

- The program is written in PL/I; the macro pass is not needed.
- The PL/I compiler options MAP and LIST are to be used.
- Additional load modules to be included are in *prefix*.RUNLIB.LOAD and *prefix*.SDSNSAMP.
- The PL/I optimizing compiler load module is in library SYS2.LINKLIB(IEL0AA).
- The DB2 subsystem identifier is SSTR.
- The load module is put into the data set *prefix*.RUNLIB.LOAD(DSN8BC4).
- Printed output is sent to the following data sets:

| **Output** | **Data set** |
|---|---|
| Precompiler listings | *prefix*.PROG.PCLIST |
| Compiler listings | *prefix*.PROG.LIST |
| Link edit listings | *prefix*.PROG.LINKLIST |

- The plan name is DSN8BC61 for the bind and run.
- The DCLGEN data from *prefix*.SRCLIB.DATA is required for the precompile.

```
DSNH INPUT('prefix.SDSNSAMP(DSN8BP4)'') -
  HOST(PLI) MACRO(NO) -
  COPTION ('MAP LIST') -
  LLIB('prefix.RUNLIB.LOAD'') -
  L2LIB('prefix.SDSNSAMP'') -
  PLILOAD('SYS2.LINKLIB(IEL0AA)'') -
  SYSTEM(SSTR) -
  LOAD('prefix.RUNLIB.LOAD'') -
  PRINT(PROG) -
  PLAN(DSN8BC61) -
  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 above, except that SMITH would be used as the first level qualifier. For example, the compiler listings would have gone to SMITH.PROG.LIST.

***Example 3:*** Invocation of the DB2-C sample application program *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 | *user_id*.TEMP.PCLIST |
  | Compiler listings | *user_id*.TEMP.SYSCPRT.LIST |
  | Prelink utility listings | *user_id*.TEMP.SYSOUT.PRELLIST |
  | Link-edit listings | *user_id*.TEMP.LINKLIST |

- The following C DD names are allocated based on the PRINT keyword value:

  | **DD name** | **Allocation** |
  |---|---|
  | SYSCPRT | Used in the compile step |
  | SYSUT10 | Used in the compile step |
  | SYSOUT | Used in the prelink step. |

  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 insure 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)
```

## 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.

## Example

End the DSN session and return to TSO.

```
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
```

## 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 only removed 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 executing applications using the package finish executing.

For additional information about packages, see Section 6 of *DB2 Application Programming and SQL Guide*.

## 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, the privilege set of the process must include one of the following:

- Ownership of the package
- BINDAGENT privilege granted by the owner of the package
- SYSCTRL or 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

```
►►──FREE PACKAGE──────────────────────────────────────────────────────────────►

►──(──┬──┌─,──────────────────────────────────────────────────────────────┐──┬──)───────►
      │  ▼                                                                 │  │
      │  ┬────────────────┬──┬─collection-id─┬─.─┬─package-id─┬──┬─────────────────────┬┘  │
      │  └─location-name.─┘  └─*─────────────┘   └─*──────────┘  └─.(─┬────────────┬─)─┘   │
      │                                                               ├─version-id─┤       │
      │                                                               └─*──────────┘       │
      └─*──────────────────────────────────────────────────────────────────────────────┘

►──┬─────────────────────┬────────────────────────────────────────────────────►◄
   └─FLAG(─┬─I─┬─)───────┘
           ├─W─┤
           ├─E─┤
           └─C─┘
```

## 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 must be defined. See Section 3 of *DB2 Installation Guide* for information on how to define a location name within SYSIBM.LOCATIONS.

The **default** is the local DB2 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.)

# *package-id* **or (*)**
Identifies the package to be freed. There is no default.

You can use an asterisk (*) to free all local packages in *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 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*, then 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.

DBRMs created before DB2 Version 2 Release 3 use an empty string for *version-id* by default.

**(*)** 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 to see. 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 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.

| For more information about dropping triggers, see Chapter 6 of *DB2 SQL Refer-*
| *ence*.

## Examples

***Example 1:*** Free version *newver* of the package TEST.DSN8BC61 located at USIBMSTODB22. Generate only warning, error, and completion messages (not informational messages).

```
FREE PACKAGE (USIBMSTODB22.TEST.DSN8BC61.(newver)) FLAG(W)
```

***Example 2:*** Free all packages at the local server in the collection named TESTCOLLECTION.

```
FREE PACKAGE (TESTCOLLECTION.*)
```

## 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.

For additional information on plans, see Section 6 of *DB2 Application Programming and SQL Guide*.

### 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, the privilege set of the process must include one of the following:

- Ownership of the plan
- BIND privilege on the plan
- BINDAGENT privilege granted by the plan owner
- SYSCTRL or SYSADM authority

### Syntax

```
              ┌─,─────────┐
►►──FREE PLAN──(──┬─▼─plan-name─┴─┬──)──┬─────────────────┬──────────────────────►◄
                  └─*─────────────┘      │        ┌─I─┐    │
                                         └─FLAG(──┼─W─┼──)─┘
                                                  ├─E─┤
                                                  └─C─┘
```

### 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 to see. 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 DSN8BC61 from DB2. Generate only warning, error, and completion messages (not informational messages).

```
FREE PLAN (DSN8BC61) FLAG (W)
```

## MODIFY irlmproc, ABEND (MVS IRLM)

The MODIFY *irlmproc*, ABEND command terminates the IRLM abnormally. IRLM processes this command even if there is a DB2 identified to it.

**Abbreviation**: F

## Environment

This command can be issued only from an MVS console.

**Data sharing scope**: Member

## Authorization

The command requires an appropriate level of MVS authority, as described in *OS/390 MVS System Commands.*

## Syntax

```
                          ┌─,DUMP───┐
►►──MODIFY──irlmproc,ABEND─┤         ├───────────────────────────────────────────►◄
                          └─,NODUMP─┘
```

## Option descriptions

Parameters must be separated by commas with no spaces.

*irlmproc*
: Identifies 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 de-register 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.

  A second invocation will cause IRLM to terminate abnormally with a U2020 abend; no dump will be taken.

## Usage notes

***Terminating IRLM:*** If there are any difficulties terminating IRLM, see “Usage note” on page 325

***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 on an MVS system console:

```
#   F KRLM001,ABEND
```

Response on the MVS system console:

```
#    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:

```
#   F KRLM001,ABEND,NODUMP
```

## MODIFY irlmproc, DIAG, DELAY (MVS IRLM)

The MODIFY *irlmproc*, DIAG, DELAY command initiates diagnostic dumps for IRLM subsystems in a data sharing group when responses to XES requests take longer than 45 seconds.

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, DELAY command under the direction of IBM Service if this situation occurs.

**Abbreviation**: F

## Environment

This command can be issued only from an MVS console.

**Data sharing scope**: Group

## Authorization

The command requires an appropriate level of MVS authority, as described in *OS/390 MVS System Commands.*

## Syntax

```
►►──MODIFY──irlmproc,DIAG,DELAY──────────────────────────────────────────────►◄
```

## Option descriptions

Parameters must be separated by commas with no spaces.

*irlmproc*
: Identifies the procedure name of the IRLM instance that is to be diagnosed.

**DIAG**
: Indicates 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.

## Usage note

# ***Restrictions:*** 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 reenter the command 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 procedure 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
```

## MODIFY irlmproc, SET (MVS IRLM)

The MODIFY irlmproc, SET command performs the following tasks:

- Dynamically sets the maximum CSA allowed for IRLM.
- Dynamically sets the number of trace buffers allowed for IRLM.

**Abbreviation**: F

## Environment

This command can be issued only from an MVS console.

**Data sharing scope**: Member

## Authorization

The command requires an appropriate level of MVS authority, as described in *OS/390 MVS System Commands.*

## Syntax

```
►►──MODIFY──irlmproc,SET──┬─,CSA=nnn──────────┬──────────────────────────────────────────►◄
                          │          ┌─10──┐  │
                          └─,TRACE=──┴─nnn─┴──┘
```

## Option descriptions

Use commas with no spaces to separate parameters.

*irlmproc*
Specifies the IRLM that is to process the command.

**SET**
# Sets the maximum amount of CSA storage or the number of trace buffers
# used for this IRLM.

# **CSA=***nnn*
# Requests that IRLM dynamically set the maximum amount of common
# storage area (CSA) for this IRLM to use for lock control structures. *nnn*
# must be a one- to three-digit number from 1 through 999. The number
# indicates what multiple of 1 MB of storage the specified IRLM will use.
# For example, CSA=6 allows the IRLM to use 6 MB of CSA.

# The lock control structures are allocated from extended common
# storage area (ECSA) when PC=NO, or from IRLM private region when
# PC=YES.

# IRLM does not immediately allocate CSA storage for the new value you
# set using this command; IRLM allocates storage as needed, not to
# exceed the amount of CSA specified. If the amount of storage currently
# allocated by IRLM is greater than the amount of CSA you specify using
# this command, IRLM does not obtain more storage until normal proc-

essing frees enough storage to bring the current allocation below the new CSA value you set.

**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 activated. 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 specified. If the number of trace buffers 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 or timeout cycle.

## Usage notes

***Determining limits for CSA values:*** Do not modify the CSA value without first contacting the system programmer to determine the amount of CSA storage that can be used for IRLM.

***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, and the value for MAXCSA returns to the value you set for the MAXCSA parameter on the IRLM startup procedure.

## Examples

***Example 1:*** Enter on an MVS system console:

```
F IR21PROC,SET,CSA=10
```

Response on the MVS system console:

```
DXR178I IR21033 MAXIMUM CSA IS SET TO 10MB
```

*Explanation:* IR21033 is the IRLM subsystem name concatenated with the IRLM system ID.

***Example 2:*** Enter on an MVS system console:

```
F IR21PROC,SET,TRACE=20
```

Response on the MVS system console:

```
DXR177I IR21033 THE MAXIMUM NUMBER OF TRACE BUFFERS
                FOR EACH TRACE TYPE IS SET TO 20
```

## MODIFY irlmproc, STATUS (MVS IRLM)

This command displays information for one or more subsystems connected to the specified IRLM designated using *irlmproc*. Each subsystem connected to the specified IRLM is listed, including subsystem name, status, work unit and lock information. 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 has been allocated since the last time this IRLM was started.

**Abbreviation**: F

## Environment

This command can be issued only from an MVS console.

**Data sharing scope**: Member or group, depending on which option you choose

## Authorization

The command requires an appropriate level of MVS authority, as described in *OS/390 MVS System Commands.*

## Syntax

```
                                ┌─,irlmx─┐
►►──MODIFY──irlmproc,STATUS─────┼────────┼──────────────────────────────────────────────►◄
                                ├─,ALLD──┤
                                ├─,ALLI──┤
                                ├─,STOR──┤
                                └─,TRACE─┘
```

## 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 DB2G2 (ID is 2).

**ALLD**
: Requests the DB2 subsystem name and status of a DB2 that is identified to an IRLM. In the data sharing group, this command lists information about all DB2s 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 DB2 identified.

- If there was a rebuild of the lock structure after the retained locks were created, it is the IRLM with the lowest member ID at the time the rebuild occurred.
- If a group restart is occurring and one DB2 is recovering on behalf of another DB2, the IRLM that is displayed is the one associated with the DB2 doing the peer recovery. For example, if DB2A is doing a peer recovery of DB2B, the display might show:

```
NAME  STATUS ... IRLM_NAME
DB2A  UP         IRLA
DB2B  DOWN       IRLA
```

**ALLI**

Requests the IRLM subsystem name, ID, status, and service level. In a data sharing group, this command lists information about all IRLMs 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 is the IRLM is connected by issuing a MODIFY irlmproc,STATUS command and checking that the output shows SCOPE=GLOBAL.

If an IRLM is down, it is displayed only if its associated DB2 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 DB2 identified.
- If there was a rebuild of the lock structure after the retained locks were created, it is the IRLM with the lowest member ID at the time the rebuild occurred.
- If the failed DB2 had recovery done on its behalf by another DB2, the IRLM that is displayed is the one associated with the DB2 that did the peer recovery.

**STOR**

For this IRLM only, displays the current and "high-water" allocation for CSA and ECSA 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 IRLMs.

## Examples

***Example 1:*** Enter on the MVS1 system console:

```
F IRTPROC,STATUS
```

Response on MVS1 system console:

```
DXR101I IR2T001 STATUS SCOPE=LOCAL
        SUBSYSTEMS IDENTIFIED                PT01
        NAME      STATUS    UNITS     HELD      WAITING     RET_LKS
        DSNT1     UP-NS     0005      0010      0002        0
```

*Explanation:* The operator on system 1 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
```

***Example 2:*** Assume you have a data sharing group. Enter on a system console:

```
F DB1GIRLM,STATUS,ALLD
```

Response on system console:

```
14.02.10 STC00086  DXR102I DJ1G001 STATUS
        SUBSYSTEMS IDENTIFIED                PT01
        NAME     STATUS    RET_LKS   IRLMID  IRLM_NAME
        DB4G     UP              0    004      DJ4G
        DB3G     UP              0    003      DJ3G
        DB2G     UP              0    002      DJ2G
        DB1G     UP              0    001      DJ1G
```

*Explanation:* The output shows all the DB2s that are connected to IRLMs in this data sharing group (the group to which the IRLM processing the request belongs). The value "UP" in the STATUS field indicates that the DB2 is active. Other possible values for STATUS include:

| | |
|---|---|
| DOWN | The DB2 is failed. All "modify" type locks held by this DB2 have been retained by IRLM. The DB2 is known to be down only if it has retained locks. |
| SYSFAIL | The IRLM that DB2 is identified to has been disconnected from the data sharing group. All "modify" type locks held by this DB2 have been retained by IRLM. The DB2 is known to be SYSFAIL only if it has retained locks. |

***Example 3:*** Again, assume data sharing is in effect. Enter the following on the system console:

```
F DB1GIRLM,STATUS,ALLI
```

The response on the console is:

```
17.17.03 STC00092  DXR103I LRLM007 STATUS
        IRLMS PARTICIPATING IN DATA SHARING GROUP FUNCTION LEVEL=006
        IRLM_NAME IRLMID STATUS LEVEL SERVICE MIN_LEVEL MIN_SERVICE
          JRLM      005    UP    013  PN92893    006      IRLM2.1
          KRLM      006    UP    006  IRLM2.1    006      IRLM2.1
          LRLM      007    UP    013  PN92893    006      IRLM2.1
```

*Explanation:* The output shows the IRLMs that are participating in this data sharing group (the group which includes the IRLM processing the request). Other information includes:

**STATUS** The value “UP” in the STATUS field indicates that the IRLM is active. STATUS shows “DOWN” if the IRLM is failed. An IRLM is known to be “DOWN” only if the DB2 that was identified to it has retained locks. This connection between a failed DB2 and IRLM is lost after a rebuild or a group restart.

**LEVEL** The current IRLM function level.

**SERVICE** The IRLM service or release that corresponds to the function level given in "LEVEL".

**MIN_LEVEL** The minimum IRLM function level this IRLM can coexist with.

**MIN_SERVICE** The IRLM service or release that corresponds to the function level given in "MIN-LEVEL".

**Group Function Level** The IRLM function level in use by all the IRLMs in the data sharing group.

***Example 4:*** Assume that this command is issued in a non-data sharing environment. Enter the following on the system console:

```
F DB1GIRLM,STATUS,ALLI
```

The response on the console is:

```
15.12.01 STC00092  DXR103I VRLM007 STATUS
        IRLMS PARTICIPATING IN DATA SHARING GROUP FUNCTION LEVEL=016
        IRLM_NAME IRLMID STATUS LEVEL SERVICE MIN_LEVEL MIN_SERVICE
          VRLM      007    UP    016  PQ15854    012      PN90337
```

*Explanation:* The output shows information only for the IRLM specified. The group function level shown is the function level for the specified IRLM. Refer to Example 3 on page 238 for additional information on interpreting output.

***Example 5:*** Enter the following command on the system console:

```
F IR21PROC,STATUS,STOR
```

The response on the console is:

```
#                   DXR100I IR21001 STOR STATS
#                           PC: NO    MAXCSA:    6M
#                   CSA USE: ACNT:    132K  AHWM:    132K  CUR:  4048K  HWM:  4086K
#                           ABOVE 16M:     72  4033K    BELOW 16M:      6     15K
#                   CLASS   TYPE  SEGS     MEM   TYPE  SEGS     MEM   TYPE  SEGS     MEM
#                   ACCNT   T-1      1     64K   T-2      1     64K   T-3      1      4K
#                   PROC    WRK     11     58K   SRB      3      3K   OTH      2      2K
#                   MISC    VAR     60   4081K   N-V      6     22K   FIX      1     24K
```

*Explanation:* The example shows that current storage allocated for IRLM is 4048 KB, and the greatest amount that has been allocated since the last time IRLM was started is 4086 KB. The storage for the locking structures (RHB and RLB) is contained within ECSA, because this IRLM is defined with PC=NO. Use the following information to interpret the display output:

**PC**
Displays the current value for the PC option of the IRLM startup procedure.

**MAXCSA**
Displays the current value for the MAXCSA option of the IRLM startup procedure. The MAXCSA value is 6 MB in this example.

**CSA USE**
Shows storage use that is accountable toward the MAXCSA value of the IRLM procedure. In this output, the current use accountable storage (ACNT) is 132 KB. The high water mark since the last time IRLM was started (AHWM) is also 132 KB.

**CUR**
Shows the total current CSA and ECSA usage. In this case, the current usage (CUR) is 4048 KB, and the high water mark (HWM) is 4086 KB. The accountable storage is a subset of this total storage.

**ACCNT**
The ACCNT row of the report is a breakdown of lock control block structures and their storage use.

**T-1** Type one structures are for resources. In this case, it shows that one storage segment is held for a total of 64 KB.

**T-2** Type two structures are for all resource requests after the first request for a specific resource. This example shows that one storage segment is held for a total of 64 KB.

**T-3** Type three structures are for requesters or work units that are waiting for or hold resources. This example shows that one storage segment is held for a total of 4 KB.

**PROC and MISC rows**
These rows contain usage information for CSA, ECSA, and private storage used to process DBMS requests. Use this information under the guidance of IBM® Support Center for diagnosing problems.

For more information, see the explanation of message DXR100I in *DB2 Messages and Codes*.

***Example 6:*** Assume the IRLM was started with PC=YES. Enter the following command on the system console:

```
F IR21PROC,STATUS,STOR
```

The response on the console is:

```
#     DXR100I JR21001 STOR STATS
#             PC: YES   MAXCSA:    N/A
#     CSA USE: ACNT:      OK  AHWM:      OK  CUR:  4362K  HWM:  5830K
#             ABOVE 16M:     78  4376K    BELOW 16M:     23     32K
#     CLASS   TYPE  SEGS     MEM   TYPE  SEGS     MEM   TYPE  SEGS     MEM
#     ACCNT   T-1      1     64K   T-2      1     64K   T-3      1      4K
#     PROC    WRK     11     58K   SRB     20     20K   OTH      2      2K
#     MISC    VAR     68   4497K   N-V      6     22K   FIX      1     24K
```

*Explanation:* This example was created using a poorly-tuned application, and shows how important a well-tuned system is for predicting system storage needs. This example illustrates what can happen when an application generates a high IRLM lock contention rate; the high value of 20 segments with 20 KB each for type SRB storage, and the high value of 23 segments with 23 KB each for storage below the 16 MB line are some of the results.

For more information about reducing lock contention, see Section 5 (Volume 2) of *DB2 Administration Guide*. For more information about tuning your system, see Chapter 7 of *DB2 Data Sharing: Planning and Administration*.

***Example 7:*** Enter the following command on the system console:

```
F PR21PROC,STATUS,TRACE
```

The response on the console is:

```
    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. For more information about the trace types, see “TRACE CT (MVS IRLM)” on page 341.

Use the TRACE CT command of MVS on page 341 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 on page 234to change the maximum number of trace buffers.

## -MODIFY TRACE (DB2)

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 an MVS 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, the privilege set of the process must include one of the following:

- TRACE privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
                                             ┌─*──────────────┐
                                             │ ┌─,───────┐    │
►►──MODIFY TRACE──(──┬─PERFM───┬──)──CLASS(──┴─▼─integer─┴────┴──)──TNO(integer)──────────────────►
                     ├─ACCTG───┤
                     ├─STAT────┤
                     ├─AUDIT───┤
                     └─MONITOR─┘

            ┌─*───────────────┐
            │ ┌─,─────────┐   │
►──IFCID(───┴─▼─ifcid_nbr─┴───┴──)──┬──────────────────┬───────────────────────────────────────►◄
                                    └─COMMENT(string)──┘
```

## Option descriptions

**TRACE**

Determines which IFCIDs are started. For more descriptions of each trace type, see "-START TRACE (DB2)" on page 300.

*Table 23. Trace types*

| Type | Description | Abbreviation |
|---|---|---|
| PERFM | Performance records of specific events | P |
| ACCTG | Accounting records for each transaction | A |
| STAT | Statistical data | S |
| AUDIT | Audit data | AU |
| MONITOR | Monitor data | MON |

One additional trace type is not described here. It is intended for service and is to be used under the direction of IBM support personnel. For details, see *DB2 Diagnosis Guide and Reference*.

**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. For descriptions of the allowable classes, see "-START TRACE (DB2)" on page 300.

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 only specify 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. (See Example on page 244 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 1 through 350, with the exception of: 4, 5, 185, 187, 217, 232, 234, 240, and 241.

The **default** is **IFCID(*).**

**COMMENT(***string***)**

Gives 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

Change trace number 6 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')
```

## 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
- You previously ran BIND PACKAGE with the SQLERROR(CONTINUE) option

For more information on using REBIND PACKAGE, see Section 6 of *DB2 Application Programming and SQL Guide*.

## 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.

Table 24 explains the authorization required to run REBIND PACKAGE, depending on the options specified.

*Table 24 (Page 1 of 2). 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. | The authorization IDs of the process must have one of the following:<br>• Ownership of the package<br>• BIND privilege on the package<br>• BINDAGENT privilege from the owner of the package<br>• PACKADM authority on the collection or on all collections<br>• SYSADM or SYSCTRL authority |
| REBIND PACKAGE with no change in ownership, although the original owner is specified for the OWNER keyword. | The authorization IDs of the process must have one of the following:<br>• OWNER *authorization-id* must be one of the primary or secondary authorization IDs of the binder<br>• BINDAGENT privilege from the owner of the package<br>• SYSADM or SYSCTRL authority |

*Table 24 (Page 2 of 2). Summary of privileges for REBIND PACKAGE*

| Option | Authorization required to run REBIND PACKAGE |
|---|---|
| REBIND PACKAGE with change of ownership. (An authorization ID that is not the original owner is specified in the OWNER keyword.) | The new OWNER must have one of the following: <br>• BIND privilege on the package <br>• PACKADM authority on the collection or on all collections <br>• SYSADM or SYSCTRL authority <br><br>**Specifying the OWNER:** If any of the authorization IDs have the BINDAGENT privilege granted from the owner, then *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. |

For additional information on the authorization required to execute BIND PLAN, see Section 5 (Volume 2) of *DB2 Administration Guide*.

## Syntax

```
  ►►──REBIND PACKAGE──────────────────────────────────────────────────────────────────────────►

        ┌─,────────────────────────────────────────────────────────────┐
  ►──(──┬─▼─┬────────────────┬──┬─collection-id─┬─.─┬─package-id─┬──┬─────────────────────┬─┴──┬─)────►
        │   └─location-name.─┘  └─*─────────────┘   └─*──────────┘  └─.(─┬──────────────┬─)─┘    │
        │                                                                ├─version-id─┤        │
        │                                                                └─*──────────┘        │
        └─*────────────────────────────────────────────────────────────────────────────────────┘

  ►──┬───────────────────────────┬──┬──────────────────────────┬──┬──────────────────────────┬──►
     └─OWNER(authorization-id)─┘  └─QUALIFIER(qualifier-name)─┘  └─CURRENTDATA(─┬─YES─┬─)─┘
                                                                              └─NO──┘

| ►──┬───────────────────────────────┬──┬─────────────────────┬──┬──────────────────┬──────────►
|    └─DBPROTOCOL──(─┬─DRDA────┬─)─┘  ├─DEFER(PREPARE)────┤  └─DEGREE(─┬─1───┬─)─┘
                     └─PRIVATE─┘      └─NODEFER(PREPARE)──┘            └─ANY─┘

  ►──┬────────────────────────────────┬────────────────────────────────────────────────────────►
     └─DYNAMICRULES(─┬─RUN────┬─)─┘
|                    ├─BIND───┤
|                    ├─DEFINE─┤
                     └─INVOKE─┘

  ►──┬──────────────────────────────────────────────────────────────────────────────────────┬──►
     │                          ┌─,─────────┐          ┌──────────────────────────────────┐   │
     └─┬─┬─ENABLE──┬──(─▼─BATCH─────┴──)─┬──▼─┬────────────────────────────────────┬─┴───┘
       │ └─DISABLE─┘       ├─DLIBATCH──┤     │    │               ┌─,───────────────┐   │
       │                   ├─DB2CALL───┤     │    ├─DLIBATCH(─▼─connection-name─┴─)─┤
       │                   ├─CICS──────┤     │    │        ┌─,──────┐                   │
       │                   ├─IMS───────┤     │    ├─CICS(─▼─applid─┴─)────────────────┤
       │                   ├─IMSBMP────┤     │    │          ┌─,─────┐                  │
       │                   ├─IMSMPP────┤     │    ├─IMSBMP(─▼─imsid─┴─)───────────────┤
       │                   ├─REMOTE────┤     │    │          ┌─,─────┐                  │
       │                   └─RRSAF─────┘     │    ├─IMSMPP(─▼─imsid─┴─)───────────────┤
       └─ENABLE(*)───────────────────────┘    │          ┌─,──────────────────┐      │
                                              └─REMOTE(─▼─┬─location-name─┬─┴─)──┘
                                                          └─<luname>──────┘

  ►──┬─────────────────────────┬──┬────────────────────┬──┬──────────────────────────┬──┬───────────────────────┬──►
#    └─EXPLAIN(─┬─YES─┬─)─┘  │          ┌─I─┐          │  └─IMMEDWRITE(─┬─NO──┬─)─┘  └─ISOLATION(─┬─RR─┬─)─┘
#               └─NO──┘      └─FLAG(─┼─W─┼─)──┘                 ├─PH1─┤                     ├─RS─┤
#                                     ├─E─┤                        └─YES─┘                     ├─CS─┤
                                      └─C─┘                                                    ├─UR─┤
                                                                                               └─NC─┘

                                         ┌─NOREOPT(VARS)─┐
| ►──┬──────────────────────────────┬──┼─────────────────┼──┬──────────────────────────┬──────►
     │               ┌─NO──┐        │  └─REOPT(VARS)─────┘  └─OPTHINT──(──'hint-id'──)─┘
     └─KEEPDYNAMIC(─┴─YES─┴─)──┘

  ►──┬─────────────────────────────────┬──┬───────────────────┬──┬──────────────────────────────┬──►
|    │       ┌─,───────────────┐       │  └─PATHDEFAULT─(1)─┘  └─RELEASE(─┬─COMMIT─────┬─)─┘
|    └─PATH(─▼─┬─schema-name─┬─┴─)──┘                                     └─DEALLOCATE─┘
               └─USER────────┘

  ►──┬──────────────────────────┬──────────────────────────────────────────────────────────────►
     └─VALIDATE(─┬─RUN──┬─)─┘
                 └─BIND─┘
```

**Note:**

1 The **PATHDEFAULT** keyword is mutually exclusive with the **PATH** keyword. Do not specify both keywords in the same REBIND command.

## Option descriptions

For descriptions of the options shown in the syntax diagram, see “Options of BIND and REBIND for PLAN, PACKAGE, and TRIGGER PACKAGE” on page 57.

## Usage note

If you rebind multiple packages, DB2 commits each successful rebind before rebinding the next package.

## Example

Rebind packages TEST.DSN8BC61.(MAY_VERSION) and PRODUCTION.DSN8BC61.(DEC_VERSION), both 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 specified on the BIND PACKAGE command.

```
REBIND PACKAGE (USIBMSTODB22.TEST.DSN8BC61.(MAY_VERSION),
                USIBMSTODB22.PRODUCTION.DSN8BC61.(DEC_VERSION)) -
  ENABLE (CICS,DLIBATCH) CICS (CON2)
```

## 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, create a new index 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).

For more information on using REBIND PLAN, see Section 6 of *DB2 Application Programming and SQL Guide*.

## 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 verifies the authorization initially 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.

Table 25 explains the authorization required to run REBIND PLAN, depending on the options specified.

*Table 25 (Page 1 of 2). 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:<br>• Ownership of the plan<br>• BIND privilege on the plan<br>• BINDAGENT privilege from the owner of the plan<br>• SYSADM or SYSCTRL authority |
| REBIND PLAN with no change in ownership, although the original owner is specified for the OWNER keyword. | The authorization IDs of the process must have one of the following:<br>• OWNER *authorization-id* must be one of the primary or secondary authorization IDs of the binder<br>• BINDAGENT privilege from the owner of the plan<br>• SYSADM or SYSCTRL authority |

*Table 25 (Page 2 of 2). Summary of privileges for REBIND PLAN*

| **Option** | **Authorization required to run REBIND PLAN** |
|---|---|
| REBIND PLAN with change of ownership. (An authorization ID that is not the original owner is specified in the OWNER keyword.) | The new OWNER must have one of the following:<br>• BIND privilege on the plan<br>• SYSADM or SYSCTRL authority<br>**Specifying the OWNER:** If any of the authorization IDs has the BINDAGENT privilege granted from the owner, then *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. |
| PKLIST, specifying individual packages | Authorization ID of the process must include one of the following:<br>• EXECUTE privilege on each package specified in the PKLIST<br>• PACKADM authority on specific collections containing packages or on collection *<br>• SYSADM authority |
| PKLIST, specifying (*), indicating all packages in the collection | Authorization ID of the process must include one of the following:<br>• EXECUTE privilege on each package in the collection<br>• EXECUTE privilege on *collection-id*.*<br>• PACKADM authority on *collection-id* or on *<br>• SYSADM authority |

For additional information on the authorization required to execute REBIND PLAN, see Section 5 (Volume 2) of *DB2 Administration Guide*.

## Syntax

```
 ►►──REBIND PLAN──(─┬─┌─,─────────┐─┬─)──┬────────────────────────────┬──┬──────────────────────────────┬──►
                    │ └▼plan-name─┘ │     └─OWNER(authorization-id)─┘  └─QUALIFIER(qualifier-name)─┘
                    └─*─────────────┘

 ►──┬────────────────────────────────────────────────────────────────────────┬──┬──────────────────┬────►
    │              ┌─,──────────────────────────────────────────────────┐    │  ├─NODEFER(PREPARE)─┤
    ├─PKLIST(──────▼─┬──────────────────┬──┬─collection-id─┬──┬─.package-id─┬─┴─)─┤  └─DEFER(PREPARE)───┘
    │                ├─location-name.───┤  └─*─────────────┘  └─.*──────────┘    │
    │                └─*.───────────────┘                                        │
    └─NOPKLIST───────────────────────────────────────────────────────────────────┘

 ►──┬──────────────────────────┬──┬──────────────────────────┬──┬───────────────────────────┬──────────►
    └─ACQUIRE(──┬─USE──────┬─)─┘  └─CACHESIZE(decimal-value)─┘  └─CURRENTDATA(──┬─NO──┬─)─┘
                └─ALLOCATE─┘                                                   └─YES─┘

|►──┬─────────────────────────────┬──┬─────────────────────────────┬──┬───────────────────────┬─────────►
|   └─CURRENTSERVER(location-name)┘  └─DBPROTOCOL──(──┬─DRDA────┬─)┘  └─DEGREE(──┬─1───┬─)─┘
                                                     └─PRIVATE─┘               └─ANY─┘

 ►──┬──────────────────────────────────┬──┬──────────────────────────────┬─────────────────────────────►
    └─DISCONNECT(──┬─EXPLICIT────┬─)───┘  └─DYNAMICRULES(──┬─RUN──┬─)─┘
                   ├─AUTOMATIC───┤                        └─BIND─┘
                   └─CONDITIONAL─┘

 ►──┬──────────────────────────────────────────────────────────────────────────────────────────┬───────►
    │                     ┌─,─────────┐            ┌──────────────────────────────────────────┐ │
    └──┬─┬─ENABLE──┬──(───▼─┬─BATCH────┬─┴──)──┬──▼─┬─DLIBATCH(──┌─,───────────────┐─)─┬─┴─┘
       │ └─DISABLE─┘        ├─DLIBATCH─┤       │    │            └▼connection-name─┘    │
       │                    ├─DB2CALL──┤       │    ├─CICS(──┌─,──────┐─)──────────────┤
       │                    ├─CICS─────┤       │    │        └▼applid─┘                 │
       │                    ├─IMS──────┤       │    ├─IMSBMP(──┌─,─────┐─)────────────┤
       │                    ├─IMSBMP───┤       │    │          └▼imsid─┘               │
       │                    ├─IMSMPP───┤       │    └─IMSMPP(──┌─,─────┐─)────────────┘
       │                    └─RRSAF────┘       │               └▼imsid─┘
       └─ENABLE(*)─────────────────────────────┘

 ►──┬─────────────────────┬──┬────────────────────┬──┬────────────────────────┬──┬───────────────────────┬──►
#   └─EXPLAIN(──┬─NO──┬─)─┘  │          ┌─I─┐     │  └─IMMEDWRITE(──┬─NO──┬─)─┘  └─ISOLATION(──┬─RR─┬─)─┘
#               └─YES─┘      └─FLAG(──┬─┼─W─┼─)───┘                 ├─PH1─┤                  ├─RS─┤
#                                     ├─E─┤                         └─YES─┘                  ├─CS─┤
                                      └─C─┘                                                  └─UR─┘

                                        ┌─NOREOPT(VARS)─┐
 ►──┬───────────────────────────────┬──┼───────────────┼──┬────────────────────────────┬────────────────►
|   │                 ┌─NO──┐       │  └─REOPT(VARS)───┘  └─OPTHINT──(──'hint-id'──)─┘
    └─KEEPDYNAMIC(────┴─YES─┴───)───┘

 ►──┬──────────────────────────────┬──┬────────────────────┬──┬─────────────────────────────────┬────────►
|   │      ┌─,──────────────┐      │  └─PATHDEFAULT─(1)────┘  └─RELEASE(──┬─COMMIT─────┬─)──────┘
|   └─PATH(─▼─┬─schema-name─┬─┴──)──┘                                     └─DEALLOCATE─┘
|             └─USER────────┘

 ►──┬─────────────────────────┬──┬─────────────────────────┬───────────────────────────────────────────►
    └─SQLRULES(──┬─DB2─┬──)───┘  └─VALIDATE(──┬─RUN──┬─)───┘
                 └─STD─┘                      └─BIND─┘
```

**Note:**

[1] The **PATHDEFAULT** keyword is mutually exclusive with the **PATH** keyword. Do not specify both keywords in the same REBIND command.

## Option descriptions

For descriptions of the options shown in the syntax diagram, see “Options of BIND and REBIND for PLAN, PACKAGE, and TRIGGER PACKAGE” on page 57.

## Usage note

If you rebind multiple plans, DB2 commits each successful rebind before rebinding the next plan.

## Example

Rebind plan DSN8BC61 to enable DB2 to take advantage of a newly created index. Use FLAG(W) to issue warning, error, and completion messages, but not informational messages. Use VALIDATE(BIND) to point out any error conditions during the bind process. Use 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. Omit the OWNER keyword to leave the plan's owner authorization ID the same. Omit the ENABLE or DISABLE keywords to use the connections previously defined for the plan.

```
REBIND PLAN (DSN8BC61) -
  FLAG (W) -
  VALIDATE (BIND) -
  ISOLATION (CS)
```

## 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.

For more information on using REBIND TRIGGER PACKAGE, see Section 6 of *DB2 Application Programming and SQL Guide*.

## 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, the privilege set of the process must include one of the following 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
- SYSADM or SYSCTRL 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, one of the following authorizations is needed:

- Ownership of the package
- BIND privilege on the package
- BINDAGENT privilege granted from owner
- PACKADM, SYSADM, or SYSCTRL

For additional information on the authorization required to execute REBIND TRIGGER PACKAGE, see Section 5 (Volume 2) of *DB2 Administration Guide*.

## Syntax

```
►►──REBIND TRIGGER PACKAGE──(──┬────────────────┬──┬─collection-id─┬─.─┬─package-id─┬─)──────────►
                               └─location-name.─┘  └─*─────────────┘   └─*──────────┘

►──┬──────────────────────────┬──┬───────────────────────┬──┬─────────────────┬──────────────────►
   └─CURRENTDATA(──┬─YES─┬─)──┘  └─EXPLAIN(──┬─YES─┬─)───┘  │        ┌─I─┐    │
                   └─NO──┘                   └─NO──┘         └─FLAG(──┼─W─┼─)─┘
                                                                      ├─E─┤
                                                                      └─C─┘

►──┬─────────────────────────┬──┬─────────────────────┬──┬───────────────────────────────┬────────►
   └─IMMEDWRITE(──┬─NO──┬─)──┘  └─ISOLATION(──┬─RR─┬─)─┘  └─RELEASE(──┬─COMMIT─────┬─)───┘
                  ├─PH1─┤                     ├─RS─┤                  └─DEALLOCATE─┘
                  └─YES─┘                     ├─CS─┤
                                              ├─UR─┤
                                              └─NC─┘
```

## 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 shown in the syntax diagram, see “Options of BIND and REBIND for PLAN, PACKAGE, and TRIGGER PACKAGE” on page 57.

For more information about specifying schema names and trigger packages for the REBIND TRIGGER PACKAGE command, see Section 5 of *DB2 Application Programming and SQL Guide*.

## 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.

## Output

REBIND TRIGGER PACKAGE updates the COLLID and NAME columns in the SYSPACKAGE catalog table.

## Example

Issue the following command to rebind trigger package TRIG1 in the ADMF001 collection of packages:

```
REBIND TRIGGER PACKAGE (ADMF001.TRIG1);
```

This command produces output 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
           NOREOPT       VARS
           KEEPDYNAMIC   NO
           DBPROTOCOL    DRDA
           QUERYOPT      1
           PATH
"SYSIBM","SYSFUN","SYSPROC","SYSADM","ADMF001"
DSNT232I - SUCCESSFUL REBIND FOR
           PACKAGE = STLEC1.ADMF001.TRIG1.()
```

## -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 DB2 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 good BSDS in the newly allocated data set and to reinstate dual BSDS mode.

**Abbreviation**: -REC BSDS

### Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following:

- BSDS privilege
- SYSCTRL or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
►►──RECOVER BSDS──────────────────────────────────────────────────────────►◄
```

### Usage note

***-RECOVER BSDS following a BSDS I/O error:*** For a detailed description of steps the installation must take to reestablish dual BSDS mode after a BSDS I/O error occurs, see Section 4 (Volume 1) of *DB2 Administration Guide*.

### Example

Reestablish dual BSDS mode.

```
-RECOVER BSDS
```

## -RECOVER INDOUBT (DB2)

The DB2 command RECOVER INDOUBT recovers threads left indoubt 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. It is also critical that the commit/abort decision (that is made by the coordinator) must be determined.

**Abbreviation**: -REC IND

## Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following:

- RECOVER privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
                                                                        ┌─,──────────────┐
►►──RECOVER INDOUBT──┬───────────────────┬──ACTION(──┬─COMMIT─┬──)──┬──ID(──┬─▼─correlation-id─┴─┬──)──┬──►◄
                     └─(connection-name)─┘           └─ABORT──┘     │      └─*──────────────────┘     │
                                                                    │        ┌─,──────────┐           │
                                                                    ├──NID(──▼─network-id─┴──)─────────┤
                                                                    │          ┌─,─────────┐          │
                                                                    └──LUWID(──▼─┬─luwid─┬─┴──)────────┘
                                                                                 └─token─┘
```

## Option descriptions

**(***connection-name***)**

# Is a one- to eight-character connection name. Allied threads (including those that are distributed) belonging to that 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 an MVS console, and you are recovering an allied thread using the ID or NID parameter, you *must* supply a connection name; no default is available.

**ACTION**
Tells whether to commit or cancel the indoubt thread. If there are any downstream participants for which the local thread is the coordinator, then the commit or abort decision is propagated to these participants.

**Abbreviation**: ACT

**(COMMIT)**
Commits the thread.

**(ABORT)**
Cancels the thread.

**ID(***correlation-id***, ...)**
Tells whether to recover a specific allied thread or all allied threads (including those that are distributed) associated with the connection name.

*correlation-id*
Is the correlation ID (of 1 to 12 characters) of a specific thread to be recovered. If you use more than one correlation ID, separate items in the list by commas.

Do not use a correlation ID that has more than one network ID associated with it. Instead, use the NID option.

**(*)** Recovers all indoubt threads associated with the connection name. Even threads having the same correlation ID are resolved.

**NID(***network-id***, ...)**
Identifies threads by their network IDs.

*network-id* is a network ID 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 the name of the form *net-node.number*, from 3 to 25 characters in length.

- *net-node* is the network node name of the system that originated the unit of work. It uses from 1 to 8 characters.
- *number* is a unique number within the system of origin. It uses from 1 to 16 characters.

For RRSAF connections, a network ID is the OS/390 RRS unit of recovery ID (URID) that is used to uniquely identify a unit of work. An OS/390 RRS URID is a 32-character number.

The network ID appears on the recovery log of the commit coordinator as a unique identification of a unit of work. It uses 16 bytes.

- 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 hex characters. The last element of the LUWID is the commit sequence number of 4 hex characters, preceded by a period.

*token*

A token is an alternate way to express an LUWID. DB2 assigns a token to
# each thread 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:*** *network-id* is not normally needed, because *correlation-id* can identify indoubt threads. However, if *correlation-id* is not unique, *network-id* must be used. This statement does not apply if a LUWID is specified.

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. See the *DB2 Administration Guide* for an explanation of global transactions.

## Examples

***Example 1:*** Recover indoubt allied threads. Schedule a commit for all threads 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 has an LUWID = DB2NET.LUNSITE0.A11A7D7B2057.0002. (The '0002' in the last segment of the LUWID represents the commit sequence number.) The second has a token = 442.

```
-RECOVER INDOUBT ACTION(ABORT)
        LUWID (DB2NET.LUNSITE0.A11A7D7B2057.0002, 442)
```

## -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 was not selected.

**Abbreviation**: -REC POST

### Environment

This command can be issued from an MVS console, a DSN session, a DB2I panel (DB2 COMMANDS), or an IMS or CICS terminal.

**Data sharing scope**: Member

### Authorization

To execute this command, the privilege set of the process must include one of the following authorities:

- RECOVER privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
►►──RECOVER POSTPONED──────────────────────────────────────────────────────►◄
```

### 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 DSNI024I indicates the completion of back-out work against the page set or partition, and the removal of the page set or partition from the restart-pending status.

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

Issue the following command to recover postponed-abort units of recovery.

```
-RECOVER POSTPONED
```

If postponed-abort units of recovery are found, output similar to the following output is generated:

```
DSNV435I - RESOLUTION OF POSTPONED ABORT URS HAS BEEN SCHEDULED
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
        FOR PAGESET DBKD0103.IPKD013A PART 00000004.
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
        FOR PAGESET DBKD0103.TPKD0103 PART 00000004.
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
        FOR PAGESET DBKD0103.IXKD013C PART (n/a).
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
        FOR PAGESET DBKD0103.IUKD013B PART (n/a).
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
        FOR PAGESET DBKD0103.IPKD013A PART 00000002.
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
        FOR PAGESET DBKD0103.TPKD0103 PART 00000002.
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
        FOR PAGESET DBKD0101.IXKD011C PART (n/a).
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
        FOR PAGESET DBKD0101.IXKD011B PART (n/a).
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
        FOR PAGESET DBKD0101.IUKD011A PART (n/a).
DSNI024I - DSNIARPL BACKOUT PROCESSING HAS COMPLETED
        FOR PAGESET DBKD0101.TLKD0101 PART (n/a).
DSN9022I - DSNVRP 'RECOVER POSTPONED' NORMAL COMPLETION
```

If no postponed units of recovery are found, the following output is returned:

```
DSNV434I - DSNVRP NO POSTPONED ABORT THREADS FOUND
DSN9022I - DSNVRP 'RECOVER POSTPONED' NORMAL COMPLETION
```

## -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 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 an MVS 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, the privilege set of the process must include one of the following:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
                       ┌─,─────────────┐
►►──RESET GENERICLU──┬─(─▼─┬─luname───────┬─┴─)─┬─────────────────────────────────►◄
                     │     └─netid.luname─┘     │
                     └─(*)──────────────────────┘
```

## Option descriptions

**(*luname*)**
# Specifies the real VTAM LU name of the partner whose generic LU name
# mapping is being purged. The NETID of this partner LU must be the same as
# the local DB2 NETID.

**(*netid.luname*)**
# Indicates the VTAM shared memory information 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 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 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 from the data sharing group.

```
-DB2A RESET GENERICLU(*)
```

## -RESET INDOUBT (DB2)

The DB2 command RESET INDOUBT purges information displayed in the indoubt thread report 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 an MVS 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, the privilege set of the process must include one of the following:

- RECOVER privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
                                ┌─,──────┐
►►──RESET INDOUBT──┬─LUNAME(─┬──▼─luname─┴─┬─)─┬───────┬─┬──────────────────────────►◄
                   │         └─*───────────┘   └─FORCE─┘ │
                   │            ┌─,─────────────┐        │
                   ├─LOCATION(──▼─location-name─┴─)──────┤
                   │            ┌─,──────────────────────┐          │
                   ├─IPADDR(─┬──▼─nnn.nnn.nnn.nnn:port─┴─┬─)─┬───────┬─┤
                   │         └─*─────────────────────────┘   └─FORCE─┘ │
                   │         ┌─,───────┐                                │
                   └─LUWID(──▼─┬─luwid─┴─┬─)─┬───────────────────────┬─┘
                               └─token─┘     └─LOCATION(location-name)─┘
```

## Option descriptions

**LUNAME(***luname***, ...)**
Purges all qualifying indoubt information that pertains to the named LUNAME.

*luname*
Is expressed as a one- to eight-character name. If you use more than one LUNAME, separate items in the list by commas.

**(*)** 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 execution of 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 executed, execution fails with message DSNL448I.

FORCE can be used to bypass warm start connectivity problems when errors occurring in the recovery log name exchange result in the partner refusing the connection attempt.

**LOCATION(***location-name***, ...)**
Purges all qualifying indoubt information pertaining to the named location.

*location-name* is expressed as a 1- to 16-character name, and identifies the partner, whether it is a requester or server. If the partner is not a DB2 for OS/390 subsystem, the location name may be expressed as a:

- One- to eight-character luname, as defined to VTAM at the server location. This name must be enclosed with the less than (<) and the greater than (>) characters to distinguish it from a DB2 location name.
- Dotted decimal TCP/IP address.

**IPADDR(***nnn.nnn.nnn.nnn:port***)**
Purges all qualifying indoubt information pertaining to the dotted decimal 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.

*nnn.nnn.nnn.nnn:port*
Is the dotted decimal IP address of the remote site followed by the resync port number. If you use more than one IP address and port, use commas to separate the items in the list.

**(*)** Purges indoubt information for all TCP/IP locations.

**LUWID**
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 hex characters. The last element of the LUWID is the commit sequence number, of 4 hex characters, preceded by a period.

*token*
A token is an alternate way to express an LUWID. DB2 assigns a token to
# each thread 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 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:*** Be cautious when using 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:*** Be cautious when using 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.

## 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 using the DB2I RUN panel.

**Data sharing scope**: Member

## Authorization

To execute this command, the privilege set of the process must include one of the following:

- 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

```
►►──RUN──┬─PROGRAM(program-name)──┬─────────────────┬─┬──┬──────────────────────┬─────────────►
         │                        └─PLAN(plan-name)─┘ │  └─LIBRARY(library-name)─┘
         └─CP──PLAN(plan-name)────────────────────────┘

►──┬─────────────────────────┬────────────────────────────────────────────────────────────────►◄
   └─PARMS(parameter-string)─┘
```

## 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 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 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 PROGRAM, but required after CP.

*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 containing the program to be run.

If *library-name* is not specified, normal MVS library searching is used. The data sets specified in the STEPLIB DD statements are first searched for the entry point name of the program. If STEPLIB is not present, then the data sets specified in the JOBLIB DD statements are searched. If the entry point name is not found there, then the link list is searched.

**Subprograms:** Normal MVS library searching is **always** used for any subprograms loaded by the main program. If the subprograms reside in the same library as the main program, then library-name must also be defined for the normal MVS search pattern (STEPLIB, JOBLIB, link list). If a library defined in that way contains both the main program and any loaded subprograms, then you need not 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 by commas, blanks, or both, and enclose the list between apostrophes. If the list contains apostrophes, represent each of them by two consecutive apostrophes. The list is passed as a varying-length character string of a minimum of 1 to a maximum of 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* 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 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 "/." System message IBM003I indicates that either you have omitted the slash, or the value passed to the PL/I run-time package was not valid.

**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 "/."

## 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 DSN8BE61. 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 (DSN8BE61) -
    LIB ('prefix.RUNLIB.LOAD') PARMS ('/O'TOOLE')
```

## -SET ARCHIVE (DB2)

The DB2 command SET ARCHIVE sets the upper limit for the number of and the deallocation time of tape units for the archive log. This command overrules the values 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 set during installation.

**Abbreviation**: -SET ARC

### Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following:

- ARCHIVE privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
►►──SET ARCHIVE──┬─┬──────────────────────┬──┬────────────────────────────────────┬─┬──────────────────►◄
                 │ └─COUNT──(──integer──)─┘  └─TIME(─┬─minutes─┬────────────┬─┬─)─┘ │
                 │                                   │         └─,seconds───┘ │      │
                 │                                   ├─,seconds───────────────┤      │
                 │                                   ├─1440───────────────────┤      │
                 │                                   └─NOLIMIT────────────────┘      │
                 └─DEFAULT───────────────────────────────────────────────────────────┘
```

### Option descriptions

The following options overrule the READ TAPE UNITS(COUNT) and DEALLC PERIOD TIME subsystem parameters specified at installation.

**COUNT(*integer*)**
Specifies the maximum number of tape units that can be dedicated to reading archive logs. This value affects the allowed concurrent reads for unique archive data sets residing on tapes.

*integer* can range from 1 to 99.

- If the number specified is greater than the current specification, the maximum number of tape units allowable for reading archive logs increases.
- If the number specified 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; a tape unit is a candidate for deallocation because of a lowered COUNT value only if there is no activity for the unit.

**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 speci-
# fied during DB2 installation.

## Usage notes

***Archive tape reading performance:*** To achieve the best performance for reading archive tapes, specify the maximum values allowed (within system constraints) for both the COUNT and TIME options.

***IEF238D "REPLY DEVICE NAME OR CANCEL":*** 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 being deallocated, the archive tape is not available to any other members of the group until the specified time has expired. 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)
```

## -SET LOG (DB2)

The DB2 command SET LOG modifies the checkpoint frequency specified during installation. This command also overrules 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 again uses the values that were set during installation. The new LOGLOAD value takes affect following the next system checkpoint.

## Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following authorities:

- ARCHIVE privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands that are issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
►►──SET LOG──┬─LOGLOAD(integer)─┬──────────────────────────────────────────────►◄
             ├─SUSPEND──────────┤
             └─RESUME───────────┘
```

## Option descriptions

The following option overrules the LOGLOAD subsystem parameter that is specified in the CHECKPOINT FREQ field on panel DSNTIPN at installation.

**LOGLOAD(*integer*)**
: Specifies the number of log records that DB2 writes between the start of successive checkpoints. You can optionally specify a value of 0 to initiate a system checkpoint without modifying the current LOGLOAD value.

  *integer* can be 0, or within the range from 200 to 16000000.

**SUSPEND**
: Specifies 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, then suspends the update activity. Message DSNJ372I is issued and remains on the console until update activity resumes.

  This option is not allowed when a system quiesce is active by either the ARCHIVE LOG or STOP DB2 commands. Update activity remains suspended until SET LOG RESUME or STOP DB2 is issued.

**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**
Specifies to resume logging and update activity for the current DB2 subsystem and remove the message DSNJ372I from the console.

## Usage notes

***How LOGLOAD values affect DB2 performance:*** The LOGLOAD value can affect the amount of time needed to restart DB2 after abnormal termination. A large value can result in lengthy restart times. A low value can result in excessive check-pointing. When you specify LOGLOAD(0), the checkpoint request is synchronous when issued from a batch job, and asynchronous when issued from an MVS or TSO console.

Use the DISPLAY LOG command to display the current LOGLOAD setting.

***When to suspend logging:*** Specify SET LOG SUSPEND when you take a remote copy of the entire database. You can make remote copies with functions such as 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.

## 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
```

## 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 effect of the SPUFI subcommand is to execute SPUFI and to present the SPUFI panel as the start of a SPUFI session. For a description of the panel and instructions on using SPUFI, see Section 2 of *DB2 Application Programming and SQL Guide*.

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.

## /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, as described in the *IMS/ESA Administration Guide: System*.

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. For a description of the privileges required to issue a DB2 command, see the description of the appropriate DB2 command in this book.

### 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 (DB2).[4]

### 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.

# 4 The subsystem recognition character is defined in the IMS SSM member for the external subsystem.

## /START (IMS)

The IMS /START command (with the SUBSYS parameter) makes 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 is only a partial description of the /START command. For a complete description, see *IMS/ESA Operator's Reference.*

## 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, as described in the *IMS/ESA Administration Guide: System.*

## Syntax

```
►►──┬─/START──────────────────────────┬──────────────────────────────────────────►◄
    │          ┌─,──────────────┐     │
    ├─SUBSYS───▼─subsystem-name─┴─────┤
    └─SUBSYS──ALL─────────────────────┘
```

## 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, then restart the subsystem connection without bringing down IMS.

## -START DATABASE (DB2)

The DB2 command START DATABASE is typically used after a previous STOP DATABASE command, or after a table space, partition, or index has been placed in group buffer pool RECOVER pending status (GRECP), or if pages have been put on the logical page list (LPL) for that object. This command makes the specified database available for use. Depending on the options you specify, the database can be made available for read-only processing, read-write processing, or utility-only processing.

In a data sharing environment, the command can be issued from any DB2 on the group that has access to the database.

**Abbreviation**: -STA DB

## Environment

This command can be issued from an MVS 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

All databases specified for which the privilege set of the process has the STARTDB privilege are started. The privilege set of the process must include one of the following:

- STARTDB privilege
- DBMAINT, DBCTRL, or DBADM authority
- SYSCTRL or SYSADM authority

When this set does not contain the STARTDB privilege for a specified database, an error message is issued.

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

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

Database DSNDB06 contains the table spaces and index spaces required to check the authorization for using START DATABASE. If a table or index space required for this authorization check is stopped, or is unavailable because it is in LPL or GRECP status, then installation SYSADM authority is required to start any data base, table space, or index space, including the ones required for the authorization check. Installation SYSOPR authority may also start DSNDB06 but only when in LPL or GRECP status and if access mode is not changed.

## Syntax

```
              ┌─,─────────────┐
►►──START DATABASE──(──┬──▼─database-name─┴──────────┬──)──────────────────────────────►
                       ├─*───────────────────────────┤
                       ├─dbname1:dbname2─────────────┤
                       ├─dbname*─────────────────────┤
                       ├─*dbname─────────────────────┤
                       ├─*dbname*────────────────────┤
                       └─*dbstring1*dbstring2*───────┘

►──┬──────────────────────────────────────────────────────────────────────────────┬──►
   │                 ┌─,──────────┐                                               │
   └─SPACENAM(──┬──▼─space-name─┴──────────────┬──)──┬──────────────────────────────┬┘
                ├─*────────────────────────────┤     │       ┌─,───────┐          │
                ├─spacename1:spacename2────────┤     └─PART(─┬─▼─integer─┴──────┬─)─┘
                ├─spacename*───────────────────┤             └─integer1:integer2─┘
                ├─*spacename───────────────────┤
                ├─*spacename*──────────────────┤
                └─*spacestring1*spacestring2*──┘

►──┬────────────────────────────┬──────────────────────────────────────────────────►◄
   │          ┌─RW────┐         │
   └─ACCESS(──┼─RO────┼──)──────┘
              ├─UT────┤
              └─FORCE─┘
```

## Option descriptions

**(*database-name*, ...)**
Is the name of a database, or 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 by commas.

**(*)** Starts all databases for which the privilege set of the process has at least DBMAINT authority (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 specifying them explicitly (for example, START 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** | **Starts** |
|---|---|
| *dbname1:dbname2* | All databases whose names 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**

Tells what particular table spaces or indexes within the database 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***, ...)**

Is 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 by commas.

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 uses.[5]

You cannot use a partial name or 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 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 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* |

[5] Consecutive pattern-matching characters (*) are not allowed, and you cannot specify two pattern-matching characters (*) in the middle of a keyword string.

**PART (*integer*, ...)**

Indicates 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 *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 other valid partitions that you specified are started.

*integer* can be written to designate either:

- A list of one or more partitions, or
- A range of all partition numbers that collate greater than or equal to *integer1* and less than or equal to *integer2*

Both a list and a range cannot be specified.

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.

**ACCESS**

Tells whether the objects started can be read from and written to, read from only, or accessed by utilities only.

**Abbreviation**: ACC

**(<u>RW</u>)**

Allows programs to read from or 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 change data 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 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.

With 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 (the utility is terminated) only if all 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.

A table space or index space started with ACCESS(FORCE) might be in an inconsistent state. See “Usage notes” for further instructions.

## Usage notes

***Data sets offline:*** It is not necessary for every disk pack containing partitions, table spaces, or indexes 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, 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.

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 this 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.

***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 OS/390, 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. For information on DSN1COPY, see *DB2 Utility Guide and Reference*.

If an application process requests a transaction lock on a table space that is in a restrictive status (RECP, PSRCP) 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), then START DATABASE ACCESS (FORCE) is not allowed.

| START DATABASE ACCESS(FORCE) does not execute if postponed abort or
| indoubt units of recovery exist. If you attempt to issue the START DATABASE
| ACCESS(FORCE) command in this situation, the command fails. FORCE cannot
| be used to reset the restart pending (RESTP) state. See Section 2 of *DB2 Utility*
| *Guide and Reference* for information about resetting the RESTP state.

***Restricted mode (RO or UT):*** When a START DATABASE command for a restricted mode (RO and UT) takes effect depends upon 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 started after START DATABASE has completed, access restrictions are effective immediately. For applications 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 upon various factors. These factors include the ACCESS mode specified on the START DATABASE command, the type of drain activity, if any, on the table space or partition, and whether there are held cursors 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 temp 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 DSNI022I is issued periodically to give you the progress of the recovery. The starting of databases, table spaces, or indexes (a synchronous task) often completes before the recovery operation starts. Therefore, when DB2 issues message DSN9022I, which indicates that synchronous processing has completed, the recovery of objects might not be complete. Message DSNI006I is issued in response to START DATABASE when the object (table space or index space) identified by TYPE and NAME had group buffer pool RECOVER pending (GRECP) or logical page list (LPL) status, and recovery was triggered. The START DATABASE command does not complete until the asynchronous task of recovery completes.

Message DSNI021I indicates that asynchronous processing for an object has completed. You can issue the command DISPLAY DATABASE to determine whether the recovery operation for all objects is complete. If it 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 DSN8S61E in database DSN8D61A. Recover the table space if it is in GRECP status or recover the pages on the LPL if one exists.

```
-START DATABASE (DSN8D61A) SPACENAM (DSN8S61E)
```

***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 DSN8S61E in database DSN8D61A 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 (DSN8D61A) SPACENAM (DSN8S61E) 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 similar to the following output:

```
DSN9022I - DSNTDDIS 'START DATABASE' NORMAL COMPLETION
```

## -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 the DSNJU003 (change log inventory) utility. For more details on the effects, see “Usage notes” on page 286 and the description of the DSNJU003 utility in *DB2 Utility Guide and Reference*.

**Abbreviation**: -STA DB2

### Environment

This command can be issued only from an MVS console. The name of the DB2 subsystem is determined by the command prefix. (This was called the subsystem recognition character (SRC) in previous DB2 releases). 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 an MVS console with the START command capability. See *OS/390 MVS System Commands*.

### Syntax

```
►►──START DB2──┬─────────────────────────────┬──┬──────────────────────┬──┬──────────────────────────┬──────────►
               │         ┌─DSNZPARM────┐     │  │          ┌─*─────┐   │  └─MSTR(jcl-substitution)───┘
               └─PARM(───┴─module name─┴──)──┘  └─ACCESS(───┴─MAINT─┴──)─┘

►──┬─────────────────────────┬──┬─────────────────────────┬──────────────────────────────────────────────────►◄
   └─DBM1(jcl-substitution)──┘  └─DIST(jcl-substitution)──┘
```

### Option descriptions

None of the following options are required.

**PARM(*module-name*)**
Specifies the load module that contains the DB2 subsystem parameters.

*module-name* is the name of a load module provided by the installation.

The **default** is **DSNZPARM**, which is provided by DB2.

**ACCESS**
Tells 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 install SYSADM and install SYSOPR.

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.

**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 the strings by commas and enclose the entire list between a single pair of apostrophes.

# **Recommendation:** Omit the keyword and use the parameters 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, these actions occur during restart:

- Log records are processed to the extent determined by the conditional restart control record.
- These 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.

***Endless wait during start:*** It is possible for the start operation to 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 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 MVS system on which they are to be started, MVS1 and MVS2.

```
ROUTE MVS1,-DB1G START DB2
ROUTE MVS2,-DB2G START DB2
```

## -START DDF (DB2)

The DB2 command START DDF starts the distributed data facility (DDF) if it has not already been started.

**Abbreviation**: -STA DDF

### Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
►►──START DDF──────────────────────────────────────────────────────────────────►◄
```

### Usage note

The START DDF command activates the DDF interface to VTAM and TCP/IP.
# When issued after STOP DDF MODE(SUSPEND), suspended threads are resumed
# and DDF activity continues.

### Example

Start the distributed data facility.

```
-START DDF
```

## -START FUNCTION SPECIFIC (DB2)

The DB2 command START FUNCTION SPECIFIC activates 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 activates 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 an MVS console, a DSN session under TSO, a DB2I panel, an IMS or CICS terminal, or a program using the instrumentation facility interface (IFI).

**Data sharing scope**: Member

### Authorization

To execute this command, the privilege set of the process must include one of the following authorities for each function:

- Ownership of the function
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

If -START FUNCTION SPECIFIC *.* or schema.partial-name* is specified, 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 an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
                           ┌─(*.*)───────────────────────────────────┐
►►──START FUNCTION SPECIFIC─┤                                         ├──────────────────────►◄
                           │    ┌─,──────────────────────────────┐   │
                           └─(──▼─┬─schema.specific-function-name─┴─)─┘
                                  └─schema.partial-name*──────────┘
```

## 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 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**
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.

## 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 UDF load modules, issue the following WLM command.

```
VARY WLM, APPLENV=applenv,REFRESH
```

## Examples

***Example 1:*** Start all functions.

```
-START FUNCTION SPECIFIC
```

Output similar to the following output is generated:

```
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 similar to the following output is generated:

```
DSN9022I - DSNX9COM '-START FUNC' NORMAL COMPLETION
```

## START irlmproc (MVS IRLM)

The START *irlmproc* command starts an IRLM component with a procedure put in place by the installation. Symbolic parameters in the procedure can be overridden on the START *irlmproc* command.

## Environment

This command can be issued only from an MVS console.

**Data sharing scope**: Member

## Authorization

The command requires an appropriate level of MVS authority, as described in *OS/390 MVS System Commands.*

## Syntax

```
                       ┌─,─────────────────────────┐
►►──START──irlmproc,───▼─┬─DEADLOK='iiii,kkkk'─────┬─┴────────────────────────────────────────────►◄
                         ├─IRLMGRP='irlm-group-name'─┤
                         ├─IRLMID=n──────────────────┤
                         ├─IRLMNM=irlmname───────────┤
                         ├─LOCKTABL=irlmltnm─────────┤
                         ├─MAXCSA=nnn────────────────┤
                         ├─MAXUSRS=nnn───────────────┤
                         ├─PC=──┬─YES─┬──────────────┤
                         │      └─NO──┘              │
                         ├─SCOPE=──┬─LOCAL────┬──────┤
                         │         ├─GLOBAL───┤      │
                         │         └─NODISCON─┘      │
                         └─TRACE=──┬─NO──┬───────────┘
                                   └─YES─┘
```

Options must be separated by commas with no spaces.

## Option descriptions

*irlmproc*
: Is the procedure name of the IRLM to be started.

# None of the following options is 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 from 1 to 9999 that specifies the length in
#  seconds of the IRLM local deadlock-detection interval. Any value from 1 to
9999 may be specified but if the value is greater than 5, IRLM uses 5.

*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. Any value from 1 to 9999 may be specified but IRLM uses 1.
The recommended value to specify is 1.

In a data sharing environment, IRLM synchronizes all of the DEADLOK values in the group to the values specified on the most recent IRLM to join the group. The DEADLOK values may be changed by starting a member with the values desired. To reduce confusion, it is recommended that the installation specify the same value for DEADLOK on all of its IRLM start-up 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 sharing 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. The IRLM with the lowest ID in the group becomes the global deadlock manager for the group when in you are in a data sharing mode.

*n* can be either a 1- to 3-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 MVS interprets quotes,
# there must be single quotes 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 MVS subsystem name assigned to this IRLM. (Although MVS 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.

**MAXCSA=** *nnn*
# Specifies the maximum amount of CSA (including ECSA) the IRLM can use
# for its lock structures. *nnn* must be a one- to three-digit number from 1 to
# 999. The number indicates what multiple of 1 MB of storage the IRLM will
use. For example, MAXCSA=5 allows the IRLM to use 5 MB of CSA and ECSA. In displays, this storage is called *accountable* storage because it is

accountable to the value you set for MAXCSA. This parameter is ignored when PC=YES.

The **default** value for MAXCSA is **6 MB**.

Use the accountable storage high water mark (AHWM) data from messages DXR100I and DXR121I to monitor IRLM's usage of common storage, and as a basis for adjustments to the MAXCSA value.

**MAXUSRS=***nnn*
Specifies the initial maximum number of members in the data sharing group. The specified value determines the size of each lock entry in the lock table portion of the lock structure, as shown in Table 26.

*Table 26. Effect of MAXUSRS on initial lock table entry size*

| 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=**
Specifies whether the IRLM is to use the MVS cross-memory services. The specification of this value indicates where the lock control block resides.

**YES** Uses the cross-memory services. The lock control block resides in IRLM private storage. The MAXCSA parameter is ignored.

**NO** Does not use the cross-memory services. The lock control block resides in ECSA. The MAXCSA parameter is used.

**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 DBMSs 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 MVS 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. The default is TRACE NO.

**NO**
Does not capture traces unless the TRACE CT command is issued. See TRACE CT (MVS IRLM) on page 341 for details.

**YES**
Captures traces in wrap-around buffers.

## Example

Enter the following command on the MVS system console:

```
S irlmproc,MAXCSA=8
```

This command starts the IRLM with 8 MB for MAXCSA which controls the CSA (including ECSA) usage for locks when PC=NO.

## -START PROCEDURE (DB2)

For both DB2-established and WLM-established stored procedure address spaces, the DB2 command START PROCEDURE activates the definition of a stored procedure that is stopped or refreshes one that is cached. You can qualify stored procedure names with a schema name.

One of the following can also occur:

- If the DB2-established stored procedures address space is not connected to DB2, MVS starts it.
- If the stored procedures address space is already connected, and some procedure listed in the command is stopped, DB2 stops and restarts the Language Environment environment. At restart, DB2 deletes the existing stored procedure load modules from memory. A deleted load module is reloaded when a CALL statement for that procedure is executed. For WLM-established stored procedures address spaces, a WLM command is needed to do the reload. For example:

  ```
  MVS VARY WLM,APPLENV=applenv,REFRESH
  ```

On successful completion of the command, queued requests for the specified stored procedures begin executing. The abend counts for those procedures are set to zero. DB2 resets the MAXQUE and TIMEOUT statistics to 0 each time you execute the START PROCEDURE command.

You do not have to issue START PROCEDURE when defining 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 an MVS 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, the privilege set of the process must include one of the following:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority
- Ownership of the stored procedure.

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
            ┌─(*.*)──────────────────────────────────┐
►►─START PROCEDURE─┼────────────────────────────────────────┼──────────────────────────────────────►◄
            │     ┌─,──────────────────────────┐     │
            └─(─▼─┬─schema.procedure-name─┬──┴─)──┘
                  ├─schema.partial-name*──┤
                  ├─procedure-name────────┤
                  └─partial-name*─────────┘
```

## 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 beginning with ABC in the PAYROLL schema.

*procedure-name*
Marks the 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.

## Usage notes

***Errors in a definition of a stored procedure:*** Errors are detected at create time for a stored procedure. See CREATE PROCEDURE in Chapter 6 of *DB2 SQL Reference* for more information.

***Management of stored procedures address space:*** The START PROCEDURE command works differently depending upon how the DB2 stored procedures address spaces are managed. WLM-established stored procedure address spaces are controlled by MVS WLM, rather than the START PROCEDURE command. For more information, see Section 5 (Volume 2) of *DB2 Administration Guide*.

## Examples

***Example 1:*** Start all stored procedures.

```
-START PROCEDURE
```

This command produces output similar to the following output:

```
DSNX946I - DSNX9ST2 START PROCEDURE SUCCESSFUL FOR *.*
DSN9022I - DSNX9COM '-START PROC' NORMAL COMPLETION
```

***Example 2:*** Make specific stored procedures available to be called, and start any requests waiting for those procedures.

```
-START PROCEDURE(USERPRC1,USERPRC2)
```

This command produces output 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
```

## -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 active. The resource limit specification table 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 an MVS 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, the privilege set of the process must include one of the following:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands issued from an MVS console are not associated with any 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 characters specified when the table was created. See Section 5 (Volume 2) of *DB2 Administration Guide* for more information about resource limit specification tables.

  The full name of the table is *authid.*DSNRLST*id*, where *authid* is the value that was specified in field RESOURCE AUTHID on installation panel DSNTIPP.

  The **default** ID is the value that was specified in field RLST NAME SUFFIX on installation panel DSNTIPO.

## Example

Start the resource limit facility.

```
-START RLIMIT ID=01
```

## -START TRACE (DB2)

The DB2 command START TRACE starts DB2 traces. For more information about the trace facility, see Section 5 (Volume 2) of *DB2 Administration Guide*.

There is an additional option to this command and values for a few options that are not described here. They are intended for service and use under the direction of IBM support personnel. For details, see *DB2 Diagnosis Guide and Reference*.

**Abbreviation**: -STA TRA

## Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following:

- TRACE privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
►►──START TRACE──(──┬─PERFM───┬─)──┬───────────────────┬──┬──────────────────┬──┬─────────────────┬──►◄
                    ├─ACCTG───┤    └─destination block─┘  └─constraint block─┘  └─COMMENT(string)─┘
                    ├─STAT────┤
                    ├─AUDIT───┤
                    └─MONITOR─┘
```

**destination block:**

```
          ┌─,───┐
►──DEST(──▼─┬─GTF─┬┴─)──────────────────────────────────────────────────────────────────►
            ├─SMF─┤
            ├─SRV─┤
            ├─OPn─┤
            └─OPX─┘
```

**constraint block:**

```
         ┌─*───────────────┐                 ┌─*──────────────────────────┐            ┌─*───────────┐
         │ ┌─,─────────┐   │                 │ ┌─,───────────────────┐    │            │ ┌─,───────┐ │
►──PLAN(─┴─▼─plan-name─┴───┴─)──AUTHID(──────┴─▼─authorization-id────┴────┴─)──CLASS(──┴─▼─integer─┴─┴─)──────►

          ┌─*──────────┐                     ┌─*─────────┐              ┌─,───────────────┐
          │ ┌─,─────┐  │                     │           │              ▼                 │
►──IFCID(─┴─▼─ifcid─┴──┴─)(1)──BUFSIZE(─────┴─k_bytes───┴─)──TDATA(──┬─CORRELATION─┬─┴─)──────────────►
                                                                     ├─TRACE───────┤
                                                                     ├─CPU─────────┤
                                                                     └─DISTRIBUTED─┘

►──LOCATION(──┬─*─────────────────┬─)──────────────────────────────────────────────────►
              │ ┌─,─────────────┐ │
              ├─▼─location-name─┴─┤
              ├─<luname>──────────┤
              └─ipaddr────────────┘
```

**Note:**

1 Allows you to specify trace events in addition to the IFCIDs activated by the CLASS option. For details, see the description of **IFCID(***ifcid***, ...)** on page 306.

## Option descriptions

You must specify a trace type.

The options PERFM, ACCTG, STAT, AUDIT, and MONITOR identify the type of trace started.

**(PERFM)**
Is intended for performance analysis and tuning, and includes records of specific events in the system.

**Abbreviation**: P

**(ACCTG)**
Is intended to be used in accounting for a particular program or authorization ID, and includes records written for each thread.

**Abbreviation**: A

**(STAT)**
Collects statistical data broadcast by various components of DB2, at time intervals that can be chosen during installation.

**Abbreviation**: S

LOCATION cannot be specified when you choose a statistics trace.

**(AUDIT)**
Collects audit data from various components of DB2.

**Abbreviation**: AU

**(MONITOR)**
Collects monitor data. Makes trace data available to DB2 monitor application programs.

**Abbreviation**: MON

**COMMENT(*string*)**
Gives a comment that is reproduced in the trace output (except in the resident trace tables), and 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.

## The destination block

**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 Table 27.

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 27. Allowable destinations for each trace type*

| Type | GTF | SMF | SRV | OP*n* | OPX |
|---|---|---|---|---|---|
| PERFM | Default | Allowed | Allowed | Allowed | Allowed |
| ACCTG | Allowed | Default | Allowed | Allowed | Allowed |
| STAT | Allowed | Default | Allowed | Allowed | Allowed |
| AUDIT | Allowed | Default | Allowed | Allowed | Allowed |
| MONITOR | Allowed | Allowed | Allowed | Allowed | Default |

The meaning of each value is as follows:

| **Value** | **Trace output is recorded by** |
|---|---|
| GTF | The MVS 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 shown in the following list:

| **IFCID record** | **SMF record type** |
|---|---|
| 1 (SYSTEM SERVICES STATISTICS) | 100 |
| 2 (DATABASE SERVICES STATISTICS) | 100 |
| 3 (AGENT ACCOUNTING) | 101 |
| 202 (DYNAMIC SYSTEM PARAMETERS) | 100 |
| 230 (DATA SHARING GLOBAL STATISTICS) | 100 |
| 239 (AGENT ACCOUNTING OVERFLOW) | 101 |
| ALL OTHERS | 102 |

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. For further information on starting a trace via an application program, see Appendix E (Volume 2) of *DB2 Administration Guide*.

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 block

The constraint block places optional constraints on the kinds of data collected by the trace. The allowable constraints depend on the type of trace started, as shown in the following table:

*Table 28. Allowable constraints for each trace type*

| Type | PLAN | AUTHID | CLASS | LOCATION |
|---|---|---|---|---|
| PERFM | Allowed | Allowed | Allowed | Allowed |
| ACCTG | Allowed | Allowed | Allowed | Allowed |
| STAT | NO | NO | Allowed | NO |
| AUDIT | Allowed | Allowed | Allowed | Allowed |
| MONITOR | Allowed | Allowed | Allowed | Allowed |

The meaning of each option is as follows:

**PLAN(***plan-name***, ...)**
Introduces a list of specific plans for which trace information is gathered. 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.

**AUTHID(***authorization-id***, ...)**
Introduces a list of specific authorization IDs for which trace information is gathered. 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.

**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 this option is omitted, all the default classes within the trace type are activated. The **default** classes for each trace type are marked by asterisks (*) in Table 29.

**(*)** Starts a trace for all classes of the trace type.

*integer*
Is any number in the list that follows. You can use any number of classes that are allowed for the type of trace started.

| *Table 29 (Page 1 of 3). Classes for DB2 trace types*

| # Trace type | Class | Description of class | Activated IFCIDs |
|---|---|---|---|
| Accounting | 1* | Standard accounting data | 3,106,239 |
| | 2 | Entry or exit from DB2 event signalling | 232 |
| | 3 | Elapsed wait time in DB2 | 6-9,32,33,44,45,117,118,127,128,170,171, \| 174,175,213-216,226,227,242,243,321,322, # 329 |
| | 4 | Installation-defined accounting record[1] | 151 |
| | 5 | Time spent processing IFI requests | 187 |
| | 6 | Reserved | |
| | 7 | Entry or exit from DB2 event signalling for package and DBRM accounting | 232,240 |
| | 8 | Wait time for a package | 6-9,32,33,44,45,117,118,127,128,170,171, \| 174,175,213-216,226,227,241-243,321,322 |
| | 10 - 29 | Reserved | |
| | 30 - 32 | Available for local use | |
| Audit | 1* | Access attempts denied due to inadequate authorization | 140 |

| *Table 29 (Page 2 of 3). Classes for DB2 trace types*

| # | Trace type | Class | Description of class | Activated IFCIDs |
|---|---|---|---|---|
| | | 2 | Explicit GRANT and REVOKE | 141 |
| | | 3 | CREATE, ALTER, and DROP operations against audited tables | 142 |
| | | 4 | First change of audited object | 143 |
| | | 5 | First read of audited object | 144 |
| | | 6 | Bind time information about SQL statements that involve audited objects | 145 |
| | | 7 | Assignment or change of authorization ID | 55,83,87,169,312 |
| | | 8 | Utilities | 23,24,25 |
| | | 9 | Installation-defined audit record[1] | 146 |
| | | 10 - 29 | Reserved | |
| | | 30 - 32 | Available for local use | |
| | Statistics | 1* | Statistics data | 1,2,105,106,202 |
| | | 2 | Installation-defined statistics record[1] | 152 |
| \| | | 3 | Deadlock, group buffer pool, data set extension information, and indications of long-running URs and active log space shortages | 172,196,250, 258, 261,262,313, 330 |
| | | 4 | DB2 exceptional conditions | 191-195,203-210,235,236,238,267,268 |
| | | 5 | DB2 data sharing statistics record | 230 |
| | | 6 - 7 | Reserved | |
| \| | | 8 | Data set I/O statistics | 199 |
| \| | | 9 - 29 | Reserved | |
| | | 30 - 32 | Available for local use | |
| | Performance | 1* | Background events | 1,2,31,42,43,76-79,102,103,105-107,153 |
| | | 2* | Subsystem events | 3,68-75,80-89,106,174,175, |
| \| | | 3* | SQL events | 22,53,55,58-66,92,95-97,106,112,177, 233,237, 250, 272,273, 325 |
| \| | | 4 | Reads to and writes from the buffer and EDM pools | 6-10,29-30,105-107,127,128,226,227, 321,322 |
| | | 5 | Write to log; archive log | 32-41,104,106,114-120,228,229 |
| | | 6 | Summary lock information | 20,44,45,105-107,172,196,213,214,218 |
| | | 7 | Detailed lock information | 21,105-107,223 |
| | | 8 | Data scanning detail | 13-18,105-107,125,221,222,231,305, 311 |
| | | 9 | Sort detail | 26-28,95-96,106 |
| | | 10 | BIND, commands, and utilities detail | 23-25,90,91,105-107,108-111,201,256 |
| | | 11 | Storage usage events | 46-52,56,57,93,94,106,113 |
| | | 12 | Storage manager | 98-101,106 |

| *Table 29 (Page 3 of 3). Classes for DB2 trace types*

| # Trace type | Class | Description of class | Activated IFCIDs |
|---|---|---|---|
| | 13 | Edit and validation exits | 11,12,19,105-107 |
| | 14 | Entry from and exit to an application | 67,106,121,122 |
| | 15 | Installation-defined performance record[1] | 154 |
| | 16 | Distributed processing | 157-163,167,183 |
| | 17 | Claim and drain information | 211-216 |
| | 18 - 19 | Reserved | |
| | 20 | Data sharing coherency summary | 249-251,256-257,261,262,267,268 |
| | 21 | Data sharing coherency detail | 255,259,263 |
| | 22 | Authorization exit parameters | 314 |
| | 23 - 29 | Reserved | |
| | 30 - 32 | Available for local use | |
| \| Monitor | 1* | Activate the READS IFCIDs | 1,2,106,124,129,147,148-150, 199, 202, 230,254,306, 316(2),317 |
| | 2 | Entry or exit from DB2 event signalling | 232 |
| \| | 3 | DB2 wait time for I/O, locks; resource usage information | 6-9,32,33,44,45,117,118,127, 128,170,171,174,175,213,214, 215,216,226,227,242,243,321,322 |
| | 4 | Installation-defined monitor record[1] | 155 |
| | 5 | Time spent processing IFI requests | 187 |
| | 6 | Changes to tables created with DATA CAPTURE CHANGES | 185 |
| | 7 | Entry or exit from DB2 event signalling for package and DBRM accounting | 232,240 |
| \| | 8 | Wait time for a package | 6-9,32,33,44,45,51,52,56,57, 117,118,127,128,170,171,174, 175,213-216,226,227,241-243,321,322 |
| | 9 - 29 | Reserved | |
| | 30 - 32 | Available for local use | |

**Notes to Table 29:**

- An asterisk (*) indicates a default class for a trace type.
- 1. For instructions on using the IFCIDs, see Appendix D (Volume 2) of *DB2 Administration Guide*.
- 2. DB2 does not collect statistical data for this record unless IFCID 318 is activated. IFCID 318 is not associated with any trace class; you must start it on its own.

**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 on page 309 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 1 through 350, with the exception of: 4, 5, 185, 187, 217, 232, 234, 240, and 241. These exceptions are invalid values for the IFCID option. IFCIDs 4 and 5 are always automatically active. 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 185 | monitor trace class 6 |
| IFCID 232 | monitor trace class 2 or 7, or accounting trace class 2 or 7 |
| IFCID 240 | monitor trace class 7 or accounting trace 7 |
| IFCID 241 | 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 8 KB to 1024 KB in 4 KB increments. If you specify a value outside of this range, then the range limit closest to the specified value is used. To allocate a buffer size of 8 KB, you would specify BUFSIZE(8).

The **default** is **BUFSIZE(*)**, which is the size set when DB2 was installed.

**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 MVS TCB or SRB executing.

**DISTRIBUTED**
Places a distributed header on the record.

**Abbreviation**: DIST

**LOCATION(***location-name***, ...)**
Introduces a list of specific location names for which trace information is gathered. The use of the LOCATION option precludes tracing threads that have no distributed data relationship. LOCATION 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 8 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.*nnn.nnn.nnn*is the dotted decimal 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 OS/390**: DB2 does not receive a location name from clients that are not DB2 for OS/390 subsystems. To start a trace for a client that is not a DB2 for OS/390 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 (123.34.101.98)
```

## Usage notes

***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. See Appendix E (Volume 2) of *DB2 Administration Guide* for more information on qualifying monitor trace class 1 IFCIDs.

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.

## Examples

***Example 1:*** Start a performance trace for threads with remote activity to location USIBMSTODB21. Only activate IFCIDs 44 (lock suspends) and 54 (lock contention). Trace class 30 is available for installation use.

```
-START TRACE (PERFM)
  DEST(GTF)
  LOCATION(USIBMSTODB21)
  CLASS(30)
  IFCID(44)
```

***Example 2:*** Start an accounting trace for plan DSN8BC61.  Write records to SMF (that will happen by default). Include a comment to identify the trace.

```
-START TRACE (ACCTG)
  PLAN (DSN8BC61)
  COMMENT ('ACCTG TRACE FOR DSN8BC61')
```

***Example 3:*** Start the statistics trace. Write records to SMF (by default).

```
-START TRACE=S
```

***Example 4:*** Start monitor tracing (usually done by an application program). Write records to OPX (by default).

```
-START TRACE(MON)
```

## /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. For a complete description, see *IMS/ESA Operator's Reference.*

### 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, as described in the *IMS/ESA Administration Guide: System.*

### Syntax

```
                      ┌─,──────────────┐
►►─/STOP──┬─SUBSYS───▼─subsystem-name─┴──┬──────────────────────────────────────────────►◄
          └─SUBSYS─ALL───────────────────┘
```

### 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.

## -STOP DATABASE (DB2)

The DB2 command STOP DATABASE makes the specified databases unavailable for applications and closes their data sets.

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 an MVS 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

All databases specified for which the set of privileges held by the privilege set of the process has the STOPDB privilege are stopped. Error messages are produced for those databases specified for which this set does not have the STOPDB privilege. The privilege set of the process must include one of the following:

- STOPDB privilege
- DBMAINT, DBCTRL, or DBADM authority
- SYSCTRL or SYSADM authority

DB2 commands issued from an MVS console are not associated with any 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 containing a registration table or index.

Database DSNDB06 contains the table spaces and index spaces required to check authorization. If you stop any table space or index space required for the START DATABASE authorization check, then installation SYSADM authority is required to restart it.

## Syntax

```
►►──STOP DATABASE──(──┬─┌─,─────────────┐─┬──)─────────────────────────────────────►
                      │ ▼               │ │
                      │ └database-name──┘ │
                      ├─*─────────────────┤
                      ├─dbname1:dbname2───┤
                      ├─dbname*───────────┤
                      ├─*dbname───────────┤
                      ├─*dbname*──────────┤
                      └─*dbstring1*dbstring2*┘

►──┬──────────────────────────────────────────────────────────────────────────┬──►
   └─SPACENAM(──┬─┌─,──────────┐──────────┬──)──┬────────────────────────────┬─┘
                │ ▼            │          │     └─PART(──┬─┌─,───────┐──┬──)─┘
                │ └space-name──┘          │              │ ▼         │  │
                ├─*───────────────────────┤              │ └integer──┘  │
                ├─spacename1:spacename2───┤              └─integer1:integer2┘
                ├─spacename*──────────────┤
                ├─*spacename──────────────┤
                ├─*spacename*─────────────┤
                └─*spacestring1*spacestring2*┘

►──┬──────────────┬───────────────────────────────────────────────────────────►◄
   └─AT(COMMIT)───┘
```

## 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.

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 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.[6]

See "Usage notes" on page 314 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 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 either:

[6] Consecutive pattern-matching characters (*) are not allowed, and you cannot specify two pattern-matching characters in the middle of a keyword string.

- A list of one or more partitions, or
- A range of all partition numbers that collate greater than or equal to *integer1* and less than or equal to *integer2*

Both a list and a range cannot be specified.

PART is valid with partitioned table spaces, partitioned indexes, and non-partitioned 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.

**AT(COMMIT)**
Marks the specified object as being in STOPP 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, MVS, and batch:*** When the STOP DATABASE command is issued from a TSO or an MVS 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.

***Databases DSNDDF and DSNRLST:*** 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. For more information on the CDB, see Section 1 (Volume 1) of *DB2 Installation Guide*, and for the RLST, see Section 5 (Volume 2) of *DB2 Administration Guide*.

***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, PSRCP) 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 DASD 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 MVS 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 MVS parmlib member or by issuing the MVS 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 30 (Page 1 of 2). 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. |

Table 30 (Page 2 of 2). Locking used by the STOP DATABASE command

| Command | Table space type | | Locks acquired |
|---|---|---|---|
| | *Non-partitioned* | | IX mass delete lock. Drain-all on table space. |
| STOP | *Partitioned LOCKPART YES* | PART | X-lock partitions specified. Drain-all on partitions specified. |
| | | | X-lock all partitions. Drain-all on all partitions. |
| | *Partitioned LOCKPART NO* | PART | X-lock table space. Drain-all on partitions specified. |
| | | | X-lock table space. Drain-all on all partitions. |
| | *Non-partitioned* | | X-lock table space. Drain-all on table space. |

## Examples

***Example 1:*** Stop table space DSN8S61E in database DSN8D61A and close the data sets that belong to that table space.

```
-STOP DATABASE(DSN8D61A) SPACENAM(DSN8S61E)
```

***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 DSN8D61A. 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(DSN8D61A) SPACENAM(XEMP2) PART(1)
```

***Example 5:*** Stop all table spaces with names that begin with "T" and end with the "IQUA03" string in database DSN8D61A.

```
-STOP DATABASE(DSN8D61A) 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(DSN8D61A) SPACE(T*IQUA03)
```

## -STOP DB2 (DB2)

The DB2 command STOP DB2 stops the DB2 subsystem.

**Abbreviation**: -STO DB2

### Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following:

- STOPALL privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
►►──STOP DB2──┬─MODE(QUIESCE)─┬──┬─CASTOUT(YES)─┬──────────────────────────────────────────────►◄
              └─MODE(FORCE)───┘  └─CASTOUT(NO)──┘
```

### Option descriptions

**MODE**
: Indicates whether currently executing programs will be allowed to complete. For the effects of this option on distributed threads, see the description of the MODE option of "-STOP DDF (DB2)" on page 319.

  **(QUIESCE)** (default)
  : Allows currently executing programs to complete processing. No new program is allowed to start.

  **(FORCE)**
  : Terminates currently executing programs, including utilities. No new program is allowed to start. MODE(FORCE) will probably cause 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.

# **<u>YES</u>**
# 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 MVS commands.

***MODE(FORCE):*** A forced stop does not cause an immediate abend. If a connected task is executing outside DB2, DB2 posts an exit 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 proc-
# essing 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 con-
# sistent data on DASD, (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 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)
```

## -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 an MVS 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, the privilege set of the process must include one of the following authorities:

- SYSOPR
- SYSCTRL
- SYSADM

DB2 commands that are issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
   ┌─MODE(QUIESCE)──────────────────┐
►►─STOP DDF─┼────────────────────────────────┼──────────────────────────────────────────────►◄
   ├─MODE(FORCE)────────────────────┤
#  └─MODE(SUSPEND)──┬─CANCEL(n)─┬───┘
#                   └─WAIT(n)───┘
```

## Option descriptions

**MODE**

# Indicates whether currently executing active distributed threads are allowed to
# complete.

# **<u>(QUIESCE)</u>**

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*)**
Specifies to cancel 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*)**
Specifies to resume 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 “Usage notes” on page 89 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 “-CANCEL THREAD (DB2)” on page 88.

Table 31 on page 321 summarizes the actions that DB2 takes when START DDF or STOP DDF are issued with different DDF states.

*Table 31. The result of commands on the DDF status*

| | Command issued | | | |
|---|---|---|---|---|
| **DDF status** | **START DDF** | **STOP DB2 or STOP DDF without MODE(FORCE)** | **STOP DB2 or STOP DDF with MODE(FORCE)** | **STOP DDF 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)
```

## -STOP FUNCTION SPECIFIC

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 an MVS 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, the privilege set of the process must include one of the following authorities for each function:

- Ownership of the function
- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

If -STOP FUNCTION SPECIFIC *.* or schema.partial-name* is specified, 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 an MVS console are not associated with any secondary authorization IDs.

### Syntax

```
►►──STOP FUNCTION SPECIFIC──┬─(*.*)───────────────────────────────────────┬──────────────────────►
                            │    ┌─,─────────────────────────────┐        │
                            └─(──▼─┬─schema.specific-function-name─┬─┴──)─┘
                                   └─schema.partial-name*──────────┘

►─┬──────────────────────────────┬───────────────────────────────────────────────────────────────►◄
  │             ┌─QUEUE──┐       │
  └─ACTION──(───┴─REJECT─┴──)────┘
```

## 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.

## Usage notes

***Limitations of STOP FUNCTION SPECIFIC***: STOP FUNCTION SPECIFIC is only applicable to external functions that run in the WLM application environment. STOP FUNCTION SPECIFIC cannot stop a built-in function or a user-defined function 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 MVS load module
- Rename or delete the MVS load module

## 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:

```
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:

```
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:

```
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:

```
DSN9022I - DSNX9COM '-STOP FUNC' NORMAL COMPLETION
```

## STOP irlmproc (MVS 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 an MVS console.

**Data sharing scope**: Member

## Authorization

The command requires an appropriate level of MVS authority, as described in *OS/390 MVS System Commands*.

## 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. If there are outstanding DB2 requests in process and IRLM will not terminate, use the MVS CANCEL command. If all other means of removing the subsystem fail, issue the MVS FORCE CANCEL command:

```
F irlmproc,ABEND,DUMP
```

## Example

Enter on the MVS1 system console:

```
P KRLM1
```

IRLM responses on MVS1 system console:

```
DXR165I IR21 TERMINATED VIA IRLM MODIFY COMMAND
DXR121I IR21 END-OF-TASK CLEANUP SUCCESSFUL - HI-CSA     325K
```

Response on MVS2 system console:

```
DXR025I JRLM SESSION LOST, SHARING STATE IS IRLM FAILED
```

The operator on system 1 has terminated the IRLM procedure named KRLM1. The operator on system 2 is informed that the IRLM in system 1 has terminated, but no operator action on system 2 is required.

# **In a data sharing environment:** You cannot issue the P command to IRLM in a data sharing group until there are no DB2s identified and the IRLM has issued 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
```

## -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.

If the DB2 established stored procedure address space is connected to DB2, MVS stops it based on the syntax of the STOP PROCEDURE command.

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 an MVS 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, the privilege set of the process must include one of the following:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority
- Ownership of the stored procedure.

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

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
                    ┌─(*.*)──────────────────────────────────┐
►►──STOP PROCEDURE──┼────────────────────────────────────────┤ ┌──────────────────────────────┐──────────►◄
                    │    ┌─,─────────────────────────┐       │ │              ┌─QUEUE──┐      │
                    └─(──▼─┬─schema.procedure-name─┬─┴──)────┘ └─ACTION──(──┴─REJECT─┴──)──┘
                           ├─schema.partial-name*──┤
                           ├─procedure-name────────┤
                           └─partial-name*─────────┘
```

## Option descriptions

| <u>(*.*)</u>
| Stops access to all stored procedures in all schemas, including procedure definitions that have not yet been accessed by DB2 applications. The DB2-established stored procedures address space terminates after active work is complete.

| **(***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 proce-
| dures in the PAYROLL schema.

*procedure-name*
| Identifies one or more specific stored procedure names to be stopped. The pro-
| cedure 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 proce-
# dure 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.

**<u>(QUEUE)</u>** 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

## 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. See Chapter 6
  | of *DB2 SQL Reference* for more information.
- Use an ALTER PROCEDURE statement.
- Rename or delete the MVS load module.

***Stored procedure address space management differences:*** The STOP PROCEDURE command operates differently depending upon how the DB2 stored procedures address spaces are established. For more information, see Section 4 (Volume 1) of *DB2 Administration Guide*.

## 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
```

## -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 an MVS 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, the privilege set of the process must include one of the following:

- SYSOPR authority
- SYSCTRL authority
- SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
►►──STOP RLIMIT──────────────────────────────────────────────────────────────►◄
```

## Example

Stop the resource limit facility.

```
-STOP RLIMIT
```

# -STOP TRACE (DB2)

The DB2 command STOP TRACE stops tracing.

There is an additional option to this command and values for a few options that are not described here. They are intended for service and use under the direction of IBM support personnel. For details, see *DB2 Diagnosis Guide and Reference*.

**Abbreviation**: -STO TRA

## Environment

This command can be issued from an MVS 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, the privilege set of the process must include one of the following:

- TRACE privilege
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

## Syntax

```
►►──STOP TRACE──(──┬─PERFM───┬─)──┬───────────────────┬──┬──────────────────┬──┬─────────────────┬──────►◄
                   ├─ACCTG───┤    └─destination block─┘  └─constraint block─┘  └─COMMENT(string)─┘
                   ├─STAT────┤
                   ├─AUDIT───┤
                   ├─MONITOR─┤
                   └─*───────┘
```

**destination block:**

```
         ┌─,─────┐
►──DEST(─▼┬─GTF─┬┴─)──────────────────────────────────────────────────────────────────────────────►
          ├─SMF─┤
          ├─SRV─┤
          └─OPn─┘
```

**constraint block:**

```
         ┌─*──────────────┐              ┌─*──────────────────────┐            ┌─*────────────┐
         │ ┌─,──────────┐ │              │ ┌─,──────────────────┐ │            │ ┌─,────────┐ │
►──PLAN(─┴─▼─plan-name──┴─┴─)──AUTHID(───┴─▼─authorization-id───┴─┴─)──CLASS(─┴─▼─integer──┴─┴─)────────►

        ┌─*──────────┐
        │ ┌─,──────┐ │
►──TNO(─┴─▼─integer┴─┴─)──LOCATION(─┬─*─────────────────┬─)───────────────────────────────────────►
                                    │ ┌─,─────────────┐ │
                                    ├─▼─location-name─┴─┤
                                    ├─<luname>──────────┤
                                    └─ipaddr────────────┘
```

## Option descriptions

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 (*)
```

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 -START 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 -START 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.

Each of the following keywords limits the command to stopping traces of the corresponding type. For further descriptions of each type, see "-START TRACE (DB2)" on page 300.

*Table 32. Trace types*

| Type | Description | Abbreviation |
|---|---|---|
| PERFM | Performance records of specific events | P |
| ACCTG | Accounting records for each transaction | A |
| STAT | Statistical data | S |
| AUDIT | Audit data | AU |
| MONITOR | Monitor data | MON |

**COMMENT(***string***)**
Gives a comment that is reproduced in the trace output record for the STOP TRACE command (except in the resident trace tables).

*string* is any SQL string; it must be enclosed between apostrophes if it includes a blank, comma, or special character.

**DEST**
Limits stopping to traces started for particular destinations. You can use more than one value, but do not use the same value twice. If you do not specify a value for DEST, DB2 does not use destination to limit which traces to stop.

**Abbreviation**: D

Possible values and their meanings are:

| **Value** | **Trace destination** |
|---|---|
| GTF | The generalized trace facility |
| SMF | The System Management Facility |
| SRV | An exit to a user-written routine |
| OP*n* | A specific destination. *n* can be a value from 1 to 8 |

See "-START TRACE (DB2)" on page 300 for a list of allowable destinations for each trace type.

**PLAN(***plan-name***, ...)**
Limits stopping to traces started for particular application plans. You can use up to eight plan names. If you use more than one name, you can use only one value for AUTHID, TNO, and LOCATION. Do not use this option with STAT.

The **default** is **PLAN(*)**, which does not limit the command.

**AUTHID(***authorization-id***, ...)**
Limits stopping to traces started for particular authorization identifiers. You can use up to eight identifiers. If you use more than one identifier, you can use only one value for PLAN, TNO, and LOCATION. Do not use this option with STAT.

The **default** is **AUTHID(*)**, which does not limit the command.

**CLASS(*integer*, ...)**
Limits stopping to traces started for particular classes. For descriptions of the allowable classes, see "-START TRACE (DB2)" on page 300. 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.

**LOCATION(*location-name*, ...)**
Introduces a list of specific location names for which traces are stopped. Limits the traces you can stop to those started for threads with connections to remote locations; the use of the LOCATION option precludes stopping traces of non-distributed threads.

You can specify up to eight location names. If you use more than one location name, you can only use one value for PLAN, AUTHID, and TNO. You cannot use this option with STAT.

# The **default** is **LOCATION( )**, which does not limit the command.

**(*)** LOCATION(*) limits the command to those traces that were started with the one or more location names specified on the LOCATION keyword of -START TRACE.

*<luname>*
Stops the DB2 trace for the remote clients that are connected to DDF through the remote SNA LU that you specify in *<luname>*.

*ipaddr*
Stops the DB2 trace for remote clients that are connected to DDF through the remote TCP/IP host. *nnn.nnn.nnn.nnn*is the dotted decimal IP address.

**Requesters other than DB2 for OS/390**: DB2 does not receive a location name from requesters that are not DB2. To display information about a requester that is not a DB2 for OS/390 subsystem, enter its LUNAME, enclosed by the less-than (<) and greater-than (>) symbols. For example, to display information about a requester with the LUNAME of LULA, enter the following command:

```
-STOP TRACE (*) LOCATION (<LULA>)
```

DB2 uses the <LUNAME> notation in messages displaying information about requesters that are not DB2 for OS/390.

## Usage notes

***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.

## 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 DSN8BC61. Include a comment.

```
-STOP TRACE (ACCTG)
  PLAN (DSN8BC61)
  LOCATION (USIBMSTODB21)
  COMMENT('ACCTG TRACE FOR DSN8BC61')
```

***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)
```

## -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 -TERM was issued. Active utilities perform this check periodically. If the utility is stopped, all its resources are released by -TERM. 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 an MVS 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, the primary or some secondary authorization ID of the process must be the ID that originally submitted the utility job, or the privilege set of the process must include one of the following:

- DBMAINT, DBCTRL, or DBADM authority
- SYSOPR, SYSCTRL, or SYSADM authority

DB2 commands issued from an MVS console are not associated with any secondary authorization IDs.

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

Database DSNDB06 contains the table spaces and index spaces required to check authorization. If a table or index space required for authorization checking is affected by a utility which you need to terminate, then installation SYSADM authority is required to terminate that utility.

### Syntax

```
►►──TERM UTILITY──(─┬─utility-id─────────┬─)──────────────────────────────────────────►◄
                    ├─partial-utility-id*─┤
                    └─*───────────────────┘
```

## 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*. For the control file name that is associated with each utility, see the description of the DSNU utility in *DB2 Utility Guide and Reference*.

If *utility-id* was created by default by the EXEC statement invoking DSNUTILB, then the token has the form *userid.jobname*.

**(***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 on jobs for each of the utilities:

| **Utility** | **Special effects of -TERM** |
|---|---|
| CATMAINT | None |
| CHECK DATA | None |
| CHECK INDEX | None |
| COPY | Inserts “T” record in SYSIBM.SYSCOPY. When you run COPY, it does not allow an incremental image copy if the “T” record exists. |
| DIAGNOSE | None |
| LOAD | See *DB2 Utility Guide and Reference* for the effect of -TERM on the LOAD utility phases |
| MERGECOPY | None |
| MODIFY | None |
| QUIESCE | None |
| RECOVER INDEX | Places the object being recovered in RECOVER pending status |

| | |
|---|---|
| RECOVER TABLESPACE | Places the object being recovered in RECOVER pending status |
| REORG | See *DB2 Utility Guide and Reference* for the effect of -TERM on the REORG utility phases |
| REPAIR | None |
| REPORT | None |
| RUNSTATS | None |
| STOSPACE | None |

## 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*)
```

## /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. For more information about the IMS Monitor, see *IMS/ESA Utilities Reference: System*.

**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, as described in *IMS/ESA Administration Guide: System*.

The syntax diagram below includes only those parameters that DB2 users need to know. For a diagram with the complete syntax of this command, see *IMS/ESA Operator's Reference*.

## Syntax

```
►►──/TRACE──SET──┬─ON──┬──┬──────────────────────┬──┬─────────────────────────┬──────────────────────────────────►◄
                 └─OFF─┘  │        ┌─ALL──┐      │  │         ┌─NOLOG─┐       │
                          └─TABLE──┴─SUBS─┴──────┘  └─OPTION───┴─LOG───┴───────┘
```

## Option descriptions

The option descriptions for the /TRACE command are described in *IMS/ESA Operator's Reference*; however, this section provides information about the two parameters 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
```

## TRACE CT (MVS IRLM)

The MVS 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 as described in *OS/390 MVS System Commands*.

## Environment

This command can be issued only from an MVS console.

**Data sharing scope**: Member

## Authorization

This command requires an appropriate level of MVS authority, as described in *OS/390 MVS System Commands*.

The syntax diagram and option descriptions for this command are purposely incomplete. Options that are not shown are described in *OS/390 MVS System Commands*.

## Syntax

```
                                    ┌─,WRAP───┐
►►──TRACE CT,────┬─WTRSTART=parmlibmem─┴─────────┴──────────┬───────────────────────────────►◄
                 │                  └─,NOWRAP─┘           │
                 ├─WTRSTOP=jobname────────────────────────┤
                 ├─ON,──COMP=irlmnm──┬──────────────────────┬┤
                 │                   └─,SUB=(─┬─DBM─┬─)─┘ │
                 │                            ├─EXP─┤     │
                 │                            ├─INT─┤     │
                 │                            ├─SLM─┤     │
                 │                            ├─XCF─┤     │
                 │                            └─XIT─┘     │
                 └─OFF────────────────────────────────────┘
```

## Option descriptions

**CT**

Specifies the component trace. (Do not use other trace options available on the MVS TRACE command).

**WTRSTART=***parmlibmem*

Identifies the member that contains source JCL. That JCL executes the CTRACE writer and defines the data set to which it writes the trace buffers. This member can be a procedure cataloged in SYS1.PROCLIB or a job.

**WRAP**

Specifies that when the system reaches the end of the group of data sets, it writes over the oldest data at the beginning of the first data set in the group. The system uses only the primary extents of the data sets.

**NOWRAP**
Specifies that the system stops writing to the data sets when they are all full. The system uses the primary and secondary extents of the data sets.

**WTRSTOP=***jobname*
Stops the CTRACE writer for a trace that is running. The system also closes the data sets that the writer used.

*jobname* identifies the trace, either by:

- Member name, if the source JCL is a procedure
- Job name, if that appears on a JOB statement in the source JCL

**ON**
Turns on the trace.

**COMP=***irlmssnm*
Gives the IRLM subsystem name.

**SUB=***subname*
Specifies the type of sublevel trace. Traces INT, EXP, and XIT are ON by default. You cannot turn off traces INT and EXP. If you do not specify a subname on the TRACE command, the trace is performed on all subnames that you control. Specifying one subname restricts the traces to that trace plus the EXP and INT traces.

| **Use:** | **To trace:** |
|---|---|
| DBM | Interactions with the identified DBMS |
| EXP | Any exception condition |
| INT | Member and group events outside normal locking activity |
| SLM | Interactions with the MVS locking component |
| XCF | All interactions with MVS cross-system coupling services |
| XIT | Only asynchronous interactions with the MVS 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 MVS link list:*** This command uses MVS component trace services. Include the IRLM load module DXRRL183, which contains a routine for stopping and starting, in the MVS link list.

***Displaying a trace:*** To display a trace, use the MVS DISPLAY command:

```
# D TRACE,COMP=IRLM001
```

***Monitoring a trace:*** To monitor a trace, use the MVS MODIFY irlmproc,STATUS,TRACE command on page 236.

***Setting the number of trace buffers:*** To set the number of trace buffers used by traces, use the MVS MODIFY irlmproc,SET command on page 234.

***Sample procedure for the CTRACE writer:*** This procedure identifies the data set to which the next sample procedure writes data.

```
//CTWTR    PROC
//         EXEC PGM=ITTTRCWR
//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:*** After you enter the command TRACE CT,WTRSTART, turn the trace on and connect the writer, using the WTR parameter in the reply for the command TRACE CT.

```
TRACE CT,WTRSTART=CTWTR
TRACE CT,ON,COMP=IRLM001,SUB=(DBM)
⋮
  (MVS asks for a reply)
⋮
R 15,WTR=CTWTR,END
TRACE CT,OFF,COMP=IRLM001,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=IRLM001
⋮
  (MVS asks for a reply)
⋮
R 15,END
⋮
TRACE CT,OFF,COMP=IRLM001
```

***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.

# Appendix A. 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:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

**The following paragraph does not apply to the United Kingdom or any other country 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 publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is as your own risk.

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

## Programming interface information

This book is intended to help you to use commands of IBM DATABASE 2 Universal Database Server for OS/390 (DB2 for OS/390) and related subsystems. This book primarily documents General-use Programming Interface and Associated Guidance Information provided by DB2.

General-use programming interfaces allow the customer to write programs that obtain the services of DB2.

However, this book also documents 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 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 marking:

Product-sensitive Programming Interface

Product-sensitive Programming Interface and Associated Guidance Information ...

End of Product-sensitive Programming Interface

## Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, or other countries, or both:

| | |
|---|---|
| AD/Cycle | Language Environment |
| AIX | MVS/ESA |
| APL2 | MVS/XA |
| BookManager | Net.Data |
| CICS | OS/2 |
| CICS/ESA | OS/390 |
| CICS/MVS | OS/400 |
| CICS OS/2 | Parallel Sysplex |
| COBOL/370 | PR/SM |
| C/370 | QMF |
| DATABASE 2 | RACF |
| DataPropagator | SQL/DS |
| DB2 | System Object Model |
| DB2 Universal Database | VTAM |
| DRDA | |
| IBM | |
| IMS | |
| IMS/ESA | |

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft™, Windows™, Windows NT, and the Windows logo are trademarks or registered trademarks of Microsoft, Inc. in the United States and/or other countries.

Other company, product, and service names may be trademarks or service marks of others.

# Glossary

The following terms and abbreviations are defined as they are used in the DB2 library. If you do not find the term you are looking for, refer to the index or to *IBM Dictionary of Computing*.

## A

**abend**. Abnormal end of task.

**abend reason code**. A 4-byte hexadecimal code that uniquely identifies a problem with DB2. A complete list of DB2 abend reason codes and their explanations is contained in *DB2 Messages and Codes*.

**access method services**. The facility that is used to define 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.

**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, whereas the archive log holds those records that are older and no longer fit on the active log.

**active member state**. A state of a member of a data sharing group. An active member is identified with a group by XCF, which associates the member with a particular task, address space, and MVS system. A member that is not active has either a failed member state or a quiesced member state.

**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. Each address space that contains a task that is connected to DB2 has exactly one address space connection, even though more than one task control block (TCB) can be present. See also *allied address space* and *task control block*.

**agent**. As used in DB2, the structure that associates all processes that are involved in a DB2 unit of work. An *allied agent* is generally synonymous with an *allied thread*. *System agents* are units of work that process independently of the allied agent, such as prefetch processing, deferred writes, and service tasks.

**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.

**allied address space**. An area of storage that is external to DB2 and that is connected to DB2. An allied address space is capable of requesting DB2 services.

**allied thread**. A thread that originates at the local DB2 subsystem and that can access data at a remote DB2 subsystem.

**ambiguous cursor**. A database cursor that is not defined with the FOR FETCH ONLY clause or the FOR UPDATE OF clause, is not defined on a read-only result table, is not the target of a WHERE CURRENT clause on an SQL UPDATE or DELETE statement, and is in a plan or package that contains either PREPARE or EXECUTE IMMEDIATE SQL statements.

**American National Standards Institute (ANSI)**. An organization consisting of producers, consumers, and general interest groups, that establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States.

**ANSI**. American National Standards Institute.

**API**. 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 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 orderable 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 server (AS)**. See *server*.

**archive log**. The portion of the DB2 log that contains log records that have been copied from the active log.

**AS**. Application server. See *server*.

**ASCII**. An encoding scheme that is used to represent strings in many environments, typically on PCs and workstations. Contrast with *EBCDIC*.

**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 one of that employee's attributes.

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

**auxiliary index**. An index on an auxiliary table in which each index entry refers to a LOB.

**auxiliary table**. A table that stores columns outside the table in which they are defined. Contrast with *base table*.

## B

**backward log recovery**. The fourth and 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**. (1) A table that is created by the SQL CREATE TABLE statement and that holds persistent data. Contrast with *result table* and *temporary table*.

(2) A table containing a LOB column definition. The actual LOB column data is not stored with the base table. The base table contains a row identifier for each row and an indicator column for each of its LOB columns. Contrast with *auxiliary table*.

**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.

**binary integer**. A basic data type that can be further classified as small integer or large integer.

**binary large object (BLOB)**. A sequence of bytes, where the size of the value ranges from 0 bytes to 2 GB - 1. Such a string does not have an associated CCSID.

**bind**. The process by which the output from the DB2 precompiler is converted to a usable control structure (which is called a package or an application plan). During the process, access paths to the data are selected and some authorization checking is performed.

**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.
**dynamic bind**. A process by which SQL statements are bound as they are entered.
**incremental bind**. A process by which SQL statements are bound during the execution of an application process, because they could not be bound during the bind process, and VALIDATE(RUN) was specified.
**static bind**. A process by which SQL statements are bound after they have been precompiled. All static SQL statements are prepared for execution at the same time.

**BLOB**. Binary large object.

**BMP**. Batch Message Processing (IMS).

**bootstrap data set (BSDS)**. A VSAM data set that contains name and status information for DB2, as well as RBA range specifications, for all active and archive log data sets. It also contains passwords for the DB2 directory and catalog, and lists of conditional restart and checkpoint records.

**BSAM**. Basic sequential access method.

**BSDS**. Bootstrap data set.

**buffer pool**. Main storage that is reserved to satisfy the buffering requirements for one or more table spaces or indexes.

**built-in function**. A function that DB2 supplies. Contrast with *user-defined function*.

## C

**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**. Call attachment facility.

**call attachment facility (CAF)**. A DB2 attachment facility for application programs that run in TSO or MVS batch. The CAF is an alternative to the DSN command processor and provides greater control over the execution environment.

**cascade delete**. The way in which DB2 enforces referential constraints when it deletes all descendent rows of a deleted parent row.

**cast function**. A function that is used to convert instances of a (source) data type into instances of a different (target) data type. In general, a cast function has the name of the target data type. It has one single argument whose type is the source data type; its return type is the target data type.

**castout**. The DB2 process of writing changed pages from a group buffer pool to DASD.

**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**. Coded character set identifier.

**CDB**. Communications database.

**CFRM policy**. A declaration by an MVS administrator regarding the allocation rules for a coupling facility structure.

**character large object (CLOB)**. A sequence of bytes representing single-byte characters or a mixture of single- and double-byte characters where the size of the value can be up to 2 GB - 1. In general, character large object 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- and double-byte characters.

**CHECK clause**. An extension to the SQL CREATE TABLE and SQL ALTER TABLE statements that specifies a table check constraint. See also *table check constraint*.

**check constraint**. See *table check constraint*.

**check integrity**. The condition that exists when each row in a table conforms to the table check constraints that are defined on that table. Maintaining check integrity requires DB2 to enforce table check constraints on operations that add or change data.

**check pending**. A state of a table space or partition that prevents its use by some utilities and some SQL statements because of rows that violate referential constraints, table check constraints, or both.

**checkpoint**. A point at which DB2 records internal status information on the DB2 log; the recovery process uses this information if DB2 abnormally terminates.

**CI**. Control interval.

**CICS**. Represents (in this publication) one of the following products:

> **CICS Transaction Server for OS/390**: Customer Information Control Center Transaction Server for OS/390
> **CICS/ESA**: Customer Information Control System/Enterprise Systems Architecture
> **CICS/MVS**: Customer Information Control System/Multiple Virtual Storage

**CICS attachment facility**. A DB2 subcomponent that uses the MVS subsystem interface (SSI) and cross storage linkage to process requests from CICS to DB2 and to coordinate resource commitment.

**CIDF**. Control interval definition field.

**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:

> Cursor stability (CS)
> Repeatable read (RR)
> Write

**claim count**. A count of the number of agents that are accessing an object.

**clause**. In SQL, a distinct part of a statement, such as a SELECT clause or a WHERE clause.

**CLIST**. Command list. A language for performing TSO tasks.

**CLOB**. Character large object.

**clustering index**. An index that determines how rows are physically ordered in a table space.

**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 consisting of a character set identifier and an associated code page identifier.

**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**. An SQL operation that derives its result from a collection of values across one or more rows. Contrast with *scalar function*.

**command**. A DB2 operator command or a DSN subcommand. A command is distinct from an SQL statement.

**command prefix**. A one- to eight-character command identifier. The command prefix distinguishes the command as belonging to an application or subsystem rather than to MVS.

**command recognition character (CRC)**. A character that permits an MVS 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. If a command has *member scope*, the command displays information only from the one member or affects only non-shared resources that are owned locally by that member. If a command has *group scope*, the command displays information from all members, affects non-shared resources that are owned locally by all members, displays information on sharable resources, or affects sharable resources.

**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.

**commit point**. A point in time when data is considered consistent.

**committed phase**. The second phase of the multi-site update process that requests all participants to commit the effects of the logical unit of work.

**communications database (CDB)**. A set of tables in the DB2 catalog that are used to establish conversations with remote database management systems.

**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 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*, *table check constraint*, and *uniqueness constraint*.

**control interval (CI)**. A fixed-length area or direct access storage in which VSAM stores records and creates distributed free space. Also, in a key-sequenced data set or file, the set of records pointed to by an entry in the sequence-set index record. The control interval is the unit of information that VSAM transmits to or from direct access storage. A control interval always includes an integral number of physical records.

**control interval definition field (CIDF)**. In VSAM, a field located in the 4 bytes at the end of each control interval; it describes the free space, if any, in the control interval.

**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.

**coordinator**. 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.

**correlation ID**. An identifier that is associated with a specific thread. In TSO, it is either an authorization ID or the job name.

**coupling facility**. A special PR/SM™ LPAR logical partition that runs the coupling facility control program and provides high-speed caching, list processing, and locking functions in a Sysplex.

**CRC**. Command recognition character.

**CRCR**. Conditional restart control record. See also *conditional restart*.

# **created temporary table**. A 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,
# so this kind of table is persistent and can be shared
# across application processes. Contrast with *declared*
# *temporary table*. See also *temporary table*.

**cross-system coupling facility (XCF)**. A component of MVS that provides functions to support cooperation between authorized programs running within a Sysplex.

**CS**. Cursor stability.

**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 named control structure that an application program uses to point to a row of interest within some set of rows, and to retrieve rows from the set, possibly making updates or deletions.

**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.

**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.

## D

**DASD**. Direct access storage device.

**database**. A collection of tables, or a collection of table spaces and index spaces.

**database access thread**. 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 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, and relationships.

**database management system (DBMS)**. A software system that controls the creation, organization, and modification of a database and the access to the data 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 2 Interactive (DB2I)**. The DB2 facility that provides for the execution of SQL statements, DB2 (operator) commands, programmer commands, and utility invocation.

**data currency**. The state in which data that is retrieved into a host variable in your program is a copy of data in the base table.

**data definition name (ddname)**. The name of a data definition (DD) statement that corresponds to a data control block containing the same name.

**Data Language/I (DL/I)**. The IMS data manipulation language; a common high-level interface between a user application and IMS.

**data partition**. A VSAM data set that is contained within a partitioned table space.

**data sharing**. The ability of two or more DB2 subsystems to directly access and change a single set of data.

**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 space**. A range of up to 2 GB of contiguous virtual storage addresses that a program can directly manipulate. Unlike an address space, a data space can hold only data; it does not contain common areas, system data, or programs.

**data type**. An attribute of columns, literals, host variables, special registers, and the results of functions and expressions.

**date**. A three-part value that designates a day, month, and year.

**date duration**. A decimal integer that represents a number of years, months, and days.

**DBA**. Database administrator.

**DBCLOB**. Double-byte character large object.

**DBCS**. Double-byte character set.

**DBD**. Database descriptor.

**DBID**. Database identifier.

**DBMS**. Database management system.

**DBRM**. Database request module.

**DB2 catalog**. Tables that are maintained by DB2 and that contain descriptions of DB2 objects, such as tables, views, and indexes.

**DB2 command**. An instruction to the DB2 subsystem allowing a user 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.

**DB2 for VSE & VM**. The IBM DB2 relational database management system for the VSE and VM operating systems.

**DB2I**. DATABASE 2 Interactive.

**DCLGEN**. Declarations generator.

**DDF**. Distributed data facility.

**ddname**. Data definition name.

**deadlock**. Unresolvable 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. DCLGEN is also a DSN subcommand.

# **declared temporary table**. A table that holds tempo-
# rary data and is defined with the SQL statement
# DECLARE GLOBAL TEMPORARY TABLE. Information
# about declared temporary tables is not stored in the
# DB2 catalog, so this kind of table is not persistent and
# can only be used by the application process that issued
# the DECLARE statement. Contrast with *created tempo-
# rary table*. See also *temporary table*.

**default value**. A predetermined value, attribute, or option that is assumed when no other is explicitly specified.

**delimited identifier**. A sequence of characters that are enclosed within double quotation marks ("). The sequence must consist of a letter followed by zero or more characters, each of which is a letter, digit, or the underscore character (_).

**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 *parent row*, *parent table*, *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.

**direct access storage device (DASD)**. A device in which access time is independent of the location of the data.

**directory**. The DB2 system database that contains internal objects such as database descriptors and skeleton cursor tables.

**distinct type**. A user-defined data type that is internally represented as an existing type (its source type), but is considered to be a separate and incompatible type for semantic purposes.

**distributed data facility (DDF)**. A set of DB2 components through which DB2 communicates with another RDBMS.

**Distributed Relational Database Architecture (DRDA)**. A connection protocol for distributed relational database processing that is used by IBM's 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.

**DL/I**. Data Language/I.

**double-byte character large object (DBCLOB)**. A sequence of bytes representing double-byte characters where the size of the values can be up to 2 GB. In general, double-byte character large object values are used whenever a double-byte character string might exceed the limits of the VARGRAPHIC 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 and therefore requires special hardware to be displayed or printed. Contrast with *single-byte character set*.

**double-precision floating point number**. A 64-bit approximate representation of a real number.

**drain**. The act of acquiring a locked resource by quiescing access to that object.

**drain lock**. A lock on a claim class that prevents a claim from occurring.

**DRDA**. Distributed Relational Database Architecture.

**DRDA access**. A method of accessing distributed data by which you can connect to another location, using an SQL statement, to execute packages that have been previously bound at that location. The SQL CONNECT or three-part name statement is used to identify application servers, and SQL statements are executed using packages that were previously bound at those servers. Contrast with *private protocol access*.

**DSN**. (1) The default DB2 subsystem name. (2) The name of the TSO command processor of DB2. (3) The first three characters of DB2 module and macro names.

**duration**. A number that represents an interval of time. See *date duration*, *labeled duration*, and *time duration*.

**dynamic SQL**. SQL statements that are prepared and executed within an application program while the program is executing. In dynamic SQL, the SQL source is contained in host language variables rather than being coded into the application program. The SQL statement can change several times during the application program's execution.

## E

**EBCDIC**. Extended binary coded decimal interchange code. An encoding scheme that is used to represent character data in the OS/390, MVS, VM, VSE, and OS/400® environments. Contrast with *ASCII*.

**EDM pool**. A pool of main storage that is used for database descriptors, application plans, authorization cache, application packages, and dynamic statement caching.

**embedded SQL**. SQL statements that are coded within an application program. See *static SQL*.

**escape character**. The symbol that is used to enclose an SQL delimited identifier. The escape character is the double quotation mark ("), except in COBOL applications, where the user assigns the symbol, which is either a double quotation mark or an apostrophe (').

**ESDS**. Entry sequenced data set.

**EUR**. IBM European Standards.

**exception table**. A table that holds rows that violate referential constraints or table check constraints that the CHECK DATA utility finds.

**exclusive lock**. A lock that prevents concurrently executing application processes from reading or changing data. Contrast with *shared lock*.

**executable statement**. An SQL statement that can be embedded in an application program, dynamically prepared and executed, or issued interactively.

**exit routine**. A user-written (or IBM-provided default) program that receives control from DB2 to perform specific functions. Exit routines run as extensions of DB2.

**explicit hierarchical locking**. 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.

**external function**. A function for which the body is written in a programming language that takes scalar argument values and produces a scalar result for each invocation. Contrast with *sourced function* and *built-in function*.

## F

**failed member state**. A state of a member of a data sharing group. When a member fails, the XCF permanently records the failed member state. This state usually means that the member's task, address space, or MVS system terminated before the state changed from active to quiesced.

**fallback**. The process of returning to a previous release of DB2 after attempting or completing migration to a current release.

**false global lock contention**. A contention indication from the coupling facility when multiple lock names are hashed to the same indicator and when no real contention exists.

**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.

**fixed-length string**. A character or graphic string whose length is specified and cannot be changed. Contrast with *varying-length string*.

**foreign key**. A key that is specified in the definition of a referential constraint. Because of the foreign key, the table is a dependent table. The key must have the same number of columns, with the same descriptions, as the primary key of the parent table.

**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.

**function**. A specific purpose of an entity or its characteristic action such as a column function or scalar function. (See also *column function* and *scalar function*.)

Functions can be user-defined, built-in, or generated by DB2. (See *built-in function*, *cast function*, *external function*, *sourced function*, and *user-defined function*.)

**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.

## G

**GB**. Gigabyte (1 073 741 824 bytes).

**GBP**. 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 DASD.

**generalized trace facility (GTF)**. An MVS service program that records significant system events such as I/O interrupts, SVC interrupts, program interrupts, or external interrupts.

**getpage**. An operation in which DB2 accesses a data page.

**global lock**. A lock that provides concurrency control within and among DB2 subsystems. The scope of the lock is across all the DB2 subsystems of a data sharing group.

**global lock contention**. Conflicts on locking requests between different DB2 members of a data sharing group when those members are trying to serialize shared resources.

**governor**. See *resource limit facility*.

**gross lock**. The *shared*, *update*, or *exclusive* mode locks on a table, partition, or table space.

**group buffer pool (GBP)**. A coupling facility cache structure that is used by a data sharing group to cache data and to ensure that the data is consistent for all members.

**group buffer pool duplexing**. The ability to write data to two instances of a group buffer pool structure: a *primary group buffer pool* and a *secondary group buffer pool*. OS/390 publications refer to these instances as the 'old' (for primary) and 'new' (for secondary) structures.

**group name**. The MVS XCF identifier for a data sharing group.

**group restart**. A restart of at least one member of a data sharing group after the loss of either locks or the shared communications area.

**GTF**. Generalized trace facility.

## H

**help panel**. A screen of information presenting tutorial text to assist a user at the terminal.

**hiperspace**. A range of up to 2 GB of contiguous virtual storage addresses that a program can use as a buffer. Like a data space, a hiperspace can hold user data; it does not contain common areas or system data. Unlike an address space or a data space, data in a hiperspace is not directly addressable. To manipulate data in a hiperspace, you bring the data into the address space in 4-KB blocks.

**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.

**HSM**. Hierarchical storage manager.

## I

**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.

**identify**. A request that an attachment service program in an address space that is separate from DB2 issues via the MVS subsystem interface to inform DB2 of its existence and to initiate the process of becoming connected to DB2.

**IFCID**. Instrumentation facility component identifier.

**IFI**. 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**. Information Management System.

**IMS attachment facility**. A DB2 subcomponent that uses MVS 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*.

**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 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.

**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 emergency 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 of class resources or attributes from a parent class downstream in the class hierarchy to a child class.

**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.

**internal resource lock manager (IRLM)**. An MVS subsystem that DB2 uses to control communication and database locking.

**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.

**IRLM**. Internal resource lock manager.

**ISO**. International Standards Organization.

**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**. Interactive System Productivity Facility.

**ISPF/PDF**. Interactive System Productivity Facility/Program Development Facility.

## J

**Japanese Industrial Standards Committee (JISC)**. An organization that issues standards for coding character sets.

**JCL**. Job control language.

**JES**. MVS Job Entry Subsystem.

**JIS**. Japanese Industrial Standard.

**job control language (JCL)**. A control language that is used to identify a job to an operating system and to describe the job's requirements.

**Job Entry Subsystem (JES)**. An IBM licensed program that receives jobs into the system and processes all output data that is produced by the jobs.

## K

**KB**. Kilobyte (1024 bytes).

**key**. A column or an ordered collection of columns identified in the description of a table, index, or referential constraint.

**key-sequenced data set (KSDS)**. A VSAM file or data set whose records are loaded in key sequence and controlled by an index.

**KSDS**. Key-sequenced data set.

## L

**labeled duration**. A number that represents a duration of years, months, days, hours, minutes, seconds, or microseconds.

**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 - 1 byte in length. See also *BLOB*, *CLOB*, and *DBCLOB*.

**leaf page**. A page that contains pairs of keys and RIDs and that points to actual data. Contrast with *nonleaf page*.

**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.

**L-lock**. Logical lock.

**load module**. A program unit that is suitable for loading into main storage for execution. The output of a linkage editor.

**LOB**. Large object.

**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*.

**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 RDBMS to which the user or application program is directly connected (in the case of DB2, by one of the DB2 attachment facilities).

**location name**. The name by which DB2 refers to a particular DB2 subsystem in a network of subsystems. Contrast with *LU name*.

**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.

**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 parent**. For explicit hierarchical locking, a lock that is held on a resource that has child locks that are lower in the hierarchy; usually the table space or partition intent locks are the parent locks.

**lock promotion**. The process of changing the size or mode of a DB2 lock to a higher level.

**lock size**. The amount of data 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.

**logical claim**. A claim on a logical partition of a non-partitioning index.

**logical drain**. A drain on a logical partition of a non-partitioning 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 concurrency between transactions. Contrast with *P-lock*.

**logical page list (LPL)**. A list of pages that are in error and that cannot be referenced by applications until the pages are recovered. The page is in *logical error*, because the actual media (coupling facility or DASD) might not contain any errors. Usually a connection to the media has been lost.

**logical partition**. A set of key or RID pairs in a non-partitioning index that are associated with a particular partition.

**logical unit**. An access point through which an application program accesses the SNA network in order to communicate with another application program.

**logical unit of work (LUW)**. The processing that a program performs between synchronization points.

**logical unit of work identifier (LUWID)**. 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.

**log initialization**. The first phase of restart processing during which DB2 attempts to locate the current end of the log.

**log record sequence number (LRSN)**. A number that DB2 generates and associates with each log record. DB2 also uses the LRSN for page versioning. The LRSNs that a particular DB2 data sharing group generates form a strictly increasing sequence for each DB2 log and a strictly increasing sequence for each page across the DB2 group.

**log truncation**. 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.

**LPL**. Logical page list.

**LRH**. Log record header.

**LRSN**. Log record sequence number.

**LU name**. Logical unit name, which is the name by which VTAM refers to a node in a network. Contrast with *location name*.

**LUW**. Logical unit of work.

**LUWID**. Logical unit of work identifier.

## M

**MB**. Megabyte (1 048 576 bytes).

**member name**. The MVS XCF identifier for a particular DB2 subsystem in a data sharing group.

**menu**. A displayed list of available functions for selection by the operator. A menu is sometimes called a *menu panel*.

**migration**. The process of converting a DB2 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 you created on the previous release.

**mixed data string**. A character string that can contain both single-byte and double-byte characters.

**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 fails, that DB2 subsystem's modify locks are converted to retained locks.

**MPP**. Message processing program (IMS).

**MTO**. Master terminal operator.

**multi-site update**. Distributed relational database processing in which data is updated in more than one location within a single unit of work.

**MVS**. Multiple Virtual Storage.

**MVS/ESA**. Multiple Virtual Storage/Enterprise Systems Architecture.

**MVS/XA**. Multiple Virtual Storage/Extended Architecture.

## N

**negotiable lock**. A lock whose mode can be downgraded, by agreement among contending users, to be compatible to all. A physical lock is an example of a negotiable lock.

**network identifier (NID)**. The network ID that is assigned by IMS or CICS, or if the connection type is RRSAF, the OS/390 RRS Unit of Recovery ID (URID).

**NID**. Network ID.

**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.

**nonpartitioning index**. Any index that is not a partitioning index.

**NUL**. In C, a single character that denotes the end of the string.

**null**. A special value that indicates the absence of information.

**NUL-terminated host variable**. A varying-length host variable in which the end of the data is indicated by the presence of a NUL terminator.

**NUL terminator**. In C, the value that indicates the end of a string. For character strings, the NUL terminator is `X'00'`.

## O

**OASN (origin application schedule number)**. In IMS, a 4-byte number that is assigned sequentially to each IMS schedule since the last cold start of IMS. The OASN is used as an identifier for a unit of work. In an 8-byte format, the first 4 bytes contain the schedule number and the last 4 bytes contain the number of IMS sync points (*commit points*) during the current schedule. The OASN is part of the NID for an IMS connection.

**OBID**. Data object identifier.

**OS/390**. Operating System/390.

## P

**package**. An object containing a set of SQL statements that have been bound statically 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.

**page**. A unit of storage within a table space (4 KB, 8 KB, 16 KB, or 32 KB) or index space (4 KB). In a table space, a page contains one or more rows of a table. In a LOB table space, a LOB value can span more than one page, but no more than one LOB 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 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.

**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. Partitions can be extended to a maximum size of 1, 2, or 4 GB, depending on the number of partitions in the partitioned page set. All partitions of a given page set have the same maximum size.

**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 subdivided into parts (based on index key range), each of which can be processed independently by utilities.

**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*.

**piece**. A data set of a nonpartitioned page set.

**physical claim**. A claim on an entire nonpartitioning index.

**physical drain**. A drain on an entire nonpartitioning index.

**physical lock (P-lock)**. A lock type 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)*.

**physical lock contention**. Conflicting states of the requesters for a physical lock. See *negotiable lock*.

**plan**. See *application plan*.

**plan allocation**. The process of allocating DB2 resources to a plan in preparation to execute it.

**plan name**. The name of an application plan.

**P-lock**. 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 inflight or in-abort, was interrupted by system failure or cancellation, and did not complete backout during restart.

**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.

**prefix**. A code at the beginning of a message or record.

**prepare**. The first phase of a two-phase commit process in which all participants are requested to prepare for commit.

**primary authorization ID**. The authorization ID used to identify the application process to DB2.

**primary group buffer pool**. For a duplexed group buffer pool, the structure used to maintain the coherency of cached data. This structure is used for page registration and cross-invalidation. The OS/390 equivalent is *old* structure. Compare with *secondary group buffer pool*.

**primary index**. An index that enforces the uniqueness of a primary key.

**private connection**. A communications connection that is specific to DB2.

**private protocol access**. A method of accessing distributed data by which you can direct a query to another DB2 system. Contrast with *DRDA access*.

**private protocol connection**. A DB2 private connection of the application process. See also *private connection*.

**privilege**. The capability of performing a specific function, sometimes on a specific object. The term includes:

> **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 one owns, or the holding of an authority, such as the privilege of SYSADM authority to terminate any utility job.

**privilege set**. For the installation SYSADM ID, the set of all possible privileges. For any other 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.

**protected conversation**. A VTAM conversation that supports two-phase commit flows.

**PSRCP**. Page set recovery pending.

## Q

**QMF**. Query Management Facility.

**query**. A component of certain SQL statements that specifies a result table.

**quiesced member state**. A state of a member of a data sharing group. An active member becomes quiesced when a STOP DB2 command takes effect without a failure. If the member's task, address space, or MVS system fails before the command takes effect, the member state is failed.

## R

**RACF**. Resource Access Control Facility.

**RBA**. Relative byte address.

**RCT**. Resource control table (CICS attachment facility).

**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. Under level RS, an application that issues the same query more than once might read additional rows that were inserted and committed by a concurrently executing application process.

**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.

**record**. The storage representation of a row or other data.

**record identifier (RID) pool**. An area of main storage above the 16-MB line that is reserved for sorting record identifiers during list prefetch processing.

**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 pending (RECP)**. A condition that prevents SQL access to a table space that needs to be recovered.

**RECP**. Recovery pending.

**redo**. A state of a unit of recovery that indicates that changes are to be reapplied to the DASD media to ensure data integrity.

**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 integrity**. The condition that exists when all intended references from data in one column of a table to data in another column of the same or a different table are valid. Maintaining referential integrity requires that DB2 enforce referential constraints on all LOAD, RECOVER, INSERT, UPDATE, and DELETE operations.

**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.

**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 RDBMS, 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 MVS system.

**REORG pending (REORP)**. A condition that restricts SQL access and most utility access to an object that must be reorganized.

**REORP**. 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 referenced by the program cannot be changed by other programs until the program reaches a commit point.

**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.

**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 control table (RCT)**. A construct of the CICS attachment facility, created by site-provided macro parameters, that defines authorization and access attributes for transactions or transaction groups.

**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**. A site-defined table that specifies the limits to be enforced by the resource limit facility.

**restart pending (RESTP)**. A restrictive state of a page set or partition that indicates that restart (backout) work needs to be performed on the object. All access to the page set or partition is denied except for access by the:

- RECOVER POSTPONED command
- Automatic online backout (which DB2 invokes after restart if the system parameter LBACKOUT=AUTO)

**RESTP**. Restart pending.

**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 failure.

**RID pool**. Record identifier pool.

**RLF**. Resource limit facility.

**RMID**. Resource manager identifier.

**RO**. Read-only access.

**rollback**. The process of restoring data changed by SQL statements to the state at its last commit point. All locks are freed. Contrast with *commit*.

**ROWID**. Row identifier.

**row identifier (ROWID)**. A value that uniquely identifies a row. This value is stored with the row and never changes.

**RS**. Read stability.

## S

**SBCS**. Single-byte character set.

**SCA**. 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. Contrast with *column function*.

**schema**. A logical grouping for user-defined functions, distinct types, triggers, and stored procedures. When an object of one of these types is created, it is assigned to one schema, which is determined by the name of the object. For example, the following statement creates a distinct type T in schema C:

```
CREATE DISTINCT TYPE C.T ...
```

**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 OS/390 equivalent is *new* structure.

**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.

**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.

**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.

**server**. A functional unit that provides services to one or more clients over a network. In the DB2 environment, a server is the target for a request from a remote RDBMS and is the RDBMS that provides the data. A server is sometimes also called an *application server (AS)*.

**session**. A link between two nodes in a VTAM network.

**shared communications area (SCA)**. A coupling facility list structure that a DB2 data sharing group uses for inter-DB2 communication.

**shared 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 table space**. A table space that is neither partitioned nor segmented.

**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*.

**SMF**. System management facility.

**SMS**. Storage Management Subsystem.

**SNA**. Systems Network Architecture.

**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 a column (aggregating) function; it returns a single value from a set of values (for example, MAX or AVG). Contrast with *external function* and *built-in function*.

**source program**. A set of host language statements and SQL statements that is processed by an SQL precompiler.

**SPUFI**. SQL Processor Using File Input.

**SQL**. Structured Query Language.

**SQL authorization ID (SQL ID)**. The authorization ID that is used for checking dynamic SQL statements in some situations.

**SQL communication area (SQLCA)**. A structure that is used to provide an application program with information about the execution of its SQL statements.

**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 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 current path is found in the CURRENT PATH special register. In static SQL, it is defined in the PATH bind option.

**SQL Processor Using File Input (SPUFI)**. SQL Processor Using File Input. 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 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 (").

**SQLCA**. SQL communication area.

**SQLDA**. SQL descriptor area.

**SQL/DS**. Structured Query Language/Data System. This product is now obsolete and has been replaced by DB2 for VSE & VM.

**SSI**. Subsystem interface (MVS).

**SSM**. Subsystem member.

**stand-alone**. An attribute of a program that means it is capable of executing separately from DB2, without using DB2 services.

**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 host variables that are specified by the statement might change).

**storage group**. A named set of DASD volumes on which DB2 data can be stored.

**stored procedure**. A user-written application program, that can be invoked through the use of the SQL CALL statement.

**string**. See *character string* or *graphic string*.

**Structured Query Language (SQL)**. A standardized language for defining and manipulating data in a relational database.

**subsystem**. A distinct instance of a relational database management system (RDBMS).

**sync point**. See *commit point*.

**synonym**. In SQL, an alternative name for a table or view. Synonyms can only be used to refer to objects at the subsystem in which the synonym is defined.

**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 internally such as prefetch processing, deferred writes, and service tasks.

**system conversation**. The conversation that two DB2 subsystems must establish to process system messages before any distributed processing can begin.

**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.

## T

**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*.

**table check constraint**. A user-defined constraint that specifies the values that specific columns of a base table can contain.

**table space**. A page set that is used to store the records in one or more tables.

**table space set**. A set of table spaces and partitions that should be recovered together for one of these 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 control block that is used to communicate information about tasks within an address space that are connected to DB2. An address space can support many task connections (as many as one per task), but only one address space connection. See also *address space connection*.

**TCB**. Task control block (MVS).

# **temporary table**. A table that holds temporary data;
# for example, temporary tables are useful for holding or
# sorting intermediate results from queries that contain a
# large number of rows. The two kinds 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 tem-*
# *porary table* and *declared temporary table*.

**thread**. The DB2 structure that describes an application's connection, traces its progress, processes resource functions, and delimits its accessibility to DB2 resources and services. Most DB2 functions execute under a thread structure. See also *allied thread* and *database access thread*.

**three-part name**. The full name of a table, view, or alias. It consists of a location name, authorization ID, and an object name, separated by a period.

**time**. A three-part value that designates a time of day in hours, minutes, and seconds.

**time duration**. A decimal integer that represents a number of hours, minutes, and seconds.

**Time-Sharing Option (TSO)**. An option in MVS 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.

**TMP**. Terminal Monitor Program.

**trace**. A DB2 facility that provides the ability to monitor and collect DB2 monitoring, auditing, performance, accounting, statistics, and serviceability (global) data.

**transaction lock**. A lock that is used to control concurrent execution of SQL statements.

**TSO**. 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.

**type 1 indexes**. Indexes that were created by a release of DB2 before DB2 Version 4 or that are specified as type 1 indexes in Version 4. Contrast with *type 2 indexes*. As of Version 6, type 1 indexes are no longer supported.

**type 2 indexes**. Indexes that are created on a release of DB2 after Version 5 or that are specified as type 2 indexes in Version 4 or Version 5.

## U

**UDF**. User-defined function.

**UDT**. User-defined data type. In DB2 for OS/390, the term *distinct type* is used instead of user-defined function.

**uncommitted read (UR)**. The isolation level that allows an application to read uncommitted data.

**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.

**unique index**. An index which ensures that no identical key values are stored in a table.

**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.

**unit of recovery**. A recoverable sequence of operations within a single resource manager, such as an instance of DB2. Contrast with *unit of work*.

**unit of work**. 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 *multi-site update* operation, a single unit of work can include several *units of recovery*. Contrast with *unit of recovery*.

**UR**. Uncommitted read.

**URID (unit of recovery ID)**. The LOGRBA of the first log record for a unit of recovery. The URID also appears in all subsequent log records for that unit of recovery.

**user-defined data type (UDT)**. See *distinct type*.

**user-defined function (UDF)**. A function that is defined to DB2 using the CREATE FUNCTION statement and that can be referenced thereafter in SQL statements. A user-defined function can be either an *external function* or a *sourced function*. Contrast with *built-in function*.

**UT**. Utility-only access.

## V

**value**. The smallest unit of data that is manipulated in SQL.

**variable**. A data element that specifies a value that can be changed. A COBOL elementary data item is an example of a variable. Contrast with *constant*.

**varying-length string**. A character or graphic string whose length varies within set limits. Contrast with *fixed-length string*.

**version**. A member of a set of similar programs, DBRMs, packages, or LOBs.

**A version of a program** 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).
**A version of a DBRM** 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.
**A version of a package** is the result of binding a DBRM within a particular database system. The package version is identified by the same program name and consistency token as the DBRM.
**A version of a LOB** 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.

**view**. An alternative representation of data from one or more tables. A view can include all or some of the columns that are contained in tables on which it is defined.

**Virtual Storage Access Method (VSAM)**. An access method for direct or sequential processing of fixed- and varying-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative-record number.

**Virtual Telecommunications Access Method (VTAM)**. An IBM licensed program that controls communication and the flow of data in an SNA network.

**VSAM**. Virtual storage access method.

**VTAM**. Virtual Telecommunication Access Method (MVS).

## W

**warm start**. The normal DB2 restart process, which involves reading and processing log records so that data under the control of DB2 is consistent. Contrast with *cold start*.

## X

**XCF**. See *cross-system coupling facility*.

# Bibliography

**DB2 Universal Database Server for OS/390 Version 6 Product Libraries:**

***DB2 Universal Database for OS/390***

- *DB2 Administration Guide, SC26-9003*
- *DB2 Application Programming and SQL Guide, SC26-9004*
- *DB2 Application Programming Guide and Reference for Java™, SC26-9018*
- *DB2 ODBC Guide and Reference, SC26-9005*
- *DB2 Command Reference, SC26-9006*
- *DB2 Data Sharing: Planning and Administration, SC26-9007*
- *DB2 Data Sharing Quick Reference Card, SX26-3843*
- *DB2 Diagnosis Guide and Reference, LY36-3736*
- *DB2 Diagnostic Quick Reference Card, LY36-3737*
- *DB2 Image, Audio, and Video Extenders Administration and Programming, SC26-9650*
- *DB2 Installation Guide, GC26-9008*
- *DB2 Licensed Program Specifications, GC26-9009*
- *DB2 Messages and Codes, GC26-9011*
- *DB2 Master Index, SC26-9010*
- *DB2 Reference for Remote DRDA Requesters and Servers, SC26-9012*
- *DB2 Reference Summary, SX26-3844*
- *DB2 Release Planning Guide, SC26-9013*
- *DB2 SQL Reference, SC26-9014*
- *DB2 Text Extender Administration and Programming, SC26-9651*
- *DB2 Utility Guide and Reference, SC26-9015*
- *DB2 What's New? GC26-9017*
- *DB2 Program Directory, GI10-8182*

***DB2 Administration Tool***

- *DB2 Administration Tool for OS/390 User's Guide, SC26-9847*

***DB2 Buffer Pool Tool***

- *DB2 Buffer Pool Tool for OS/390 User's Guide and Reference, SC26-9306*

***DB2 DataPropagator***

- *DB2 Replication Guide and Reference, SC26-9642*

***Net.Data for OS/390***

The following books are available at
# *http://www.ibm.com/software/net.data/library.html:*

- *Net.Data Library: Administration and Programming Guide for OS/390*
- *Net.Data Library: Language Environment Interface Reference*
- *Net.Data Library: Messages and Codes*
- *Net.Data Library: Reference*

***DB2 PM for OS/390***

- *DB2 PM for OS/390 Batch User's Guide, SC26-9167*
- *DB2 PM for OS/390 Command Reference, SC26-9166*
- *DB2 PM for OS/390 General Information, GC26-9172*
- *DB2 PM for OS/390 Installation and Customization, SC26-9171*
- *DB2 PM for OS/390 Messages, SC26-9169*
- *DB2 PM for OS/390 Online Monitor User's Guide, SC26-9168*
- *DB2 PM for OS/390 Report Reference Volume 1, SC26-9164*
- *DB2 PM for OS/390 Report Reference Volume 2, SC26-9165*
- *DB2 PM for OS/390 Using the Workstation Online Monitor, SC26-9170*
- *DB2 PM for OS/390 Program Directory, GI10-8183*

***Query Management Facility***

- *Query Management Facility: Developing QMF Applications, SC26-9579*
- *Query Management Facility: Getting Started with QMF on Windows, SC26-9582*
- *Query Management Facility: High Peformance Option User's Guide for OS/390, SC26-9581*
- *Query Management Facility: Installing and Managing QMF on OS/390, GC26-9575*
- *Query Management Facility: Installing and Managing QMF on Windows, GC26-9583*
- *Query Management Facility: Introducing QMF, GC26-9576*
- *Query Management Facility: Messages and Codes, GC26-9580*
- *Query Management Facility: Reference, SC26-9577*
- *Query Management Facility: Using QMF, SC26-9578*

**Ada/370**

- *IBM Ada/370 Language Reference, SC09-1297*
- *IBM Ada/370 Programmer's Guide, SC09-1414*
- *IBM Ada/370 SQL Module Processor for DB2 Database Manager User's Guide, SC09-1450*

**APL2**

- *APL2 Programming Guide, SH21-1072*
- *APL2 Programming: Language Reference, SH21-1061*
- *APL2 Programming: Using Structured Query Language (SQL), SH21-1057*

**AS/400**

- *DB2 for OS/400 SQL Programming, SC41-4611*
- *DB2 for OS/400 SQL Reference, SC41-4612*

**BASIC**

- *IBM BASIC/MVS Language Reference, GC26-4026*
- *IBM BASIC/MVS Programming Guide, SC26-4027*

**BookManager READ/MVS**

- *BookManager READ/MVS V1R3: Installation Planning & Customization, SC38-2035*

**C/370**

- *IBM SAA AD/Cycle C/370 Programming Guide, SC09-1841*
- *IBM SAA AD/Cycle C/370 Programming Guide for Language Environment/370, SC09-1840*
- *IBM SAA AD/Cycle C/370 User's Guide, SC09-1763*
- *SAA CPI C Reference, SC09-1308*

**Character Data Representation Architecture**

- *Character Data Representation Architecture Overview, GC09-2207*
- *Character Data Representation Architecture Reference and Registry, SC09-2190*

**CICS/ESA**

- *CICS/ESA Application Programming Guide, SC33-1169*
- *CICS for MVS/ESA Application Programming Reference, SC33-1170*
- *CICS for MVS/ESA CICS-RACF Security Guide, SC33-1185*
- *CICS for MVS/ESA CICS-Supplied Transactions, SC33-1168*
- *CICS for MVS/ESA Customization Guide, SC33-1165*
- *CICS for MVS/ESA Data Areas, LY33-6083*
- *CICS for MVS/ESA Installation Guide, SC33-1163*
- *CICS for MVS/ESA Intercommunication Guide, SC33-1181*
- *CICS for MVS/ESA Messages and Codes, GC33-1177*
- *CICS for MVS/ESA Operations and Utilities Guide, SC33-1167*
- *CICS/ESA Performance Guide, SC33-1183*
- *CICS/ESA Problem Determination Guide, SC33-1176*
- *CICS for MVS/ESA Resource Definition Guide, SC33-1166*
- *CICS for MVS/ESA System Definition Guide, SC33-1164*
- *CICS for MVS/ESA System Programming Reference, GC33-1171*

**CICS/MVS**

- *CICS/MVS Application Programmer's Reference, SC33-0512*
- *CICS/MVS Facilities and Planning Guide, SC33-0504*
- *CICS/MVS Installation Guide, SC33-0506*
- *CICS/MVS Operations Guide, SC33-0510*
- *CICS/MVS Problem Determination Guide, SC33-0516*
- *CICS/MVS Resource Definition (Macro), SC33-0509*
- *CICS/MVS Resource Definition (Online), SC33-0508*

**IBM C/C++ for MVS/ESA**

- *IBM C/C++ for MVS/ESA Library Reference, SC09-1995*
- *IBM C/C++ for MVS/ESA Programming Guide, SC09-1994*

**IBM COBOL**

- *IBM COBOL Language Reference, SC26-4769*
- *IBM COBOL for MVS & VM Programming Guide, SC26-4767*

**Conversion Guide**

- *IMS-DB and DB2 Migration and Coexistence Guide, GH21-1083*

**Cooperative Development Environment**

- *CoOperative Development Environment/370: Debug Tool, SC09-1623*

**Data Extract (DXT)**

- *Data Extract Version 2: General Information, GC26-4666*
- *Data Extract Version 2: Planning and Administration Guide, SC26-4631*

**DataPropagator NonRelational**

- *DataPropagator NonRelational MVS/ESA Administration Guide, SH19-5036*
- *DataPropagator NonRelational MVS/ESA Reference, SH19-5039*

**Data Facility Data Set Services**

- *Data Facility Data Set Services: User's Guide and Reference, SC26-4388*

**Database Design**

- *DB2 Design and Development Guide, Gabrielle Wiorkowski and David Kull, Addison Wesley, ISBN 0-20158-049-8*
- *Handbook of Relational Database Design, C. Fleming and B. Von Halle, Addison Wesley, ISBN 0-20111-434-8*

**DataHub**

- *IBM DataHub General Information, GC26-4874*

**DB2 Connect**

- *DB2 Connect Enterprise Edition for OS/2 and Windows NT: Quick Beginnings, GC09-2828*
- *DB2 Connect Personal Edition Quick Beginnings, GC09-2830*
- *DB2 Connect User's Guide, SC09-2838*

**DB2 Server for VSE & VM**

- *DB2 Server for VM:  DBS Utility, SC09-2394*
- *DB2 Server for VSE: DBS Utility, SC09-2395*

**DB2 Universal Database (UDB)**

- *DB2 UDB Administration Guide Volume 1: Design and Implementation, SC09-2839*
- *DB2 UDB Administration Guide Volume 2: Performance, SC09-2840*
- *DB2 UDB Administrative API Reference, SC09-2841*
- *DB2 UDB Application Building Guide, SC09-2842*
- *DB2 UDB Application Development Guide, SC09-2845*
- *DB2 UDB Call Level Interface Guide and Reference, SC09-2843*
- *DB2 UDB SQL Getting Started, SC09-2856*
- *DB2 UDB SQL Reference Volume 1, SC09-2847*
- *DB2 UDB SQL Reference Volume 2, SC09-2848*

**Device Support Facilities**

- *Device Support Facilities User's Guide and Reference, GC35-0033*

**DFSMS/MVS**

- *DFSMS/MVS: Access Method Services for the Integrated Catalog, SC26-4906*
- *DFSMS/MVS: Access Method Services for VSAM Catalogs, SC26-4905*
- *DFSMS/MVS: Administration Reference for DFSMSdss, SC26-4929*
- *DFSMS/MVS: DFSMShsm Managing Your Own Data, SH21-1077*
- *DFSMS/MVS: Diagnosis Reference for DFSMSdfp, LY27-9606*
- *DFSMS/MVS Storage Management Library: Implementing System-Managed Storage, SC26–3123*
- *DFSMS/MVS: Macro Instructions for Data Sets, SC26-4913*
- *DFSMS/MVS: Managing Catalogs, SC26-4914*
- *DFSMS/MVS: Program Management, SC26-4916*
- *DFSMS/MVS: Storage Administration Reference for DFSMSdfp, SC26-4920*
- *DFSMS/MVS: Using Advanced Services, SC26-4921*
- *DFSMS/MVS: Utilities, SC26-4926*
- *MVS/DFP: Using Data Sets, SC26-4749*

**DFSORT**

- *DFSORT Application Programming: Guide, SC33-4035*

**Distributed Relational Database**

- *Data Stream and OPA Reference, SC31-6806*
- *IBM SQL Reference, SC26-8416*
- *Open Group Technical Standard (the Open Group presently makes the following books available through its Web site at http://www.opengroup.org):*
  - *DRDA Volume 1: Distributed Relational Database Architecture (DRDA), ISBN 1-85912-295-7*
  - *DRDA Version 2 Volume 2: Formatted Data Object Content Architecture, available only on Web* #
  - *DRDA Volume 3: Distributed Database Management (DDM) Architecture, ISBN 1-85912-206-X*

**Domain Name System**

- *DNS and BIND, Third Edition, Paul Albitz and Cricket Liu, O'Reilly, SR23-8771*

**Education**

- *IBM Dictionary of Computing, McGraw-Hill, ISBN 0-07031-489-6*
- *1999 IBM All-in-One Education and Training Catalog, GR23-8105*

**Enterprise System/9000 and Enterprise System/3090**

- *Enterprise System/9000 and Enterprise System/3090 Processor Resource/System Manager Planning Guide, GA22-7123*

**High Level Assembler**

- *High Level Assembler for MVS and VM and VSE Language Reference, SC26-4940*
- *High Level Assembler for MVS and VM and VSE Programmer's Guide, SC26-4941*

**Parallel Sysplex Library**

- *OS/390 Parallel Sysplex Application Migration, GC28-1863*
- *System/390 MVS Sysplex Hardware and Software Migration, GC28-1862*
- *OS/390 Parallel Sysplex Overview: An Introduction to Data Sharing and Parallelism, GC28-1860*
- *OS/390 Parallel Sysplex Systems Management, GC28-1861*
- *OS/390 Parallel Sysplex Test Report, GC28-1963*
- *System/390 9672/9674 System Overview, GA22-7148*

**ICSF/MVS**

- *ICSF/MVS General Information, GC23-0093*

**IMS/ESA**

- *IMS Batch Terminal Simulator General Information, GH20-5522*
- *IMS/ESA Administration Guide: System, SC26-8013*
- *IMS/ESA Administration Guide: Transaction Manager, SC26-8731*
- *IMS/ESA Application Programming: Database Manager, SC26-8727*
- *IMS/ESA Application Programming: Design Guide, SC26-8016*
- *IMS/ESA Application Programming: Transaction Manager, SC26-8729*
- *IMS/ESA Customization Guide, SC26-8020*
- *IMS/ESA Installation Volume 1: Installation and Verification, SC26-8023*
- *IMS/ESA Installation Volume 2: System Definition and Tailoring, SC26-8024*
- *IMS/ESA Messages and Codes, SC26-8028*
- *IMS/ESA Operator's Reference, SC26-8030*
- *IMS/ESA Utilities Reference: System, SC26-8035*

**ISPF**

- *ISPF V4 Dialog Developer's Guide and Reference, SC34-4486*
- *ISPF V4 Messages and Codes, SC34-4450*
- *ISPF V4 Planning and Customizing, SC34-4443*
- *ISPF V4 User's Guide, SC34-4484*

**Language Environment**

- *Debug Tool User's Guide and Reference, SC09-2137*

**National Language Support**

- *National Language Support Reference Volume 2, SE09-8002*

**NetView**

- *NetView Installation and Administration Guide, SC31-8043*
- *NetView User's Guide, SC31-8056*

**ODBC**

- *Microsoft ODBC 3.0 Programmer's Reference and SDK Guide, Microsoft Press, ISBN 1-55615-658-8*

**OS/390**

- *OS/390 C/C++ Programming Guide, SC09-2362*
- *OS/390 C/C++ Run-Time Library Reference, SC28-1663*
- *OS/390 C/C++ User's Guide, SC09-2361*
- *OS/390 eNetwork Communications Server: IP Configuration, SC31-8513*
- *OS/390 Hardware Configuration Definition Planning, GC28-1750*
- *OS/390 Information Roadmap, GC28-1727*
- *OS/390 Introduction and Release Guide, GC28-1725*
- *OS/390 JES2 Initialization and Tuning Guide, SC28-1791*
- *OS/390 JES3 Initialization and Tuning Guide, SC28-1802*
- *OS/390 Language Environment for OS/390 & VM Concepts Guide, GC28-1945*
- *OS/390 Language Environment for OS/390 & VM Customization, SC28-1941*
- *OS/390 Language Environment for OS/390 & VM Debugging Guide, SC28-1942*
- *OS/390 Language Environment for OS/390 & VM Programming Guide, SC28-1939*
- *OS/390 Language Environment for OS/390 & VM Programming Reference, SC28-1940*
- *OS/390 MVS Diagnosis: Procedures, LY28-1082*
- *OS/390 MVS Diagnosis: Reference, SY28-1084*
- *OS/390 MVS Diagnosis: Tools and Service Aids, LY28-1085*
- *OS/390 MVS Initialization and Tuning Guide, SC28-1751*
- *OS/390 MVS Initialization and Tuning Reference, SC28-1752*
- *OS/390 MVS Installation Exits, SC28-1753*
- *OS/390 MVS JCL Reference, GC28-1757*
- *OS/390 MVS JCL User's Guide, GC28-1758*
- *OS/390 MVS Planning: Global Resource Serialization, GC28-1759*
- *OS/390 MVS Planning: Operations, GC28-1760*
- *OS/390 MVS Planning: Workload Management, GC28-1761*
- *OS/390 MVS Programming: Assembler Services Guide, GC28-1762*
- *OS/390 MVS Programming: Assembler Services Reference, GC28-1910*
- *OS/390 MVS Programming: Authorized Assembler Services Guide, GC28-1763*
- *OS/390 MVS Programming: Authorized Assembler Services Reference, Volumes 1-4, GC28-1764, GC28-1765, GC28-1766, GC28-1767*
- *OS/390 MVS Programming: Callable Services for High-Level Languages, GC28-1768*
- *OS/390 MVS Programming: Extended Addressability Guide, GC28-1769*

- *OS/390 MVS Programming: Sysplex Services Guide, GC28-1771*
- *OS/390 MVS Programming: Sysplex Services Reference, GC28-1772*
- *OS/390 MVS Programming: Workload Management Services, GC28-1773*
- *OS/390 MVS Routing and Descriptor Codes, GC28-1778*
- *OS/390 MVS Setting Up a Sysplex, GC28-1779*
- *OS/390 MVS System Codes, GC28-1780*
- *OS/390 MVS System Commands, GC28-1781*
- *OS/390 MVS System Messages Volume 1, GC28-1784*
- *OS/390 MVS System Messages Volume 2, GC28-1785*
- *OS/390 MVS System Messages Volume 3, GC28-1786*
- *OS/390 MVS System Messages Volume 4, GC28-1787*
- *OS/390 MVS System Messages Volume 5, GC28-1788*
- *OS/390 MVS Using the Subsystem Interface, SC28-1789*
- *OS/390 Security Server (RACF) Auditor's Guide, SC28-1916*
- *OS/390 Security Server (RACF) Command Language Reference, SC28-1919*
- *OS/390 Security Server (RACF) General User's Guide, SC28-1917*
- *OS/390 Security Server (RACF) Introduction, GC28-1912*
- *OS/390 Security Server (RACF) Macros and Interfaces, SK2T-6700 (OS/390 Collection Kit ), SK27-2180 (OS/390 Security Server Information Package )*
- *OS/390 Security Server (RACF) Security Administrator's Guide, SC28-1915*
- *OS/390 Security Server (RACF) System Programmer's Guide, SC28-1913*
- *OS/390 SMP/E Reference, SC28-1806*
- *OS/390 SMP/E User's Guide, SC28-1740*
- *OS/390 RMF User's Guide, SC28-1949*
- *OS/390 TSO/E CLISTS, SC28-1973*
- *OS/390 TSO/E Command Reference, SC28-1969*
- *OS/390 TSO/E Customization, SC28-1965*
- *OS/390 TSO/E Messages, GC28-1978*
- *OS/390 TSO/E Programming Guide, SC28-1970*
- *OS/390 TSO/E Programming Services, SC28-1971*
- *OS/390 TSO/E User's Guide, SC28-1968*
- *OS/390 DCE Administration Guide, SC28-1584*
- *OS/390 DCE Introduction, GC28-1581*
- *OS/390 DCE Messages and Codes, SC28-1591*
- *OS/390 UNIX System Services Command Reference, SC28-1892*
- *OS/390 UNIX System Services Planning, SC28-1890*
- *OS/390 UNIX System Services User's Guide, SC28-1891*
- *OS/390 UNIX System Services Programming: Assembler Callable Services Reference, SC28-1899*

j

### PL/I for MVS & VM

- *IBM PL/I MVS & VM Language Reference, SC26-3114*
- *IBM PL/I MVS & VM Programming Guide, SC26-3113*

### OS PL/I

- *OS PL/I Programming Language Reference, SC26-4308*
- *OS PL/I Programming Guide, SC26-4307*

### Prolog

- *IBM SAA AD/Cycle Prolog/MVS & VM Programmer's Guide, SH19-6892*

### Remote Recovery Data Facility

- *Remote Recovery Data Facility Program Description and Operations, LY37-3710*

### Storage Management

- *DFSMS/MVS Storage Management Library: Implementing System-Managed Storage, SC26-3123*
- *MVS/ESA Storage Management Library: Leading a Storage Administration Group, SC26-3126*
- *MVS/ESA Storage Management Library: Managing Data, SC26-3124*
- *MVS/ESA Storage Management Library: Managing Storage Groups, SC26-3125*
- *MVS Storage Management Library: Storage Management Subsystem Migration Planning Guide, SC26-4659*

### System/370 and System/390

- *ESA/370 Principles of Operation, SA22-7200*
- *ESA/390 Principles of Operation, SA22-7201*
- *System/390 MVS Sysplex Hardware and Software Migration, GC28-1210*

### System Network Architecture (SNA)

- *SNA Formats, GA27-3136*
- *SNA LU 6.2 Peer Protocols Reference, SC31-6808*
- *SNA Transaction Programmer's Reference Manual for LU Type 6.2, GC30-3084*
- *SNA/Management Services Alert Implementation Guide, GC31-6809*

### TCP/IP

- *IBM TCP/IP for MVS: Customization & Administration Guide, SC31-7134*
- *IBM TCP/IP for MVS: Diagnosis Guide, LY43-0105*
- *IBM TCP/IP for MVS: Messages and Codes, SC31-7132*
- *IBM TCP/IP for MVS: Planning and Migration Guide, SC31-7189*

**VS COBOL II**

- *VS COBOL II Application Programming Guide for MVS and CMS, SC26-4045*
- *VS COBOL II Application Programming: Language Reference, GC26-4047*
- *VS COBOL II Installation and Customization for MVS, SC26-4048*

**VS FORTRAN**

- *VS FORTRAN Version 2: Language and Library Reference, SC26-4221*
- *VS FORTRAN Version 2: Programming Guide for CMS and MVS, SC26-4222*

**VTAM**

- *Planning for NetView, NCP, and VTAM, SC31-8063*
- *VTAM for MVS/ESA Diagnosis, LY43-0069*
- *VTAM for MVS/ESA Messages and Codes, SC31-6546*
- *VTAM for MVS/ESA Network Implementation Guide, SC31-6548*
- *VTAM for MVS/ESA Operation, SC31-6549*
- *VTAM for MVS/ESA Programming, SC31-6550*
- *VTAM for MVS/ESA Programming for LU 6.2, SC31-6551*
- *VTAM for MVS/ESA Resource Definition Reference, SC31-6552*

# Index

# How to send your comments

DB2 Universal Database for OS/390
Command Reference
Version 6

Publication No. SC26-9006-01

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 for OS/390 documentation. You can use any of the following methods to provide comments.

- Send your comments by e-mail to db2pubs@vnet.ibm.com and include the name of the product, the version number of the product 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, page number, or a help topic title).
- Send your comments from the Web. Visit the DB2 for OS/390 Web site at:

  **http://www.ibm.com/software/db2os390**

  The Web site has a feedback page that you can use to send comments.
- Complete the readers' comment form at the back of the book and return it by mail, by fax (800-426-7773 for the United States and Canada), or by giving it to an IBM representative.

# Readers' Comments

**DB2 Universal Database for OS/390**
**Command Reference**
**Version 6**

**Publication No. SC26-9006-01**

How satisfied are you with the information in this book?

| | Very Satisfied | Satisfied | Neutral | Dissatisfied | Very Dissatisfied |
|---|---|---|---|---|---|
| Technically accurate | ☐ | ☐ | ☐ | ☐ | ☐ |
| Complete | ☐ | ☐ | ☐ | ☐ | ☐ |
| Easy to find | ☐ | ☐ | ☐ | ☐ | ☐ |
| Easy to understand | ☐ | ☐ | ☐ | ☐ | ☐ |
| Well organized | ☐ | ☐ | ☐ | ☐ | ☐ |
| Applicable to your tasks | ☐ | ☐ | ☐ | ☐ | ☐ |
| Grammatically correct and consistent | ☐ | ☐ | ☐ | ☐ | ☐ |
| Graphically well designed | ☐ | ☐ | ☐ | ☐ | ☐ |
| Overall satisfaction | ☐ | ☐ | ☐ | ☐ | ☐ |

Please tell us how we can improve this book:

May we contact you to discuss your comments? ☐ Yes ☐ No

| | |
|---|---|
| Name | Address |
| Company or Organization | |
| Phone No. | |

**Readers' Comments**
SC26-9006-01

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department BWE/H3
PO Box 49023
San Jose, CA 95161-9945

Fold and Tape **Please do not staple** Fold and Tape

SC26-9006-01

Cut or Fold
Along Line

Program Number: 5645-DB2

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC26-9006-01