



IBM DATABASE 2 Version 2

SC26-4378-0

Command and Utility Reference

Release 1

First Edition (September 1988)

This edition applies to Release 1 of IBM DATABASE 2 Version 2, Program Number 5665-DB2, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Changes" in the first section of the publication. Specific changes are indicated by a vertical bar to the left of the change. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's program may be used. Any functionally equivalent program may be used instead.

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. If you request publications from the address given below, your order will be delayed because publications are not stocked there.

A Reader's Comment Form is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publishing, P. O. Box 49023, San Jose, California, U.S.A. 95161-9023. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. Introduction	1
How to Use This Book	3
Prerequisite Knowledge	3
DB2 Tasks and DB2 Books	4
Summary of Changes	5
Summary of Changes to this Publication	6
How to Read the Syntax Diagrams	7
DSN Subcommand Parsing	11
DB2 Command Parsing	11
Utility Statement Parsing	15
Naming Conventions	16
Binding Process	17
Chapter 2. Commands	21
BIND (DSN)	28
/CHANGE (IMS/VS)	34
DCLGEN (DECLARATIONS GENERATOR) (DSN)	36
/DISPLAY (IMS/VS)	42
-DISPLAY DATABASE (DB2)	45
-DISPLAY RLIMIT (DB2)	50
-DISPLAY THREAD (DB2)	51
-DISPLAY TRACE (DB2)	53
-DISPLAY UTILITY (DB2)	58
DSN (TSO)	60
DSNC (CICS Attachment Facility)	63
DSNC DISCONNECT (CICS Attachment Facility)	64
DSNC DISPLAY (CICS Attachment Facility)	65
DSNC MODIFY (CICS Attachment Facility)	67
DSNC STOP (CICS attachment facility)	69
DSNC STRT (CICS attachment facility)	70
DSNH (TSO CLIST)	71
DSNU (TSO CLIST)	88
END (DSN)	94
FREE (DSN)	95
MODIFY irImproc,ABEND (MVS IRLM)	97
MODIFY irImproc,START trace (MVS IRLM)	98
MODIFY irImproc,STATUS (MVS IRLM)	100
MODIFY irImproc,STOP trace (MVS IRLM)	102
-MODIFY TRACE (DB2)	104
REBIND (DSN)	106
-RECOVER BSDS (DB2)	111
-RECOVER INDOUBT (DB2)	112
RUN (DSN)	114
SPUFI (DSN)	116
/SSR (IMS/VS)	117
/START (IMS/VS)	118
-START DATABASE (DB2)	119
-START DB2 (DB2)	122
START irImproc (MVS IRLM)	125
-START RLIMIT (DB2)	127
-START TRACE (DB2)	128
/STOP (IMS/VS)	136

-STOP DATABASE (DB2)	137
-STOP DB2 (DB2)	139
STOP irlmproc (MVS IRLM)	140
-STOP RLIMIT (DB2)	141
-STOP TRACE (DB2)	142
-TERM UTILITY (DB2)	146
/TRACE (IMS/VS)	148
Chapter 3. Running DB2 Utilities	151
Description of Utilities	153
Data Sets Used by Utilities	156
Invoking Utilities	158
Monitoring and Controlling Utilities	167
CHANGE LOG INVENTORY (DSNJU003) (Utility)	170
CHECK (Utility)	178
COPY (Utility)	184
DIAGNOSE (Utility)	189
DSN1CHKR (Service Aid)	193
DSN1COPY (Service Aid)	199
DSN1LOGP (Service Aid)	208
DSN1PRNT (Service Aid)	219
LOAD (Utility)	222
MERGECOPY (Utility)	243
MODIFY (Utility)	247
PRINT LOG MAP (DSNJU004) (Utility)	250
QUIESCE (Utility)	251
RECOVER (Utility)	253
REORG (Utility)	261
REPAIR (Utility)	268
REPORT (Utility)	280
RUNSTATS (Utility)	283
STOSPACE (Utility)	290
Appendix. DB2 Limits	293
Glossary	295
Bibliography	301
Index	303

Chapter 1. Introduction

How to Use This Book	3
How This Book is Organized	3
Prerequisite Knowledge	3
DB2 Tasks and DB2 Books	4
Summary of Changes	5
Version 2	5
Summary of Changes to this Publication	6
How to Read the Syntax Diagrams	7
DSN Subcommand Parsing	11
DB2 Command Parsing	11
Parts of a DB2 Command	12
Characters with Special Meanings	14
Examples of Keyword Entry	14
Utility Statement Parsing	15
Example of Parsing	15
Example of Option Description	16
Naming Conventions	16
Binding Process	17

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

How to Use This Book

This book is intended to serve as a reference (it is not intended as a guide).

IBM DATABASE 2 (DB2) guides:

- Tell what tasks can be performed with a particular program and explain how (with which commands or statements) these tasks can be performed.
- Present examples of the use of these commands or statements. The examples use the sample database and tables that are provided with DB2.
- Are especially useful for new users, who need to have tasks explained.

This reference:

- Contains detailed descriptions of each command and utility.
- Is organized so that information can be retrieved quickly. (Commands and utilities are listed alphabetically.)
- Is particularly useful for users who do not need the close guidance a guide provides, who know the task and program well, and who only want to check on the details of syntax and semantics (although, of course, this reference can also be useful to new users).

How This Book is Organized

This book is divided into one introductory chapter, a second chapter that describes every command, a third chapter that describes every utility, and one appendix.

The first chapter describes the syntax of DB2 commands and utilities (on page 7), and describes the rules for parsing them (on page 11).

Chapter 2 contains two types of information:

- Six tables, at the beginning, list commands by category, give brief descriptions of their functions, and refer you to descriptions later in the section.
- The rest of the section describes commands in alphabetic order.

Chapter 3 explains how to use DB2 utilities, and describes each utility in alphabetic order.

The book contains one appendix, Appendix, "DB2 Limits" on page 293.

Prerequisite Knowledge

It is assumed that you possess an understanding of system administration, database administration, or application programming in the DB2 environment, as provided by the appropriate guide, and that you have some knowledge of the following:

- One of the transaction managers (CICS, IMS/VS), or TSO
- A programming language (Assembler language, PL/I, COBOL, APL2, BASIC, FORTRAN, COB2, or C)
- OS/VS MVS Job Control Language
- Structured Query Language (SQL).

DB2 Tasks and DB2 Books

The most rewarding task associated with a database management system is asking questions of it and getting answers, the task we call “end use.” But other tasks are also necessary—defining the parameters of the system, putting the data in place, and so on. We group the tasks associated with DB2 into the following major categories:

End use: End users want to issue SQL statements to retrieve data. Possibly they also insert, update, or delete data, still by means of SQL statements. They may need an elementary introduction to SQL, detailed instructions for using SPUFI, and an alphabetized reference to the types of SQL statements. Those are found in *SQL User’s Guide* and *SQL Reference*.

The same users may issue SQL statements through QMF or some other program, and the library for that program may provide all the instruction or reference material they need. For a list of 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. Since they write SQL statements, they need *SQL User’s Guide* and *SQL Reference*, just as end users do.

They 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/VS or DB2 and CICS. The material you need for writing a host program containing SQL is in *Application Programming Guide*. And for those handling errors, we also recommend *Messages and Codes*.

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

- Section 2 of *System and Database Administration Guide* deals with defining the DB2 system, estimating storage needs, and running the jobs that install the DB2 program.
- Section 3 of *System and Database 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 4 of *System and Database Administration Guide* describes ways to control 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 5 of *System and Database 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 6 of *System and Database 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, though possibly at the expense of other parts.

In addition, the appendixes in *System and Database Administration Guide* contain valuable information on DB2 sample tables, NLS support, writing exit routines, and interpreting DB2 trace output.

If you are involved with DB2 only to install the system, design the database, or plan operational procedures, *System and Database Administration Guide* may be all you need. If you also intend to carry out your own plans by creating DB2 objects, granting privileges, running utility jobs, and so on, then you also need:

- *SQL Reference*; you will use SQL statements to create, alter, and drop objects and grant and revoke privileges.
- *Command and Utility Reference*; you will do many of your tasks by running utility jobs.
- *Messages and Codes*.

Diagnosis: A diagnostician detects and describes errors in the DB2 program. He or she may also recommend or apply a remedy. The documentation needed for the task is in *Diagnosis Guide and Reference* and *Messages and Codes*.

Titles of books in the DB2 library begin with *IBM DATABASE 2 Version 2*. However, references from one DB2 book to another are shortened and do not include the product title. Instead, they point directly to the section that holds the information. For a complete list of books in the DB2 library, and the sections in each book, see the back cover.

Summary of Changes

Version 2

DB2 Version 2 offers all of the functions available with Version 1, plus:

- **Referential integrity**, which provides the ability to have DB2 ensure that references from one table to another are valid. You can define referential constraints which are automatically enforced on the tables you name.
- **Performance** enhancements that significantly improve transaction, query, data definition, and utility response time. Not only is response time improved, but the amount of machine resource used by DB2 is reduced, freeing it for other purposes.
- **MVS/ESA** support enhancements which allow MVS/ESA customers to take advantage of improvements in performance and recovery for multiple address space operations. These improvements provide greater efficiency in hardware resource use from which DB2 directly benefits.
- **Application development and tuning flexibility** enhancements. Before loading data, you do not need to create indexes as often as you did in past releases of DB2. If and when additional indexes are needed, they can be created more quickly and they are well-organized.

- **Authorization control** that is more flexible. A user can be represented not only by a single (primary) authorization identifier, but also by one or more secondary identifiers, which can serve as group identifiers. This is helpful when using a security system like RACF. Users can also create objects to be owned by their secondary identifiers.
- **Audit trace** that allows you to determine who has accessed data stored in DB2 tables. The trace can record events of several types, such as unauthorized access attempts, write accesses, and read accesses.
- **Resource limit facility, or governor**, to control the amount of resources used when certain SQL queries are run. The governor terminates dynamic queries that exceed a predetermined time limit.
- **Recovery of data** enhancements. The RECOVER utility allows multiple table spaces and partitions to be recovered at the same time.
- **Recovery point in time**, with the addition of the QUIESCE utility. This allows you to establish a point at which a list of table spaces is consistent.
- **LOAD utility** enhancements in detecting and processing unique index violations. When the LOAD utility detects duplicate values for unique indexes, duplicate data is not loaded.
- **Segmented table spaces**, which provide performance advantages for storing more than one table in a single table space.
- **Alter storage attributes** ability. You can reassign table spaces, index spaces, and partitions to different storage groups or to user-managed data sets.
- **DFHSM** (Data Facility Hierarchical Storage Manager) support enhancements. You can perform synchronous automatic recall controlled by a time value you set.
- **DL/I batch** support. This programming enhancement allows you to access IMS/VS data and DB2 data in the IMS/VS batch environment.
- **National Language Support** to display DB2I help and task panels in Kanji.
- **Serviceability improvements** to decrease service costs and problem resolution time.

Throughout this book, the term *MVS* is used to represent both *MVS/Enterprise Systems Architecture (MVS/ESA™)* and *MVS/Extended Architecture (MVS/XA™)*. When it is necessary to make a technical distinction between the two environments, the specific term is used. *CICS* is used to represent both *CICS/OS/VS* and *CICS/MVS*.

Summary of Changes to this Publication

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

“Chapter 2. Commands” on page 21 describes changes to many of the commands available in Version 1. “Chapter 2. Commands” also introduces the following new commands:

- -DISPLAY RLIMIT
- -MODIFY TRACE
- -START RLIMIT
- -STOP RLIMIT

“Chapter 3. Running DB2 Utilities” on page 151 describes changes to many of the utilities and service aids available in Version 1. “Chapter 3. Running DB2 Utilities” also introduces the following new utilities and service aids:

- CHECK DATA option of the CHECK utility
- DIAGNOSE utility
- DSN1CHKR service aid
- QUIESCE utility
- REPORT utility


In addition, the DSN1COPY, DSN1LOGP, and DSN1PRNT service aids, previously documented in *Diagnosis Guide and Reference*, are now documented in this publication.


All new, in addition to changed, information is marked with revision bars.


How to Read the Syntax Diagrams



This section explains how to read the diagrams that define the syntax of the statements described in this book.

You read the syntax diagrams from left to right and top to bottom.

The  symbol indicates the beginning of a statement.

The  symbol indicates that the statement syntax is continued.

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

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

The  symbol indicates the end of a statement.

Keywords appear in uppercase (for example, PARM1). They must be entered exactly as shown. Variables appear in all lowercase letters (for example, parm_x).

If a comma (,), arithmetic symbol (>, <, =, and so forth), or set of parentheses (()) is shown with either a keyword or a variable, it must be entered as part of the statement or command.

No Parameters

An example of a statement with no parameters is shown below.

 STATEMENT 

You must write:

STATEMENT

Required Parameters

The conventions assume that parameters must be separated by one or more blanks.

Required parameters appear on the same horizontal line (the main path) as the command or statement.



You must write:

STATEMENT PARAM1 PARAM2

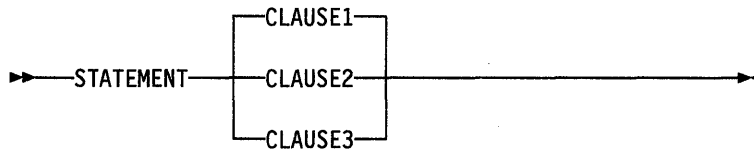
Parentheses around parameters or clauses must be included:



You must write:

STATEMENT (PARAM1) (PARAM2)

When there is a vertical list of parameters, one of which is on the main path, you must choose one of them.



Examples: STATEMENT . CLAUSE1
or STATEMENT CLAUSE2
or STATEMENT CLAUSE3

Optional Parameters

A single optional parameter appears below the main path.

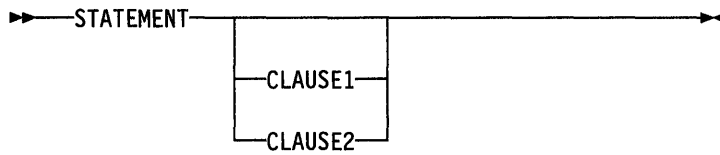


Examples: STATEMENT
or STATEMENT PARAMETER



Examples: STATEMENT
or STATEMENT (PARAMETER)

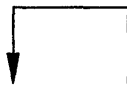
If you have a choice of more than one optional parameter, the parameters will appear in a vertical list below the main path.



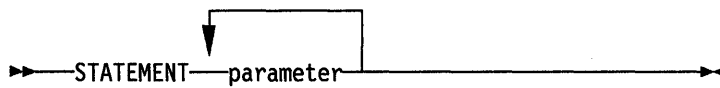
Examples: STATEMENT
 or STATEMENT CLAUSE1
 or STATEMENT CLAUSE2

Multiple Parameters

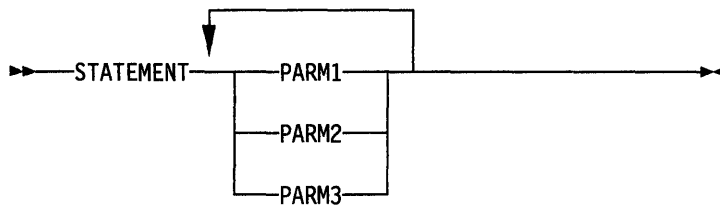
The repeat symbol:



indicates that you can specify more than one parameter or a single parameter more than once.

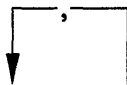


Examples: STATEMENT PARMX
 or STATEMENT PARMY
 or STATEMENT PARMX PARMZ

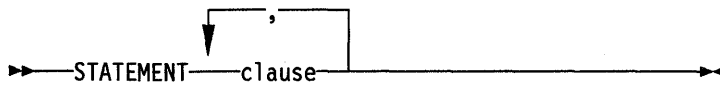


Examples: STATEMENT PARM2
 or STATEMENT PARM1 PARM3
 or STATEMENT PARM1 PARM2 PARM3
 or STATEMENT PARM3 PARM1 PARM1

If the repeat symbol contains a comma

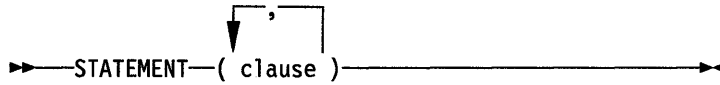


you must separate multiple parameters by commas.

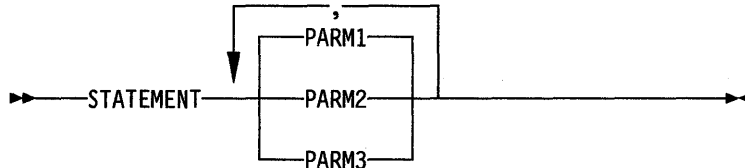


Examples: STATEMENT CLAUSEX

or STATEMENT CLAUSEY
 or STATEMENT CLAUSEX,CLAUSEY



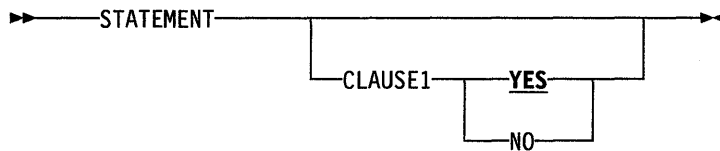
Examples: STATEMENT (CLAUSEX)
 or STATEMENT (CLAUSEY)
 or STATEMENT (CLAUSEX,CLAUSEY)



Examples: STATEMENT PARM2
 or STATEMENT PARM1, PARM3
 or STATEMENT PARM1, PARM2, PARM3
 or STATEMENT PARM3, PARM1, PARM1

Default Parameters

A parameter underscored or printed in boldface type (or both) is a default parameter. If you don't write it in the statement, you will get the same result as if you had actually written it.



Example: STATEMENT

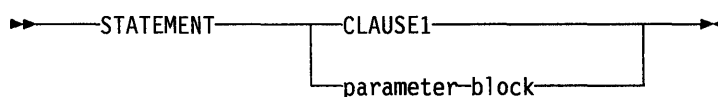
...is the equivalent of

STATEMENT CLAUSE1 YES

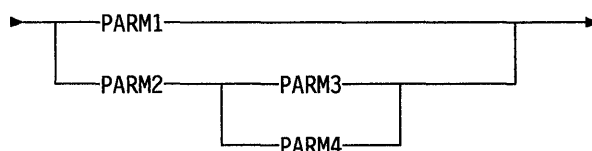
Some defaults are not underlined because they are determined by prior SQL statements; for example, the default that is used if the BUFFERPOOL parameter of the CREATE TABLESPACE statement is omitted is determined by the BUFFERPOOL specified when the database was created, so it cannot be underlined.

Variables Representing Several Parameters

Sometimes a set of several parameters is represented by a single variable. For example, in the following diagram, the variable parameter-block could be replaced by any of the interpretations of the diagram that is headed **Parameter-Block**.



Parameter-Block



Example: STATEMENT CLAUSE1
or STATEMENT PARAM1
or STATEMENT PARAM2 PARAM4

DSN Subcommand Parsing

The parsing of DSN subcommands (performed by the DSN command processor; see "DSN (TSO)" on page 60) conforms to standard TSO command parsing conventions. For information about TSO command parsing, please refer to the appropriate MVS/XA or MVS/ESA publication.

The recognition character *must* be a hyphen.

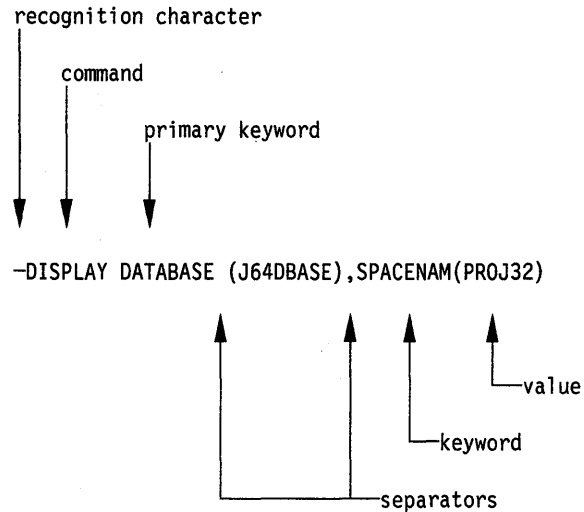
Abbreviations

The names of the **DSN command and its subcommands** cannot be abbreviated. But, within the subcommands, keywords may be abbreviated by truncation, up to the point at which their meaning becomes ambiguous. For example, in the DCLGEN command, the keyword TABLE can be abbreviated T, because no other keyword begins with T. But, in Release 1, two keywords begin with L: LANGUAGE and LIBRARY. Hence their minimum abbreviations are LA and LI.

That situation also illustrates why we recommend that you use the full operand names in all your stored procedures: Release 2 of DB2 adds a third keyword to DCLGEN that also begins with L: LABEL. After that addition, the minimum abbreviation for LANGUAGE is LAN, and for LABEL, LAB. Conceivably, possible later releases could change other currently acceptable minimum abbreviations.

DB2 Command Parsing

DB2 commands follow a pattern like this:



Parts of a DB2 Command

The parts of a command are:

- A *recognition character*. Typically, the recognition character is a hyphen (-), and it is shown as a hyphen throughout this book. But:
 - If the command is given from an MVS console, the recognition character must be the subsystem recognition character (SRC).
 - If the command is given from an IMS/VS terminal, the recognition character must be the command recognition character (CRC).

Both the subsystem and command recognition characters can be defined by your installation. For more information, see the Section 2 of *System and Database Administration Guide*.

 - If the command is given from a CICS terminal, the recognition character *must* be a hyphen.
- The *command name*. Command names have synonyms, as described under “Synonyms,” following.
- *Operands*. These are combinations of keywords and parameters that can be specified for the command.
 - *Keywords* may be required or optional. They must be entered exactly as shown in the descriptions of the commands (or they may be abbreviated, as described under “Synonyms” on page 13).
 - A keyword may have zero or more *parameters*. A parameter list, if present, must be enclosed in parentheses.
 - *Separators*. These may 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.

Synonyms

DB2 command names have DB2 defined-synonyms; the acceptable synonyms are listed with each command description, as well as in the following table.

The following two tables list each DB2 command name, associated primary keywords, primary keyword values (if they have abbreviations), and their synonyms.

Command (synonym)	Primary Keywords (synonym)
-DISPLAY (-DIS)	DATABASE, RLIMIT (RLIM), THREAD (THD), TRACE, UTILITY (UTIL)
-MODIFY (-MOD)	TRACE
-RECOVER (-REC)	BSDS, INDOUBT (IND)
-START (-STA)	DATABASE, DB2, RLIMIT (RLIM), TRACE
-STOP (-STO)	DATABASE, DB2, RLIMIT (RLIM), TRACE

Figure 1. DB2 Command and Primary Keyword Synonyms

Keyword (synonym)	Value (synonym)	Associated Commands
ACCESS (ACC)	values do not have synonyms	-START DATABASE, -START DB2
ACTIVE (A)	values do not have synonyms	-DISPLAY DATABASE
ACTION (ACT)	values do not have synonyms	-RECOVER INDOUBT
CLASS (C)	values do not have synonyms	-DISPLAY TRACE, -MODIFY TRACE, -START TRACE, -STOP TRACE
DEST (D)	values do not have synonyms	-DISPLAY TRACE, -START TRACE, -STOP TRACE
RESTRICT (RES)	values do not have synonyms	-DISPLAY DATABASE
SPACENAM (SPACE)	values do not have synonyms	-DISPLAY DATABASE, -START DATABASE, -STOP DATABASE
TYPE (T)	ACTIVE (A) INDOUBT (I)	-DISPLAY THREAD
TDATA	CORRELATION (COR) TRACE (TRA) CPU	-START TRACE
<i>trace-type</i>	GLOBAL (G) ACCTG (A) STAT (S) PERFM (P) AUDIT (AU) MONITOR (MON)	-DISPLAY TRACE, -MODIFY TRACE, -START TRACE, -STOP TRACE

Figure 2. DB2 Primary Keyword and Value Synonyms

Characters with Special Meanings

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

- b** (blank) is a separator.
- ,** (comma) is a separator.
Multiple separators are equivalent to a single separator, except in strings enclosed between apostrophes.
- '** (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 CHARACTER SET option is set, during installation, to KATAKANA, then letters are not folded to uppercase, whether in an SQL string constant or not.
- "** (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.
- =** (equal sign) separates a single parameter from a keyword.
- (** (open parenthesis) marks the beginning of a parameter list.
-)** (close parenthesis) marks the end of a parameter list.
- :** (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 may be used this way only in commands in which the operation is specifically permitted.
- *** (asterisk) means "all" or "subset beginning with." 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 may 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 may 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 . . . namen) ACCESS(RO)
- SPACENAM (name1,name2) ACCESS(RO)
- ACCESS (RO),SPACENAM = name
- Combinations of the above

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

Utility Statement Parsing

Utility statements are read from the SYSIN input stream. The statements in that stream must obey these rules:

- If the SYSIN records are fixed-length 80-character records, columns 73 through 80 are ignored.
- The records are concatenated before being parsed; hence a statement or any of its syntactical constructs may span more than one record. No continuation character is necessary.
- The SYSIN stream must begin with one of these *utility names*:

CHECK	MERGECOPY	REORG	STOSPACE
COPY	MODIFY	REPAIR	
DIAGNOSE	QUIESCE	REPORT	
LOAD	RECOVER	RUNSTATS	

At least one blank character must follow the name.

- Other syntactical constructs in the utility statement **describe options**; they may be separated from each other by an arbitrary number of blanks.
- The SYSIN stream can contain multiple utility control statements.

The name determines which options may follow it, until the next name is encountered.

Options are typically described by an *option keyword*, followed by a *parameter*. The parameter value may or may not be a keyword. Parameters' values may be enclosed in parentheses, but are not generally required to be. The syntax diagrams for utility statements included in "Chapter 3. Running DB2 Utilities" show parentheses where they are required.

Example of Parsing

The following example illustrates the distinctions among utility names, option keywords, and parameters, and the use of optional parentheses.

The following two statements are both syntactically correct and functionally equivalent:

```
COPY TABLESPACE (DSN8D21A.DSN8S21E) SHRLEVEL(CHANGE) DEVT (SYSDA)
COPY TABLESPACE DSN8D21A.DSN8S21E SHRLEVEL CHANGE DEVT SYSDA
```

The following list indicates how each token was parsed:

COPY	utility name
TABLESPACE	option keyword
DSN8D21A.DSN8S21E	parameter value
SHRLEVEL	option keyword
CHANGE	parameter keyword
DEVT	option keyword
SYSDA	parameter keyword

Example of Option Description

Where the syntax of each utility statement is described in “Chapter 3. Running DB2 Utilities,” parameters are indented under the option keyword they must follow. Here is a typical example:

WORKDDN *ddname*

Names a temporary work file.

ddname is the data set name of the temporary file. The default is **SYSUT1**.

In the example, WORKDDN is an option keyword, and *ddname* is a parameter. As noted above, parameter values may be enclosed in parentheses but are not always required to be. The description of the image copy data set could be written as either

WORKDDN SYSUT1 or WORKDDN (SYSUT1)

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 and utilities. See Chapter 2 of *SQL Reference* for an explanation of additional naming conventions.

connection-name

An identifier of one to eight characters identifying 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 IMS/VS and CICS processes, the connection name is the system identification name.

See Section 2 of *System and Database Administration Guide* for more information about connection names.

correlation-id

An identifier of one to twelve characters identifying a process within an address space connection. A correlation identifier can be one of the following:

- For DSN processes running in TSO foreground, the correlation identifier is the TSO logon identifier.
- For DSN processes running in TSO batch, the correlation identifier is the job name.
- For CAF processes, the correlation identifier is the TSO logon identifier.
- For IMS/VS processes, the correlation identifier is the PST#.PSBNAME
- For CICS processes, the correlation identifier is the entry identifier.tread_number.transaction_identifier.

See Section 2 of *System and Database Administration Guide* for more information about correlation identifiers.

dataset-name

An identifier of one to forty-four characters designating a data set.

ddname

An identifier of one to eight characters designating the name of a DD statement.

hexadecimal-constant

A set of digits or any of the letters from A to F (upper or lower case).

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.

member-name

An identifier of one to eight characters naming a member of a partitioned data set.

plan-name

A letter followed by a maximum of seven characters, each of which is a letter or a number. A plan name should not begin with DSN; this may conflict with DB2-provided plan names. If a plan name beginning with DSN is specified, DB2 issues a warning message.

string

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

subsystem-name

An identifier naming the DB2 subsystem as it is known to MVS.

utility-id

An identifier of one to sixteen alphanumeric characters that uniquely identifies a utility process within DB2. The identifier may contain periods.

Binding Process

DB2 has three kinds of bind processing:

- Bind processing invoked by DSN subcommands or through DB2I
- Automatic rebind
- Dynamic bind.

The main differences among these are in the manner of invocation, the handling of messages, and the output.

Only the BIND subcommand can create an application plan. To allocate DB2 resources for program execution, an application plan is required.

The REBIND subcommand reevaluates an existing application plan. See “REBIND (DSN)” on page 107 for a description of that subcommand. Automatic rebind occurs if DB2 discovers during thread creation that the specified application plan has been invalidated.

The third type of bind, dynamic bind, processes dynamic SQL statements during program execution. If VALIDATE(RUN) is used, it also processes statements that were invalid at bind time.

Inputs to BIND: The BIND process may use any of these types of information:

- Data base request modules (DBRMs). These are members of a cataloged MVS partitioned data set produced by a successful DB2 precompilation.
- Rows of the SYSIBM.SYSSTMT table. Each row describes an SQL statement that has been bound previously. The rows are part of the application plan and are input to the REBIND subcommand and to automatic rebind.
- Dynamic SQL statements. These are bound, one at a time, during execution of an application program.

BIND Processing: The following are the primary functions performed by bind:

- Validation of DB2 object definitions

A precompiled SQL statement can contain names of DB2 objects (such as tables, storage groups, and indexes). BIND checks these names, using information in DB2 catalog tables.

BIND checks whether SQL statements satisfy the rules of the SQL language. If a statement is found that contains an error, processing continues, so that any remaining errors can be discovered, but no application plan is built. Errors must be corrected and a BIND or REBIND subcommand must be issued.

If VALIDATE (RUN) is specified, an exception is made if an error is a reference to a table that does not exist. When this error occurs, a warning message is issued and building of the application plan continues. The application plan will then check for the existence of the table during program execution. If the table is still undefined, DB2 returns an SQL return code to the application program.

- Validation of the structure and syntax of SQL statements

BIND checks whether precompiled statements conform to SQL format and syntax. If a statement does not conform, DB2 issues an error message. Validation of the DBRM containing the error stops, and no application plan is built.

However, BIND continues to check any other DBRMs. Any error must be corrected and the BIND subcommand must be issued again.

- Authorization checking

The privilege set must include appropriate privileges for any tables that are referred to. If the privilege set does not include necessary privileges, and VALIDATE (RUN) has been specified or implied, authorization checking will be performed each time a statement for which authorization checks were deferred is executed. If privileges have not been granted by that time, a negative SQL return code is returned.

- Access path selection :.BIND chooses an access path to data (an index, for example, is one access path to data). An SQL request may be too complex for current resources. BIND issues an error message if it cannot proceed because resources are lacking. No application plan is built, but processing continues.

- Generation of an application plan

BIND builds an application plan that will be used when resources are allocated to an application program and when the program is executed. (Dynamic BIND builds a temporary plan for program execution.)

An application plan identifies the processing that DB2 must perform to execute each SQL statement. When a statement is bound, it is changed from its SQL form to a set of actions to be taken against databases to satisfy the request the statement is making.

See *Messages and Codes* for more information about messages from BIND.

BIND Output to Catalog Tables: BIND stores information in the following DB2 catalog tables:

- SYSIBM.SYSDBRM, one row for each database request module
- SYSIBM.SYSPLAN, one row for each application plan
- SYSIBM.SYSPLANDEP, one row for each object a plan is dependent on
- SYSIBM.SYSSTMT, one row for each SQL statement in each database request module.

The information is used when resources are allocated for application programs and when application plans are rebound. Your installation might query the catalog tables when considering whether to rebound an application plan or checking whether a particular plan is available.

For a description of each table and the information that BIND puts in it, see the appendix discussing DB2 catalog tables in *SQL Reference*.

Authorization IDs and BIND: For embedded SQL statements, it is the authorization ID of the owner of the plan that is used by DB2 for authorization checking and as the implicit qualifier for the names of tables, views, and indexes. The plan owner must have all appropriate table privileges as well as the authority to issue the BIND subcommand.

If a plan is bound with VALIDATE(BIND), all objects referenced in the SQL statements of the plan and all privileges required for the SQL statements must exist at bind time. If any object or privilege does not exist, the bind operation is unsuccessful.

If a plan is bound with `VALIDATE(RUN)`, authorization checking is still performed at bind time, but referenced objects and required privileges need not exist at this time. If any privilege required for a statement does not exist at bind time, an authorization check is performed whenever the statement is first executed within a unit of recovery and all privileges required for the statement must exist at that time. If any privilege does not exist, execution of the statement is unsuccessful. Note that when the authorization check is performed at execute time, it is performed against the plan owner, not the executor.

Chapter 2. Commands

The DSN Command and Its Subcommands	23
DB2 Commands	24
IMS/VS Commands	25
CICS Attachment Facility Commands	25
MVS IRLM Commands	26
TSO CLISTS	27
BIND (DSN)	28
/CHANGE (IMS/VS)	34
DCLGEN (DECLARATIONS GENERATOR) (DSN)	36
/DISPLAY (IMS/VS)	42
-DISPLAY DATABASE (DB2)	45
-DISPLAY RLIMIT (DB2)	50
-DISPLAY THREAD (DB2)	51
-DISPLAY TRACE (DB2)	53
-DISPLAY UTILITY (DB2)	58
DSN (TSO)	60
DSNC (CICS Attachment Facility)	63
DSNC DISCONNECT (CICS Attachment Facility)	64
DSNC DISPLAY (CICS Attachment Facility)	65
DSNC MODIFY (CICS Attachment Facility)	67
DSNC STOP (CICS attachment facility)	69
DSNC STRT (CICS attachment facility)	70
DSNH (TSO CLIST)	71
DSNU (TSO CLIST)	88
END (DSN)	94
FREE (DSN)	95
MODIFY irlmproc,ABEND (MVS IRLM)	97
MODIFY irlmproc,START trace (MVS IRLM)	98
MODIFY irlmproc,STATUS (MVS IRLM)	100
MODIFY irlmproc,STOP trace (MVS IRLM)	102
-MODIFY TRACE (DB2)	104
REBIND (DSN)	106
-RECOVER BSDS (DB2)	111
-RECOVER INDOUBT (DB2)	112
RUN (DSN)	114
SPUFI (DSN)	116
/SSR (IMS/VS)	117
/START (IMS/VS)	118
-START DATABASE (DB2)	119
-START DB2 (DB2)	122
START irlmproc (MVS IRLM)	125
-START RLIMIT (DB2)	127
-START TRACE (DB2)	128
/STOP (IMS/VS)	136
-STOP DATABASE (DB2)	137
-STOP DB2 (DB2)	139
STOP irlmproc (MVS IRLM)	140
-STOP RLIMIT (DB2)	141
-STOP TRACE (DB2)	142
-TERM UTILITY (DB2)	146
/TRACE (IMS/VS)	148

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

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

The commands are divided into these categories:

- “The DSN Command and Its Subcommands”
- “DB2 Commands”
- “IMS/VS Commands”
- “CICS Attachment Facility Commands”
- “MVS IRLM Commands”
- “TSO CLISTS”

The DSN Command and Its Subcommands

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

DSN Command or Subcommand	Function	Refer to page
BIND	Builds an application plan.	28
DB2 commands	Execute a DB2 command.	Figure 4 on page 24
DCLGEN	(DECLARATIONS GENERATOR) Produces declarations for tables or views.	36
DSN	Invokes the DSN command processor.	60
END	Ends the DSN session.	94
FREE	Deletes application plans.	95
REBIND	Updates an application plan.	106
RUN	Invokes an application program.	114
SPUFI	Invokes the SQL Processor Using File Input.	116

Figure 3. DSN Command and Subcommands

DB2 Commands

Environment: The START DB2 command may be entered only from an MVS console. All other DB2 commands may be entered from:

- An MVS console
- The DSN command processor
- A DB2I panel
- An IMS/VS terminal
- A CICS terminal
- An application program running in any of the supported environments.

Recognition character: The recognition character shown in this table is a hyphen, but other recognition characters can be used. Use the subsystem recognition character (SRC) defined for your DB2 subsystem, unless you issue the command as a DSN subcommand; in that case, you must use the hyphen.

DB2 Command	Function	Refer to page
-DISPLAY DATABASE	Displays status information for DB2 databases.	45
-DISPLAY RLIMIT	Displays status information for the Resource Limit Facility.	50
-DISPLAY THREAD	Displays information about DB2 threads.	51
-DISPLAY TRACE	Displays information about DB2 traces.	53
-DISPLAY UTILITY	Displays the status of a utility.	58
-MODIFY TRACE	Changes the IFCIDs (trace events) associated with a particular active trace.	104
-RECOVER BSDS	Reestablishes dual bootstrap data sets.	111
-RECOVER INDOUBT	Recovers threads left in doubt.	112
-START DATABASE	Makes the named database available for use.	119
-START DB2	Initializes the DB2 subsystem. (May be entered only from an MVS console.)	122
-START RLIMIT	Starts the Resource Limit Facility.	127
-START TRACE	Initiates DB2 trace activity.	128
-STOP DATABASE	Makes specified databases unavailable for applications.	137

Figure 4 (Part 1 of 2). DB2 Commands

DB2 Command	Function	Refer to page
-STOP DB2	Stops the DB2 subsystem.	139
-STOP RLIMIT	Stops the Resource Limit Facility.	141
-STOP TRACE	Stops trace activity.	142
-TERM UTILITY	Terminates execution of a utility.	146

Figure 4 (Part 2 of 2). DB2 Commands

IMS/VS Commands

Environment: Each IMS/VS command originates from an IMS/VS terminal.

IMS/VS Command	Function	Refer to page
/CHANGE	Resets an indoubt recovery unit.	34
/DISPLAY	Displays the status of the connection between IMS/VS and the specified subsystem (DB2), or displays the outstanding recovery units associated with the specified subsystem (DB2).	42
/SSR	Allows the IMS/VS operator to enter an external subsystem (DB2) command.	117
/START	Makes available the connection between IMS/VS and the specified external subsystem (DB2).	118
/STOP	Prevents application programs from accessing the external subsystem's (DB2's) resources.	136
/TRACE	Allows users to direct and control IMS/VS tracing activities.	148

Figure 5. IMS/VS Commands

CICS Attachment Facility Commands

Environment: Each CICS attachment facility command originates from a CICS terminal.

CICS Attachment Facility Commands	Function	Refer to page
DSNC	Allows you to enter DB2 commands from CICS.	63
DSNC DISCONNECT	Disconnects threads.	64
DSNC DISPLAY	Displays information on CICS transactions.	65
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..	67
DSNC STOP	Stops the CICS attachment facility.	69
DSNC STRT	Starts the CICS attachment facility.	70

Figure 6. CICS Attachment Facility Commands

MVS IRLM Commands

Environment: Each MVS IRLM command originates from an MVS console.

MVS Command	Function	Refer to page
MODIFY irImproc,ABEND	Abnormally terminates IRLM.	97
MODIFY irImproc,START trace	Starts IRLM tracing.	98
MODIFY irImproc,STATUS	Displays IRLM status.	100
MODIFY irImproc,STOP trace	Stops IRLM tracing.	102
START irImproc	Starts an IRLM component with an installation-supplied procedure.	125
STOP irImproc	Shuts down IRLM normally.	140

Figure 7. MVS Commands Affecting the IRLM

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.	71
DSNU	Generates JCL to execute DB2 utility jobs. Can be invoked directly or by using DB2I.	88

Figure 8. TSO CLISTS

BIND (DSN)

The DSN subcommand BIND builds an application plan. All DB2 programs require an application plan to allocate DB2 resources and support SQL requests made during execution. See “Usage Notes” on page 33 for a description of types of bind processing.

Environment

BIND can be invoked using DB2I, or the BIND subcommand can be issued from TSO through the DSN processor running in either foreground or background.

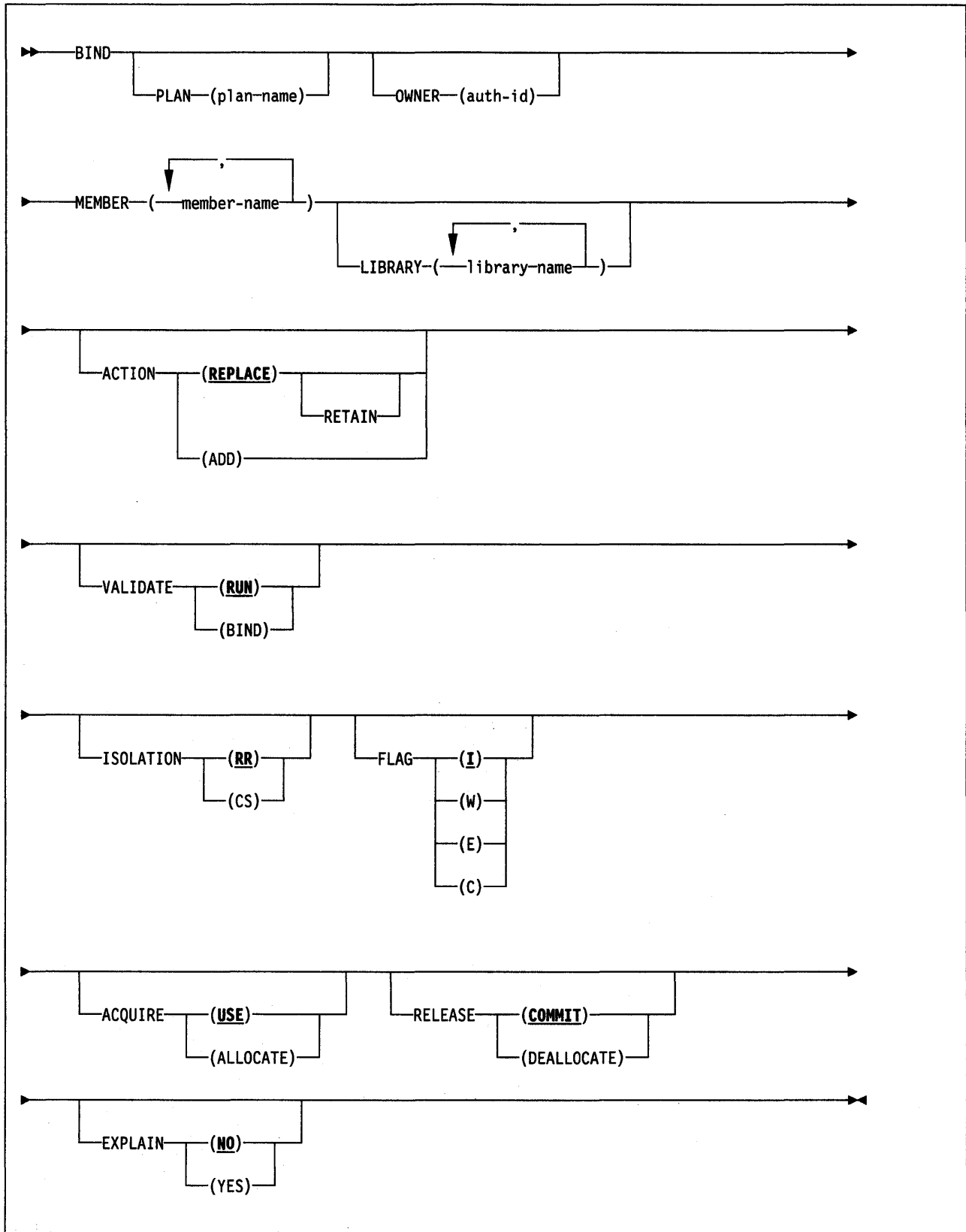
Authorization

To issue the BIND subcommand to add a new plan, the privilege set designated by the authorization ID of the owner of the plan must include the BINDADD privilege or the SYSADM authority.

To issue the BIND subcommand to replace a new plan, the privilege set defined below must include one of the following:

- Ownership of the plan
- BIND privilege on the plan
- SYSADM authority.

If the owner of the new plan differs from the owner of the existing plan, the privilege set is the privileges designated by the authorization ID of the new owner of the plan. Otherwise, the privilege set is the privileges designated by the primary authorization ID of the process.



Keyword and Parameter Descriptions**PLAN** (*plan-name*)

Names the application plan. If there are no errors, an application plan is prepared and its description is entered into the SYSIBM.SYSPLAN catalog table.

Default: If PLAN is omitted, all bind functions will be performed, including error diagnostics, but no application plan will be produced.

OWNER (*authorization-id*)

Designates the authorization ID of the owner of the plan.

If any of the authorization IDs of the process has the SYSADM authority, *authorization-id* may be any value. Otherwise, *authorization-id* must be one of the authorization IDs of the process.

Default: If the OWNER keyword is omitted, the primary authorization ID of the process becomes the owner of the plan.

MEMBER (*dbrm-member-name, ...*)

Lists the names of database request modules (DBRMs) that to be included in the application plan. Each DBRM is output from a DB2 precompilation. The partitioned data set of which this DBRM is a member can be named after the LIBRARY keyword or described in a JCL statement for ddname DBRMLIB.

The following parameters are optional.

LIBRARY (*dbrm-pds-name, ...*)

Lists the partitioned data sets (libraries) that contain as members the DBRMs named after MEMBER. Libraries are searched in the order specified. They must be cataloged.

If the libraries listed do not contain a member named in the MEMBER list, and a JCL statement exists for DBRMLIB DD, then the libraries described by that statement are searched.

Default: If LIBRARY is omitted, only the libraries described by the DBRMLIB DD statement are searched.

ACTION

Tells whether the application plan can be added or replaced.

(REPLACE)

Tells that the old application plan is to be replaced by a new one with the same name.

REBIND is sometimes an alternative to BIND with REPLACE. REBIND can be used if changes have been made that affect the plan, but the SQL statements in the program have not changed: for example, if authorizations have changed or a new index has been created. However, BIND with REPLACE must be used if the SQL statements have changed, or if the program has been recompiled.

If there are no errors, a new application plan and a new entry in the SYSIBM.SYSPLAN table replace the old plan and its entry. If no plan with the given name already exists, the new plan is added. The authorization ID designated explicitly or implicitly by the OWNER keyword becomes the owner of the new plan. If RETAIN is not used, all EXECUTE privileges that were granted by the old owner are revoked, but BIND privileges with the GRANT option are retained. The new owner of the plan has both BIND and EXECUTE privileges.

If ownership of the plan changes, the new owner grants the old owner the BIND privilege.

If there is an error, the old plan and its entry remain.

(ADD)

Tells that the application plan named does not exist, and that a new application plan is to be produced and added to the SYSIBM.SYSPLAN catalog table. If the plan name already exists in the SYSIBM.SYSPLAN table, binding stops and a diagnostic message is issued.

If there is an error, no application plan is produced and no entry is made in SYSIBM.SYSPLAN. If the privilege set lacks proper authorization, an error will occur.

RETAIN

Continues BIND and EXECUTE authorities with the GRANT option when an application plan is replaced. It has no effect if a plan is added.

The authorization ID designated explicitly or implicitly by the OWNER keyword becomes the owner of the new plan. If that user is not the previous owner of the plan, all grants for the plan issued by the previous owner are changed to name the new owner as the grantor. Those grants include the internal GRANT done when a plan is added.

If ownership of the plan changes, the new owner grants the old owner the BIND and EXECUTE privileges.

VALIDATE

Tells whether full validity checking occurs during execution of the BIND subcommand, or whether some checking is deferred until the application plan is actually used. There are two validity checks that may be deferred—checking for table existence and for appropriate access authority.

(RUN)

Defers full checking until the plan is used. Thus, any SQL statement that fails validation during the bind process is validated each time the plan is explicitly rebound with BIND or REBIND, or executed by a user. If the plan is explicitly rebound, the authorization ID of the user is used in validity checking. If the plan is executed, the authorization ID of the last user who explicitly bound the plan is used in validity checking. Only warning messages are produced when the BIND subcommand executes. The **default** is **VALIDATE (RUN)**.

(BIND)

Checks validity during bind processing. Error messages are produced.

ISOLATION

Tells how far an application bound to this plan should be isolated from the effects of other executing applications. (For more information about ISOLATION, see the section on locking in Section 3 of *System and Database Administration Guide*).

(RR)

Repeatable Read. Tells that database values read or changed by this application cannot be changed by other programs until this application commits or terminates. The **default** is **ISOLATION (RR)**.

(CS)

Cursor Stability. Tells that database values read by this application are protected only while they are being used. (Changed values are protected until this application reaches a commit point.) As soon as the program

moves from one row to another, other programs may read or change the first row.

FLAG

Tells what messages to display. Use one of the values listed to show messages of the corresponding types.

- (I)** All: informational, warning, error, and completion messages. The **default** is **FLAG (I)**.
- (W)** Only warning, error, and completion messages.
- (E)** Only error and completion messages.
- (C)** Only completion messages.

ACQUIRE

Tells whether resources are acquired when they are first accessed, or when the plan is allocated.

(USE)

Opens table spaces and acquires locks only when the application program bound to the plan first uses them. The **default** is **ACQUIRE (USE)**.

ALLOCATE

Opens all table spaces and acquires all table space locks when the plan is allocated. The value has no effect on dynamic SQL statements; the default is used for those.

RELEASE

Tells whether resources are released at each COMMIT point, or when the application terminates.

(COMMIT)

Releases resources at each commit point. The **default** is **RELEASE (COMMIT)**.

But if ACQUIRE (ALLOCATE) is used, RELEASE (DEALLOCATE) must be used as well.

(DEALLOCATE)

Releases resources only when the application plan terminates. The value has no effect on dynamic SQL statements; the default is used for those.

EXPLAIN

Obtains information about how SQL statements in the DBRM will be executed, and inserts that information into the *x.PLAN_TABLE* table where *x* is the authorization ID of the owner of the plan. This table must exist before the binding process begins. For a description of the table, see the section about EXPLAIN(SQL) in &SQLGREF..

The value of the BIND option EXPLAIN can be overridden by an SQL EXPLAIN statement in the program. Otherwise, the value of EXPLAIN in BIND applies to all explainable SQL statements in the program, and to the fullselect portion of any DECLARE CURSOR statements. In all inserts to *plan_owner_authid.PLAN_TABLE*, the value of QUERYNO is the statement number in the program.

(NO)

Provides no EXPLAIN information. The **default** is **EXPLAIN (NO)**.

(YES)

Inserts information in *userid.PLAN_TABLE*.

Usage Notes

See "Binding Process" on page 17 for information about bind processing.

Example

Example: The name of the bind plan is DSN8BC21. Include DBRM member DSN8BC21. The plan, DSN8BC21, already exists and is to be replaced by the new application plan. Because RETAIN is not specified, only the owner of the new plan will have the EXECUTE privilege on the specified plan. CS means that database values being read by this application are protected only while they are being used. (Changed values are protected until the application reaches a commit point.)

```
BIND PLAN (DSN8BC21)
  MEMBER (DSN8BC21)
  ACTION (REPLACE)
  ISOLATION (CS)
```

/CHANGE (IMS/VS)

The IMS/VS command /CHANGE resets an indoubt unit of recovery as identified by the OASN keyword of the /DISPLAY command. This command deletes the item from the standpoint of IMS/VS, but it does not communicate to DB2.

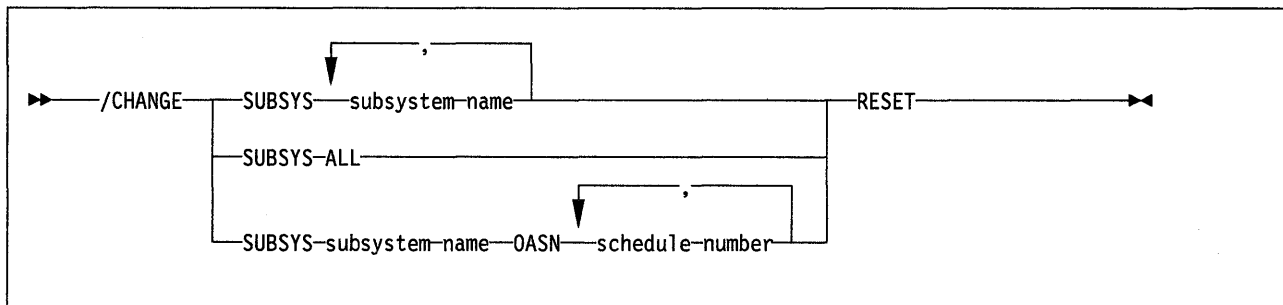
Abbreviation: /CHA

Environment

This command originates from an IMS/VS terminal.

Authorization

This command requires an appropriate level of IMS/VS authority, as described in *IMS/VS Version 2 System Administration Guide*.



Keyword and Parameter Descriptions

SUBSYS

Deletes IMS/VS recovery elements from one or more subsystems.

One of the following subparameters must be coded.

subsystem-name

Names one or more subsystems from which recovery elements will be deleted.

ALL

Deletes IMS/VS recovery elements from all subsystems.

subsystem-name OASN schedule-number

Deletes one or more origin application schedule numbers from one subsystem, named by *subsystem-name*.

schedule-number may 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 Notes

The preceding description of the /CHANGE command is a partial description only. For a complete description, see *IMS/VS Version 2 Operator's Reference Manual*.

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, COB2, or C data declaration for a table or view named in the catalog.

Environment

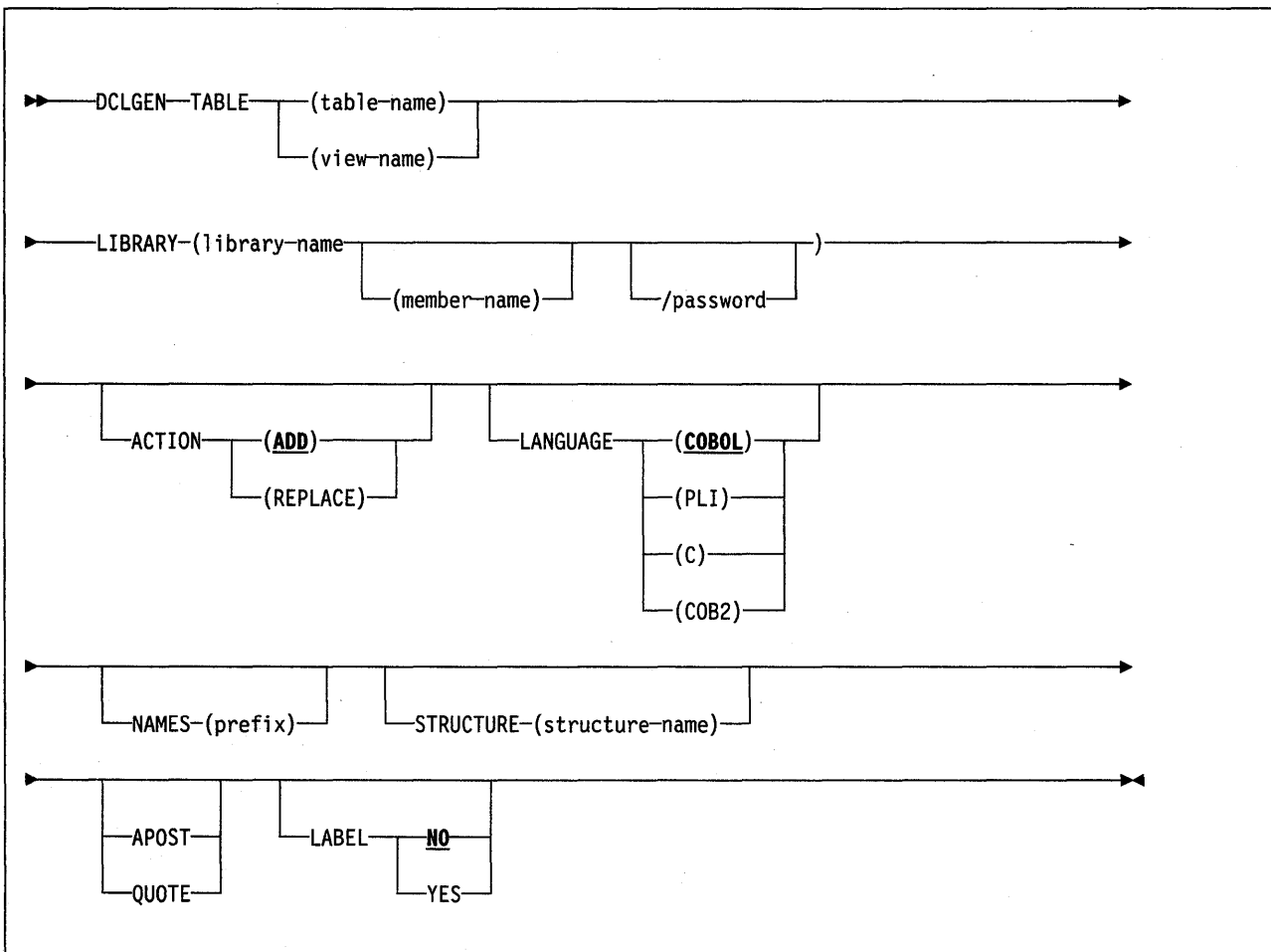
The declarations generator is invoked by the DCLGEN subcommand of the DSN command processor. This subcommand can be issued through the DSN command processor, running in either foreground or background mode, or it can be issued through DB2I.

Authorization

The privilege set defined below must include the SELECT privilege on the table or view identified in the statement. The SELECT privilege may have been explicitly granted, or inherent in another privilege. The SELECT privilege is inherent in:

- Ownership of the table or view
- SYSADM authority
- DBADM authority for the database.

The privilege set is the union of privileges designated by each authorization ID of the process.



Keyword and Parameter Descriptions

TABLE

Tells 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 contains special characters other than underscores (_), it must be enclosed between apostrophes (''). If the language is COBOL, underscores in the name will be translated into hyphens (-) by DCLGEN.
- If the name contains apostrophes, each one must be doubled (''). (Some host languages do not permit apostrophes in variable names.)
- If the name contains DBCS characters, DCLGEN can only be invoked in background mode. Use HEX mode in the ISPF editor to enter the table name.

LIBRARY (*library-name(member-name)/password*)

Names the data set into which the declarations will go. This data set must already exist and be accessible to the declarations generator. It may be either sequential or partitioned.

If the library name is not enclosed between apostrophes, DCLGEN constructs the following full data set name:

user-prefix.library-name.language.(member-name)

where:

user-prefix Is the user prefix of the primary authorization ID of the transaction.

language Is the value of the LANGUAGE parameter: COBOL, COB2, PLI, or C.

(member-name) Is optional; if not used it, the output goes to a sequential data set.

password is optional.

The following parameters are optional.

ACTION

Tells 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

Names the language of the generated declaration.

Possible languages are:

(COBOL) The default can be set during DB2 installation. The IBM supplied default is **LANGUAGE (COBOL)**.

(PLI), for PL/I.

(C), for C.

(COB2), for COB2.

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.

The field names consist of *prefix* concatenated with a number of from one to three digits. *prefix* may have up to 28 characters, with the first character alphabetic. For example, if *prefix* is ABCDE, the field names will be ABCDE1, ABCDE2, and so on, up to a maximum of ABCDE999. Special characters may be used, but use caution to avoid possible name conflicts

The column names in the table will be taken as **default** names for the fields in the output.

Note: Parsing of the DCLGEN command conforms to standard TSO parsing conventions. For information about TSO command parsing, see the *MVS/XA TSO Extensions TSO Guide to Writing a Terminal Monitor Program or Command Processor*.

STRUCTURE (*structure-name*)

Names the generated data structure. *structure-name* can have up to 31 characters, with the first character alphabetic. Special characters may be used, but use caution to avoid possible name conflicts

For SQL output, the name is the same as the table or view name. For COBOL, PL/I, C, or COB2 structure names, the default is the prefix "DCL" concatenated with the table or view name. Guard against possible conflicts with names in the source program. If the structure name specified is the same as the table or view name, DCLGEN will allow it but will issue a warning message.

Note: Parsing of the DCLGEN command conforms to standard TSO parsing conventions. For information about TSO command parsing, see the *MVS/XA TSO Extensions TSO Guide to Writing a Terminal Monitor Program or Command Processor*.

APOST**QUOTE**

Names the string delimiter character used in the host language.

APOST names the apostrophe (') as the host language string delimiter; the SQL delimiter will be the quotation mark (").

QUOTE names the quotation mark (") as the host language delimiter; the SQL delimiter will be the apostrophe (').

If neither APOST nor QUOTE is specified, the **default** will be APOST for PL/I, and either APOST or QUOTE for COBOL, depending on what was specified on panel DSNTIPF during DB2 install.

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

Tells whether to include column labels in the output as comments. (Column labels may be assigned by the LABEL ON statement.)

NO

Omits the column labels.

The default is **LABEL (NO)**.

YES

Includes the column labels.

Usage Notes

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.

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.

Table names with double-byte characters: If the table name in the DCLGEN subcommand contains double-byte characters, the linear form of the command cannot be entered from TSO foreground or from the DB2I DCLGEN panel. Submit it from TSO background. Use HEX mode in the ISPF editor to enter your table name.

Prompts: Online TSO will prompt for missing or incorrectly specified parameters.

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.

The output can be edited to add WITH DEFAULT to NOT NULL for columns that do not allow null values. If the output is edited, a default value must be provided.

COBOL: The size of values in the application must be within the declared number of digits.

DB2 uses the full size of binary integers in a way similar to NOTRUNC processing in COBOL. It may 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 may exceed 9999, use S9(5). For large integers that may 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.

Examples

Example 1: The statement

```
DCLGEN TABLE(VEEMPL) -
      LIBRARY('DSN210.SRCLIB.DATA(DSN8MPEM)') -
      LANGUAGE(PLI) -
      APOST
```

produces the following statements in DSN210.SRCLIB.DATA(DSN8MPEM):

DCLGEN (DSN)

```

/*****
/* DCLGEN TABLE(VEEMPL) - */
/* LIBRARY('DSN210.SRCLIB.DATA(DSN8MPEM)') - */
/* LANGUAGE(PLI) - */
/* APOST */
/* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS */
/*****
EXEC SQL DECLARE VEEMPL TABLE
    ( EMPNO          CHAR(6) NOT NULL,
      FIRSTNAME     VARCHAR(12) NOT NULL,
      MIDINIT       CHAR(1) NOT NULL,
      LASTNAME      VARCHAR(15) NOT NULL,
      WORKDEPT      CHAR(3) NOT NULL
    ) ;

/*****
/* PLI DECLARATION FOR TABLE VEEMPL */
/*****
DCL 1 DCLVEEMPL,
    5 EMPNO      CHAR(6),
    5 FIRSTNAME 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(VEEMPL) -
    LIBRARY('DSN210.SRCLIB.DATA(DSN8MPEM)') -
    LANGUAGE(PLI) -
    NAMES(FIELD) -
    STRUCTURE(EMPRECORD) -
    APOST

```

produces the following statements in DSN210.SRCLIB.DATA(DSN8MPEM):

```

/*****
/* DCLGEN TABLE(VEEMPL) - */
/* LIBRARY('DSN210.SRCLIB.DATA(DSN8MPEM)') - */
/* LANGUAGE(PLI) - */
/* NAMES(FIELD) - */
/* STRUCTURE(EMPRECORD) - */
/* APOST */
/* ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS */
/*****
EXEC SQL DECLARE VEEMPL TABLE
    ( EMPNO          CHAR(6) NOT NULL,
      FIRSTNAME     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/VS)

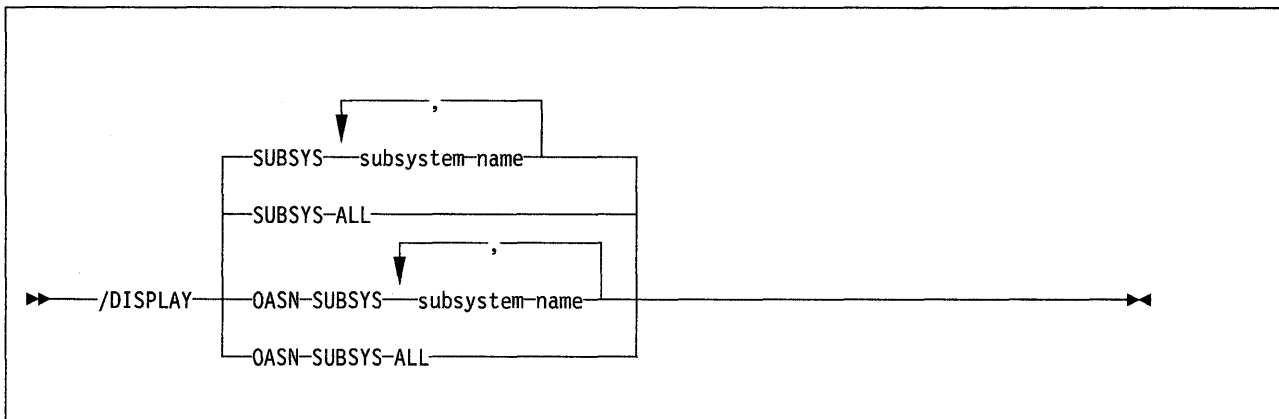
The IMS/VS command /DISPLAY displays the status of the connection between IMS/VS and an external subsystem (as well as all application programs communicating with the external subsystem), or the outstanding recovery units associated with the subsystem.

Environment

This command originates from an IMS/VS terminal.

Authorization

This command requires an appropriate level of IMS/VS authority, as described in the *IMS/VS Version 2 System Administration Guide*.



Keyword and Parameter Descriptions

One of the following parameters is required:

SUBSYS

Tells which subsystems to display information about.

subsystem-name

Names one or more subsystems. See "Usage Notes," below, for a description of possible subsystem status.

ALL

Displays information about all subsystems.

OASN

Displays the outstanding recovery units (origin application schedule numbers; OASN) associated with the external subsystems. The OASN is assigned by IMS/VS 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.

SUBSYS subsystem-name

Names one or more subsystems to display information about.

SUBSYS ALL

Displays the outstanding recovery units associated with all external subsystems.

Usage Notes

Subsystem status may be one of the following:

- **CONNECTED**

An IMS/VS control region or dependent region has successfully completed a host system IDENTIFY request to the external subsystem. At this point, the two systems may 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 /STOP SUBSYS command, or the external subsystem initialization exit routine indicated not to issue the IDENTIFY request.

- **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/VS regions must be quiesced.

- **INVALID SUBSYSTEM NAME = subsystem-name**

The indicated subsystem name has not been defined to the IMS/VS 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/VS in the external subsystem PROCLIB member, but IMS/VS still has outstanding recovery elements from a previous execution when it 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.

Note: The command recognition character (CRC) will also be displayed for each external subsystem.

A thread between an IMS/VS 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 will be 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.

/DISPLAY (IMS/VS)

Thread status may be as follows:

- **CONNECTED (CONN)**

An IMS/VS control region or dependent region has successfully completed a host system IDENTIFY request to the external subsystem.

- **ACTIVE**

An IMS/VS 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 or not a region is message-driven.

The preceding description of the /DISPLAY command is a partial description only. For a complete description, see *IMS/VS Version 2 Operator's Reference Manual*.

-DISPLAY DATABASE (DB2)

The DB2 command `-DISPLAY DATABASE` displays information about the status of DB2 databases; it also displays information about the status of table spaces, tables in segmented table spaces, and index spaces within a database. `-DISPLAY DATABASE` also indicates if a table space is in any of the "pending" status'.

Abbreviation: `-DIS DB`

Environment

This command can be entered from an MVS console, from the DSN processor under TSO, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

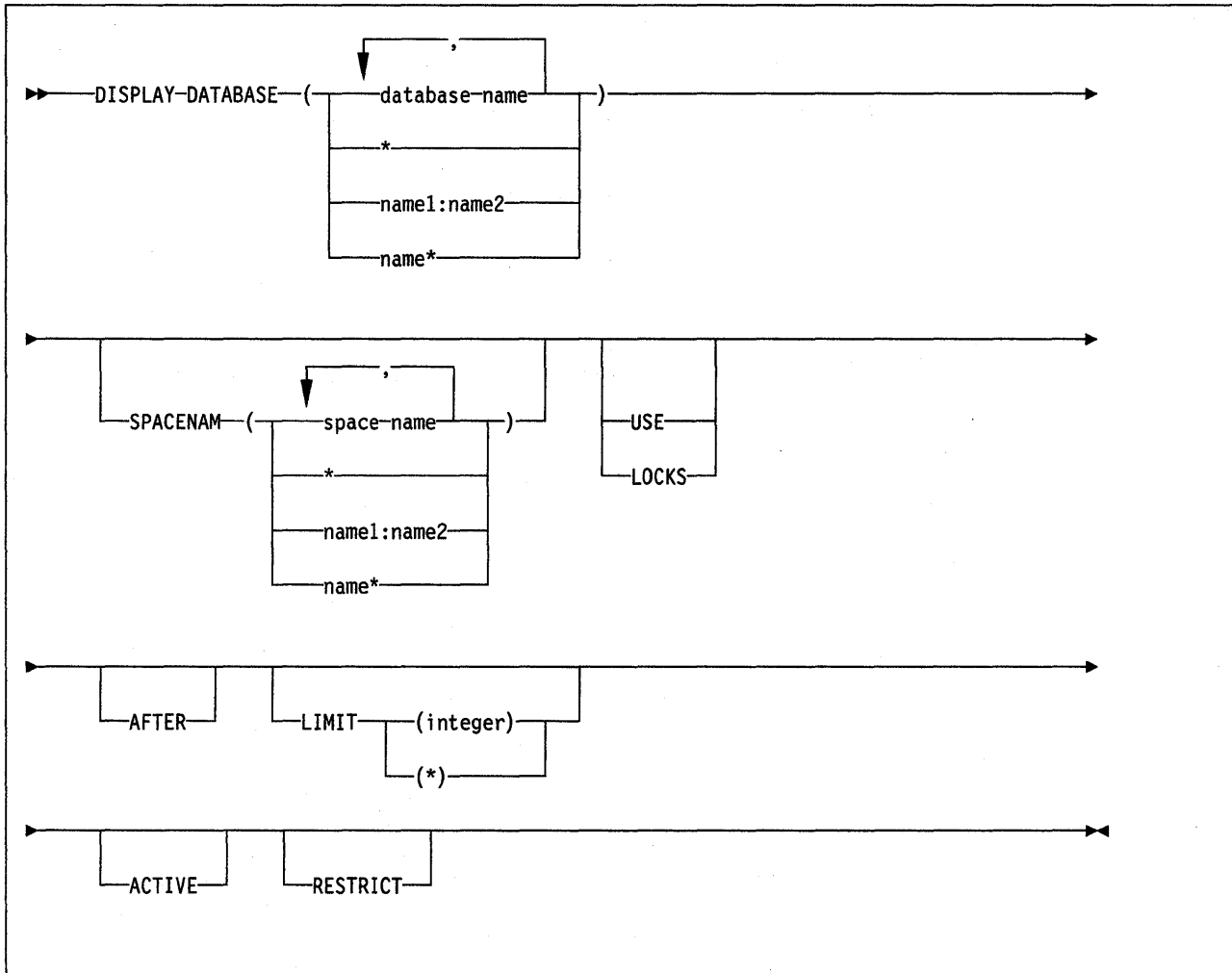
The privilege to display the status of all databases requires no privilege. The resulting display lists those databases for which the privilege set defined below has the `DISPLAYDB` privilege.

The privilege to display the status of a list of databases requires no privilege. The resulting display lists those databases for which the privilege set has the `DISPLAYDB` privilege. Error messages are produced for those specified databases for which the privilege set does not have the `DISPLAYDB` privilege.

The `DISPLAYDB` privilege may have been explicitly granted, or may be inherent in another privilege; it is inherent in the following authorities:

- `SYSADM` authority
- `SYSOPR` authority
- `DBADM`, `DBCTRL`, or `DBMAINT` authority for the database.

-DISPLAY DATABASE (DB2)



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see "DB2 Command Parsing" on page 11.

(database-name)

Identifies one or more databases whose status is to be displayed. *database-name* may have any of the forms in the following list (where *name1* and *name2* represent any strings of from 1 to 8 characters, and *name* represents any string of from 1 to 7 characters):

Form **Displays the status of...**

name1 The database *name1*

name1:name2

All databases whose names collate greater than or equal to *name1* and less than or equal to *name2*

*name**

All databases whose names begin with the string *name*

(*)

Displays information on all databases defined to the DB2 subsystem for which the privilege set has the required authorization.

The following parameters are optional.

SPACENAM

Tells what to display. This parameter is optional, but, if used, also specify the corresponding database name. You cannot use SPACENAM if more than one database is specified.

Abbreviation: SPACE

(*)

Displays information about all table spaces and indexes of the specified database.

(*space-name*)

Lists one or more spaces whose status is to be displayed. *space-name* may be written like *database-name* to designate:

- The name of a single table space or index space
- A range of names
- A partial name

USE

Displays information about correlation-ids and connection-ids for all applications allocated to spaces whose status is displayed.

LOCKS

Displays information about locks for all table spaces and tables whose status is displayed.

LOCKS overrides USE; if both are used, USE is ignored.

AFTER

Cannot be used with more than one database name, or with more than one table space or index space name.

If only a database name is used, AFTER continues the display to 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.

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.

(*)

Limits the display to 12K bytes, the size of the message buffer.

ACTIVE

Limits the display to table spaces or index spaces that have been allocated to applications, 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 index spaces whose use is restricted.

Abbreviation: RES

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

1. It is started for read-only processing.
2. It is started for utility-only processing.
3. It is currently being processed by a utility.
4. It is stopped.
5. It contains a restricted table space or index space.

A table space or index space is restricted if:

1. It is in one of situations 1 through 4, above.
2. It is waiting for deferred restart.
3. It is in copy pending, check pending, or recover pending status.

Usage Notes

The message DSN9022I will be issued to indicate that processing is complete.

The DB2 catalog tables can always be displayed. 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. This will result in an error. See the *SQL Reference* for a list of table space names and assigned tables.

If the LOCK and USE keywords are omitted, message DSNT392I is issued to display the following status information:

CHKP	The table space is in CHECK PENDING state.
COPY	An image copy is required for this object.
DEFER	The data base is marked for deferred restart.
INDBT	In doubt processing is required.
RECP	The table space or index space is in RECOVER PENDING state.
REST	The data base is currently under restart processing.
RO	The data base is started for read activity only.
RW	The data base is started for read and write activity.
STOP	The data base is stopped.
STOPE	The data base is being stopped due to a connect failure.
STOPP	A stop is pending or is in progress for this data base.
UT	The data base is started for utility processing only.
UTRO	Although the data base is started as RW, a utility is in process and only RO access is allowed.
UTRW	The data base is started as RW and a utility is in process.
UTUT	Although the data base is started as RW, a utility is in process and only UT access is allowed.

Examples

Example 1: Display information about database DSN8D21A. USE causes the correlation-id and connection-name information to be displayed. Display information only about table space DSN8S21D.

```
-DISPLAY DATABASE (DSN8D21A)  
  USE  
  SPACENAM (DSN8S21D)
```

Example 2: Display information about all databases that are in a status that restricts their use.

```
-DISPLAY DATABASE (*)  
  RESTRICT
```

Note that this selection criteria is based on the database only; the table spaces are not searched.

Example 3: Display information about all databases that containing objects that restrict their use.

```
-DISPLAY DATABASE (*) SPACENAM (*)  
  RESTRICT
```

-DISPLAY RLIMIT (DB2)

The -DISPLAY RLIMIT command 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.

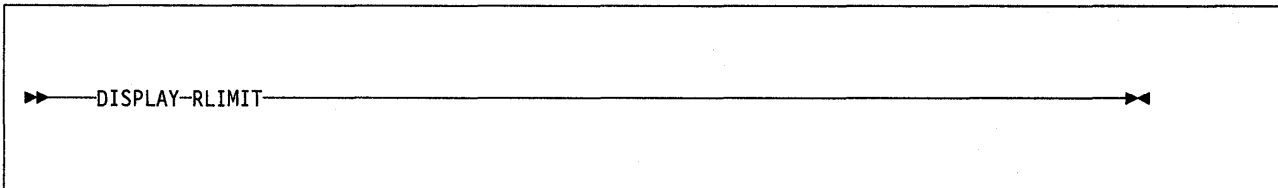
Abbreviation: -DIS RLIM

Environment

This command can be entered from an MVS console, from the DSN command processor, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

To issue this command, the privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, must include SYSADM or SYSOPR authority.



-DISPLAY THREAD (DB2)

The DB2 -DISPLAY THREAD command displays information about DB2 connections.

The information displayed is one of the following:

- All active or indoubt connections having the specified name
- All active or indoubt connections having the connection name of the issuer
- All active or indoubt connections (if issued by the MVS console operator).

Note: The information returned by the -DISPLAY THREAD command reflects a dynamic status. By the time the information is displayed, the status may have changed. Moreover, the information is consistent only within one address space and is *not necessarily* consistent across all address spaces displayed.

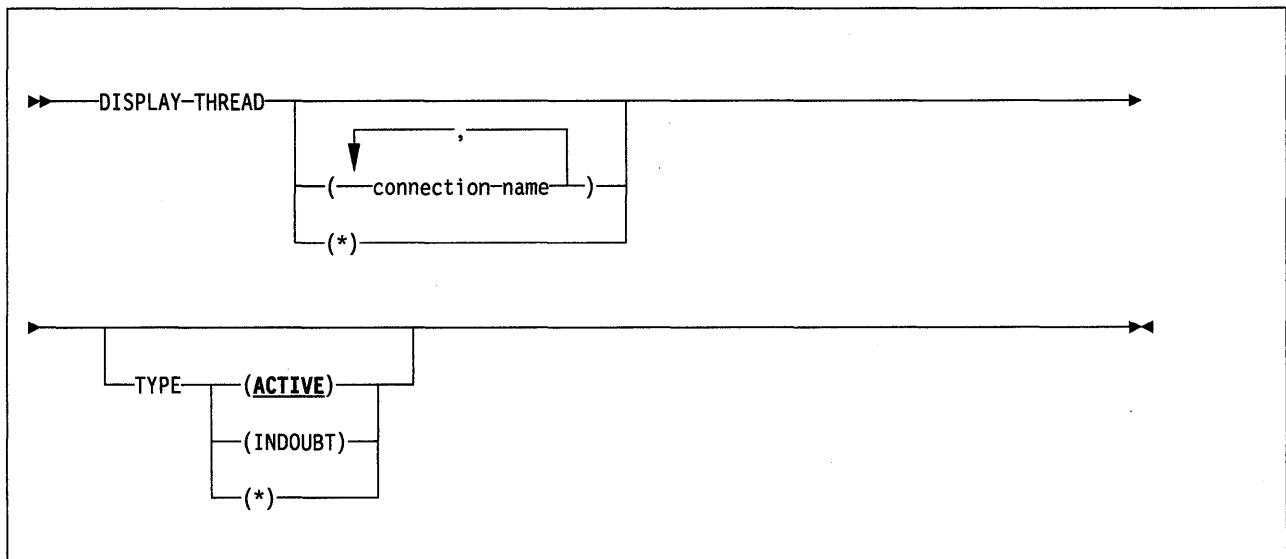
Abbreviation: -DIS THD

Environment

This command can be entered from an MVS console, from the DSN processor under TSO, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

To issue this command, the privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, must include the SYSADM or SYSOPR authorities, or the DISPLAY privilege.



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see "DB2 Command Parsing" on page 11.

The following parameters are optional.

-DISPLAY THREAD (DB2)

(*connection-name*)

Lists one or more connection-names (of one to eight characters each). Threads are selected only from the address spaces associated with those connection-names.

(*)

Displays all connections in all address spaces attached to DB2.

The **default** is to display only the connections associated with the transaction manager from which the command was entered.

If the command is entered from an MVS console, a connection-name or *; *must* be used; no default is available.

TYPE

Tells what type of thread to display.

Abbreviation: T

(ACTIVE)

Displays only active threads. If, during command processing, an active thread becomes indoubt, it may appear twice—once as active and once as indoubt.

Abbreviation: A

The **default** is **ACTIVE**.

The information produced by ACTIVE may be useful for debugging purposes, especially messages DSNV403I and DSNV404I; the contents of those messages are described in *Messages and Codes*.

(INDOUBT)

Displays only indoubt threads. External intervention is needed to resolve their status: They may have been indoubt at the last restart, or may have become indoubt since the last restart.

Abbreviation: I

(*)

Displays both active and indoubt threads.

Example

Example: Display all current indoubt threads with the same connection name as the user of the command.

```
-DISPLAY THREAD  
TYPE (INDOUBT)
```

-DISPLAY TRACE (DB2)

The DB2 -DISPLAY TRACE command displays a list of active traces. For more information about this trace facility, refer to Section 6 of *System and Database Administration Guide* and *Diagnosis Guide and Reference*.

Abbreviation: -DIS TRACE

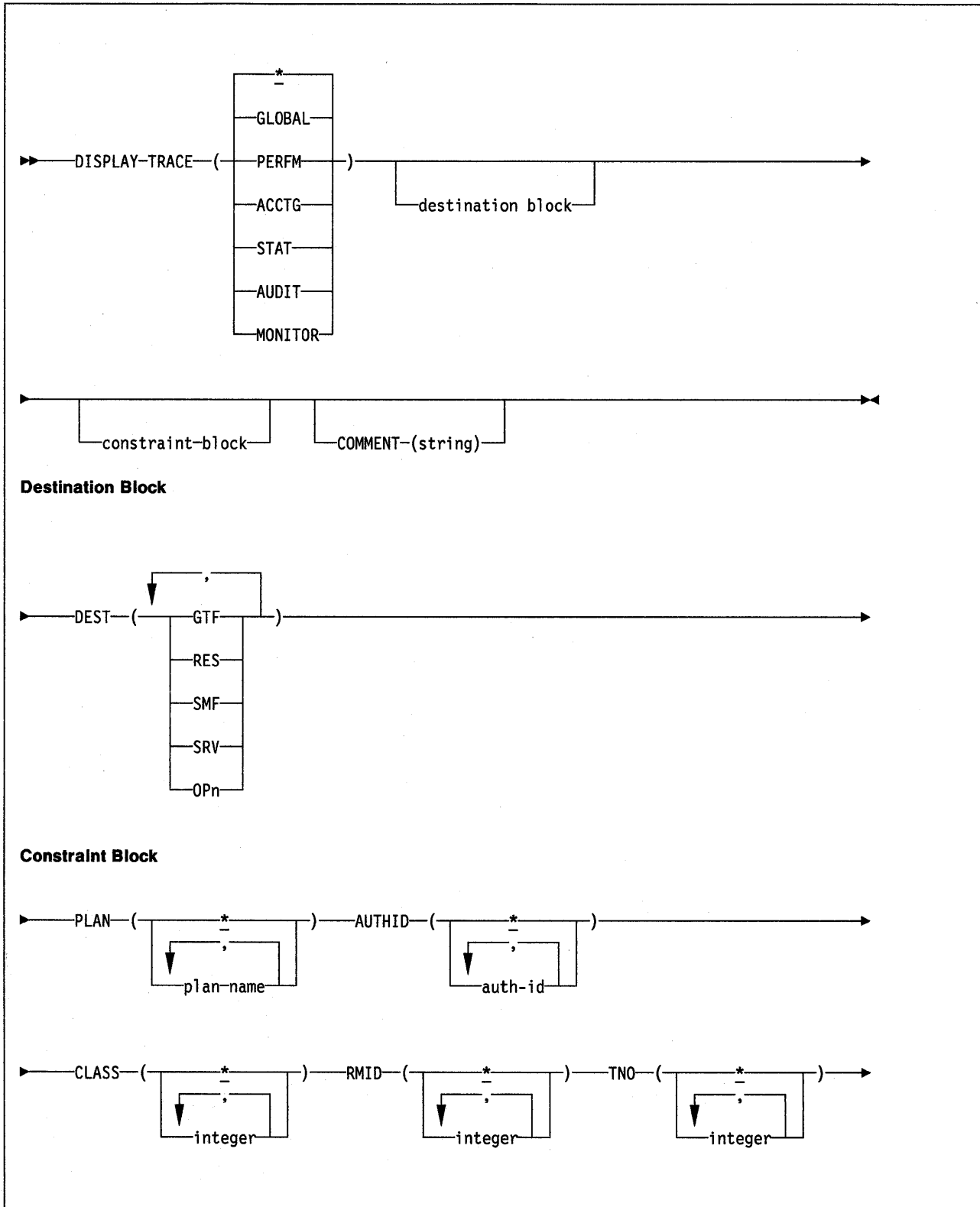
Environment

This command can be entered from an MVS console, from the DSN command processor, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

To execute this command, the privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, must include the SYSADM or SYSOPR authority, or the DISPLAY privilege.

-DISPLAY TRACE (DB2)



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see "DB2 Command Parsing" on page 11.

All parameters are optional. The command -DISPLAY TRACE lists all active traces. Each option that 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 fuller descriptions of each type, see "-START TRACE (DB2)" on page 129.

Type	Description
GLOBAL	Service data from the entire DB2 subsystem Abbreviation: G
PERFM	Performance records of specific events Abbreviation: P
ACCTG	Accounting records for each transaction Abbreviation: A
STAT	Statistical data Abbreviation: S
AUDIT	Audit data Abbreviation: AU
MONITOR	Monitor data Abbreviation: MON

COMMENT (*string*)

Gives a comment. -DISPLAY TRACE command (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. The comment does not appear in the display; it may 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 a value is specified, the list is not limited.

Abbreviation: D

Possible values and their meanings are:

Value	Trace destination
GTF	The Generalized Trace Facility
RES	A wrap-around table residing in CSA storage
SMF	The System Management Facility

-DISPLAY TRACE (DB2)

SRV A serviceability routine reserved for problem diagnosis

OPn A specific destination. *n* can be a value from 1 to 8.

See “-START TRACE (DB2)” on page 129 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 8 plan names can be used. If more than one name is used, only one value can be used for AUTHID. 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 8 identifiers can be used. If more than one identifier is used, only one value may be used for PLAN. 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 129.

The **default** is **CLASS(*)**, which does not limit the list.

RMID (*integer*)

Limits the list to traces started for particular resource managers. For descriptions of the allowable classes resource manager identifiers, see “-START TRACE (DB2)” on page 129. Do not use this option with STAT or ACCTG.

The **default** is **RMID(*)**, which does not limit the list.

TNO

Limits the list to particular traces, identified by their trace numbers (1 to 32, 01 to 09). Up to 8 trace numbers can be used. If more than one number is used, only one value each for PLAN and AUTHID can be used.

The **default** is **TNO(*)**, which does not limit the list.

Output

Output from -DISPLAY TRACE is a set of messages (like the following example) that shows which traces are active.

TNO	TYPE	CLASS	AUTHID	PLAN	DEST	RMID
1	GLOBAL	*	*	*	GTF	*
2	GLOBAL	*	SYSADM01	PLAN8888	GTF	07,14
3	ACCTG	*	*	*	GTF	*
4	STAT	*	*	*	GTF	*
5	PERFM	01,02,03,04	SYSADM01	*	GTF	*
6	AUDIT	*	*	*	SMF	*
7	MON	1	*	*	OP1	*

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`. (The `-START TRACE` command appears under “`-START TRACE (DB2)`” on page 129.)

`-DISPLAY TRACE (ACCTG) PLAN (DSN8BC21)
COMMENT ('ACCTG TRACE FOR DSN8BC21')`

Example 3: There are only 32 active traces, at most. List all of them.

`-DISPLAY TRACE`

Example 4: List all active performance traces.

`-DISPLAY TRACE=P`

Example 5: List all active monitor traces.

`-DISPLAY TRACE (MON)`

-DISPLAY UTILITY (DB2)

The DB2 -DISPLAY UTILITY command displays the status of utility jobs. A job can be active, stopped, or terminating.

The output from -DISPLAY UTILITY consists of informational messages only. One set of messages is returned for each job identified by the command.

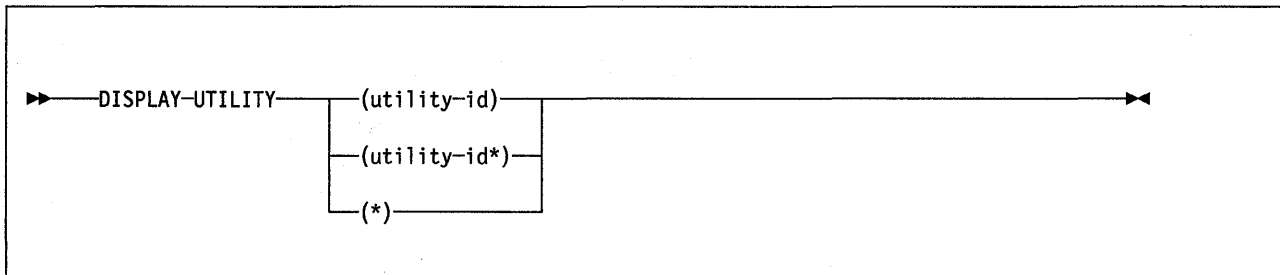
Abbreviation: -DIS UTIL

Environment

This command can be entered from an MVS console, from the DSN processor, from a DB2I panel, or from an IMS/VS or CICS terminal.

Authorization

None is required.



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see “DB2 Command Parsing” on page 11.

Use at least one of the following parameters 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 CLIST, on page 93.

If *utility-id* was omitted, *utility-id* has the form *userid.jobname*.

(partial-id*)

Identifies a set of utility jobs. A status message is shown for each utility identifier that begins with the characters of *partial-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.

Output

The output from -DISPLAY UTILITY consists of informational messages only. One set of messages is returned for each job identified by the command.

Usage Notes

The status displayed in the returned message is the status at the time the DB2 utility function received the command. Execution has proceeded, so 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, by the time the message is received by the requestor, the utility job step may have terminated so that the utility identifier is no longer known to DB2.

Output

An example of the -DISPLAY UTILITY command's output is shown below.

```
DSNU100I - DSNUGDIS - USERID = SAMPID
          UTILID = RUNTS
          PROCESSING UTILITY STATEMENT 1
          UTILITY = RUNSTATS
          PHASE = RUNSTATS   COUNT = 0
          STATUS = STOPPED
DSN9022I - DSNUGCC '-DISPLAY UTILITY' NORMAL COMPLETION
```

Example

Example: Display status information for all utility jobs currently known to DB2.

-DISPLAY UTILITY (*)

DSN (TSO)

The TSO command DSN invokes the DSN command processor, which processes the following subcommands:

ABEND
BIND
DCLGEN
END
FREE
REBIND
RUN
SPUFI

* Comment subcommand
— DB2 commands (to DSN, these are subcommands)

The ABEND subcommand listed above is described in *Diagnosis Guide and Reference*. Use it only when diagnosing a problem with DSN or DB2.

The DSN command processor also allows for issuing TSO commands, except for FREE, RUN, TEST, and TIME. To use TSO TEST to debug an application program, invoke it along with the DSN command processor (for example,

TEST 'DSN210.DSNLOAD(DSN)' CP).

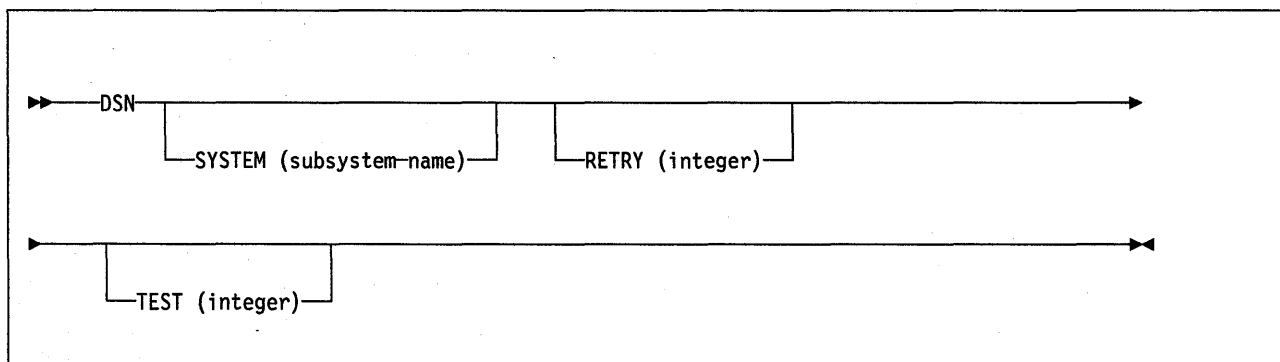
Environment

DSN runs under TSO in either foreground or background mode. Note that, when run in background mode, DSN cannot prompt for corrections or additional required information.

DSN can also be invoked from a CLIST running in either foreground or background mode. DB2I invokes the DSN command processor to do most of its work.

Authorization

None is required for the DSN command, but authorization is required for most subcommands.



Keyword and Parameter Descriptions

The following keywords and parameters are optional.

SYSTEM

For *subsystem-name*, substitute the name of the DB2 subsystem.

The **default** is DSN. This may be modified when DB2 was installed.

RETRY

For *integer*, specify the number of additional times connection to the DB2 subsystem should be attempted if DB2 is not up when DSN is issued. Retries occur at 30-second intervals.

Default: The default number of retries is 0. The maximum number of retries is 120.

TEST

Starts the DSN trace facility. For further information on tracing execution of the DSN command processor, see *Diagnosis Guide and Reference*.

Usage Notes

Giving the DSN command begins a DSN session, during which the user can enter DSN subcommands. These rules govern the session:

- In foreground operation, the DSN processor prompts the user for input by displaying the prompt string 'DSN' at the terminal. In background mode, it reads from the SYSTSIN data set.
- Except for delimited table names in the DCLGEN command, input in lowercase letters is changed to uppercase.
- If ATTENTION (PA1) is pressed while DSN is running, 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 C is entered, 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 DSN is invoked from a CLIST, or a CLIST invoked under DSN, CONTROL PROMPT must be specified in the CLIST in order to receive message DSNE005.

- When DSN finishes processing a subcommand, it again prompts for input. That cycle continues until you end the session.
- You may end the session by doing one of the following:
 - Giving the END command. Control passes to TSO.
 - Pressing ATTENTION and responding to the message by pressing ATTENTION again.
 - Issuing another DSN command. The old session ends and a new one begins.

DSN (TSO)

Examples

Example 1: Invoke the DSN command processor. 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.

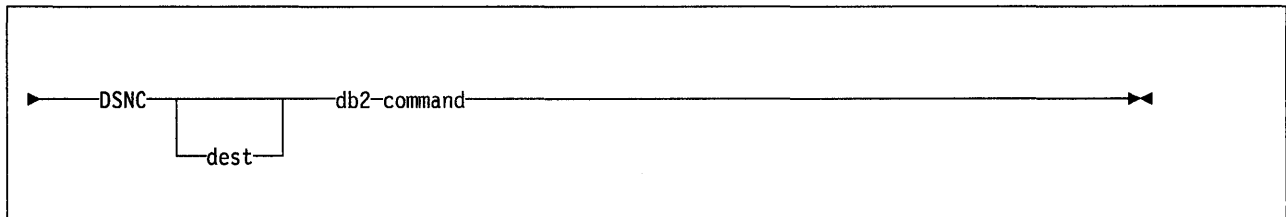
Environment

This command originates from a CICS terminal.

Authorization

This command requires the appropriate level of CICS authority, as described in the *CICS/OS/VS: Installation and Operations Guide*.

Entering the DSNC command requires no privileges. For a description of the privileges required to issue a DB2 command using the DSNC command, see the command's description.



Parameter Descriptions

dest

This optional parameter specifies the destination to which output is to be directed. For instance, if you specify the name of a device, the output that is generated from the DB2 command will be sent to that device.

db2-command

This parameter is the exact DB2 command that you want to enter from a CICS terminal. It must be preceded by a hyphen.

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.

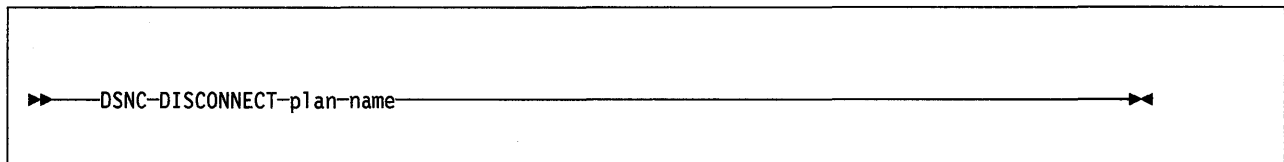
Abbreviation: DSNC DISC

Environment

This command originates from a CICS terminal.

Authorization

This command requires an appropriate level of CICS authority, as described in the *CICS/OS/VS: Installation and Operations Guide*.



Keyword and Parameter Descriptions

plan-name

Names a valid application plan.

Usage Notes

The DSNC DISCONNECT command does not prevent threads from being created on behalf of transactions. It only causes currently connected threads to be terminated as soon as they are not being used by a transaction.

You may also need to stop the transactions associated with a particular plan ID by means of the CICS master terminal STOP command to prevent new instances of the transaction from causing a re-creation of a thread.

Example

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).

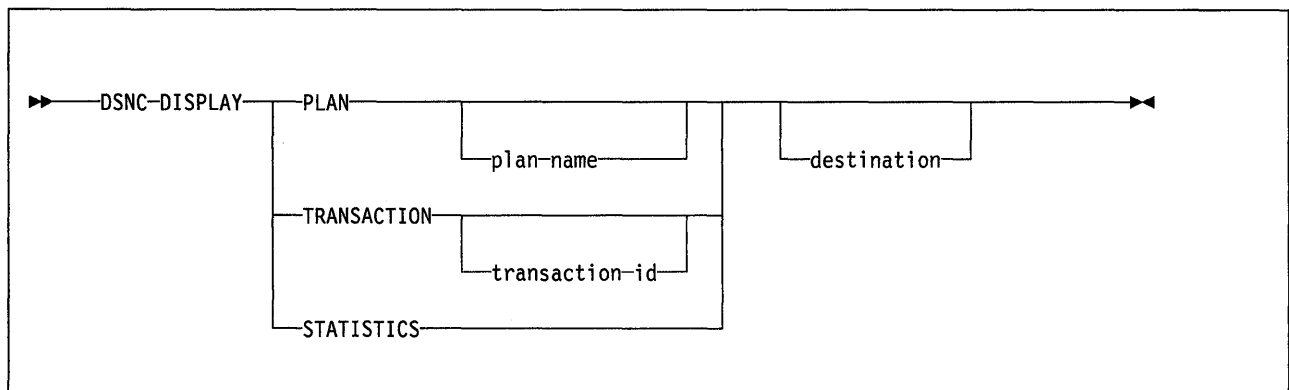
Abbreviation: DSNC DISP

Environment

This command originates from a CICS terminal.

Authorization

This command requires an appropriate level of CICS authority, as described in the *CICS/OS/VS: Installation and Operations Guide*.



Keyword and Parameter Descriptions

PLAN *plan-name*

Displays information about transactions by plan name.

plan-name is a valid plan name for which information will be 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 will be displayed. For a group transaction entry in the resource control table, you may 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

DSNC DISPLAY (CICS)

destination

Is the identifier of another terminal to receive the requested display information. It must be a valid terminal listed in the CICS Terminal Control Table (TCT) and supported by CICS Basic Mapping Support (BMS).

Usage Notes

Because the optional destination may be 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 in order to display the desired information at the desired terminal.

Examples

Example 1: Display information on all 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 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
```

DSNC MODIFY (CICS Attachment Facility)

The DSNC MODIFY command modifies the ERRDEST entry in the resource control table (RCT), or the maximum active thread value associated with a given transaction or group name.

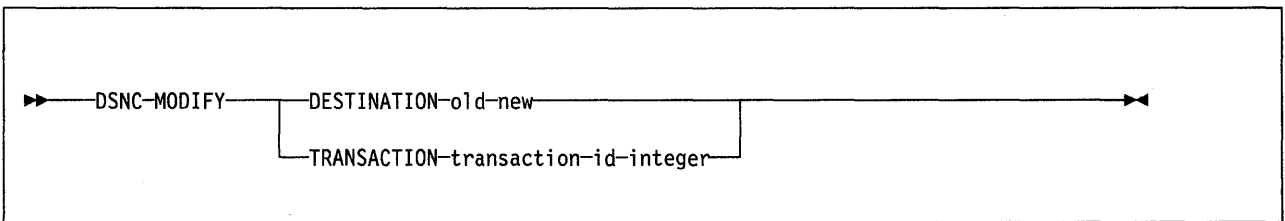
Abbreviation: DSNC MODI

Environment

This command originates from a CICS terminal.

Authorization

This command requires an appropriate level of CICS authority, as described in the *CICS/OS/VS: Installation and Operations Guide*.



Keyword and Parameter 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 may be entered to identify the entry.

integer

Is a new maximum value.

DSNC MODIFY (CICS)

Usage Notes

The DSNC MODIFY TRANSACTION command will change the value of the THRDA parameter of the DSNCRCT TYPE = ENTRY macro. The *integer* specified in the DSNC MODIFY TRANSACTION command cannot be larger than the value specified for the THRDM parameter of the DSNCRCT TYPE = ENTRY macro. 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 *System and Database Administration Guide*.

Example

Example: Change the specification of the ERRDEST parameter in the resource control table from MTO1 to MTO2.

```
DSNC MODI DEST MT01 MT02
```


DSNC STOP (CICS attachment facility)

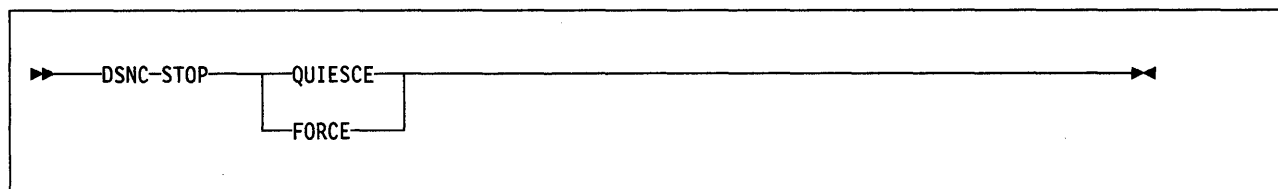
The CICS attachment facility command DSNC STOP stops the attachment facility.

Environment

This command originates from a CICS terminal.

Authorization

This command requires an appropriate level of CICS authority, as described in the *CICS/OS/VS: Installation and Operations Guide*.



Keyword and Parameter 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 may be running.

Usage Notes

Using FORCE will leave threads in an indoubt situation. Restarting may require reconnection of CICS and DB2 in order to resolve any indoubt situations.

Example

Example: Stop the CICS attachment facility.

```
DSNC STOP FORCE
```

DSNC STRT (CICS attachment facility)

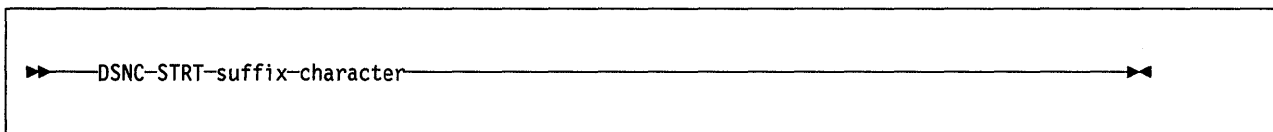
The CICS attachment facility command DSNC STRT starts the attachment facility, allowing CICS application programs to access DB2 databases.

Environment

This command originates from a CICS terminal.

Authorization

This command requires an appropriate level of CICS authority, as described in the *CICS/OS/VS: Installation and Operations Guide*.



Keyword and Parameter Descriptions

suffix-character

Specifies that the CICS attachment facility is to be started. The resource control table loaded during startup will be DSNCRCT0 by default unless for *suffix-character* you substitute another character that is appended to DSNCRCT, replacing the 0. (For example, if you specify 1 as a suffix character, module DSNCRCT1 will be loaded.)

Example

Example: Start the CICS attachment facility, using DSNCRCTA.

```
DSNC STRT A
```

DSNH (TSO CLIST)

The DSNH command procedure (a TSO CLIST) allows you to prepare an application program for execution, and execute it under TSO, by giving a single command.

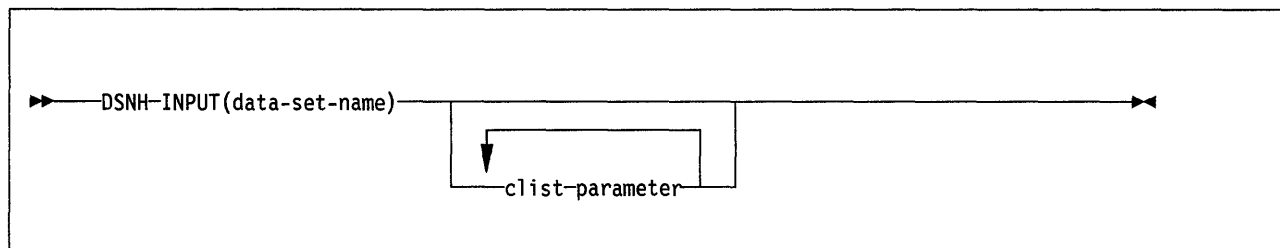
Environment

The DSNH CLIST can be 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.

Authorization

If you specify the BIND(YES) parameter, the DSN BIND subcommand will be executed. See "BIND (DSN)" on page 28 for a description of the privileges necessary to bind a plan.

If you specify the RUN(YES) parameter, the DSN RUN subcommand will be executed. See "RUN (DSN)" on page 115 for a description of the privileges necessary to run a plan.



DSNH processing is a sequential process that may include any of the following steps, taken in the order shown:

- MP—PL/I macro processing
- PC—precompilation
- TR—CICS translation
- BI—binding
- CO—compilation
- PL—pre-linkedit
- LE—linkediting
- RU—running

Any steps in the process that are skipped must have been previously completed successfully by DSNH. You can end the process at any point you choose.

The CLIST parameters described below provide the processing options for each step; specify them when you invoke DSNH. Some parameters are used for more than one step: The list that follows shows where each parameter is used. An entry "Y" in any row and column shows that the option listed at the beginning of the row is used in the step whose number appears at the top of the column.

DSNH (TSO CLIST)

STEPS =	MP	PC	TR	BI	CO	LE	RU	PL
OPTIONS	-	-	-	-	-	-	-	-
ACQUIRE	-	-	-	Y	-	-	*	-
ACTION	-	-	-	Y	-	-	-	-
ASMHLIB	-	-	-	-	Y	-	-	-
ASMHLOAD	-	-	-	-	Y	-	-	-
ASMLIB	-	-	-	-	Y	-	-	-
ASMLOAD	-	-	-	-	Y	-	-	-
BIND	-	-	-	Y	-	-	-	-
BLIB	-	-	-	Y	-	-	-	-
BnLIB	-	-	-	Y	-	-	-	-
BMEM	-	-	-	Y	-	-	-	-
CCLLIB	-	-	-	-	-	Y	-	-
CCLOAD	-	-	-	-	Y	-	-	-
CCMSGs	-	-	-	-	Y	-	-	Y
CCPLIB	-	-	-	-	-	Y	-	-
CCSLIB	-	-	-	-	Y	-	-	-
CCULIB	-	-	-	-	Y	-	-	-
CICSOPT	-	-	Y	-	-	-	-	-
CICSPRE	-	-	Y	-	-	Y	-	-
CICSXLAT	-	-	Y	-	-	-	-	-
CLIB	Y	-	-	-	Y	-	-	-
CnLIB	Y	-	-	-	Y	-	-	-
COBLIB	-	-	-	-	-	Y	-	-
COBLOAD	-	-	-	-	Y	-	-	-
COB2CICS	-	-	-	-	-	Y	-	-
COB2LIB	-	-	-	-	-	Y	-	-
COB2LOAD	-	-	-	-	Y	-	-	-
COMPILE	-	-	-	-	Y	-	-	-
CONTROL	Y	Y	Y	Y	Y	Y	Y	-
COPTION	Y	-	-	-	Y	-	-	-
DATE	-	Y	-	-	-	-	-	-
DBRMLIB	-	Y	-	Y	-	-	-	-
DECIMAL	-	Y	-	-	*	-	-	-
DELIMIT	-	Y	Y	-	Y	-	-	-
ENTRY	-	-	-	-	-	Y	-	-
EXPLAIN	-	-	-	Y	-	-	-	-
FLAG	Y	Y	Y	Y	Y	-	-	-
FORTLIB	-	-	-	-	-	Y	-	-
FORTLOAD	-	-	-	-	Y	-	-	-
GRAPHIC	-	Y	-	-	-	-	-	-
HOST	Y	Y	Y	*	Y	Y	Y	-
IMSPRE	-	-	-	-	-	Y	-	-
INPUT	Y	Y	*	Y	Y	Y	Y	-
ISOLATION	-	-	-	Y	-	-	-	-
LINECOUNT	Y	Y	Y	-	Y	-	-	-
LINK	-	-	-	-	-	Y	-	-

STEPS =	MP	PC	TR	BI	CO	LE	RU	PL
OPTIONS	-	-	-	-	-	-	-	-
LLIB	-	-	-	-	-	Y	-	-
LnLIB	-	-	-	-	-	Y	-	-
LOAD	-	-	-	-	-	Y	Y	-
LOPTION	-	-	-	-	-	Y	-	-
MACRO	Y	-	Y	-	-	-	-	-
OPTIONS	-	Y	Y	-	-	Y	Y	-
OUTNAME	Y	Y	-	-	Y	Y	Y	-
PARMS	-	-	-	-	-	-	Y	-
PASS	-	Y	-	-	-	-	-	-
PCLOAD	-	Y	-	-	-	-	-	-
PLAN	-	-	-	Y	-	-	Y	-
PLIB	-	Y	-	-	-	-	-	-
PnLIB	-	Y	-	-	-	-	-	-
PLILIB	-	-	-	-	-	Y	-	-
PLILOAD	Y	-	-	-	Y	-	-	-
PRECOMP	-	Y	-	-	-	-	-	-
PRINT	Y	Y	Y	-	Y	Y	-	-
PSECSPAC	Y	Y	Y	-	Y	-	Y	-
PSPACE	Y	Y	Y	-	Y	-	Y	-
RCTERM	Y	Y	Y	Y	Y	Y	Y	-
RELEASE	-	-	-	Y	-	-	-	-
RETAIN	-	-	-	Y	-	-	-	-
RUN	-	Y	Y	-	-	Y	Y	-
RUNIN	-	-	-	-	-	-	Y	-
RUNOUT	-	-	-	-	-	-	Y	-
SOURCE	Y	Y	Y	-	Y	-	-	-
SPACEUN	Y	Y	Y	-	Y	-	Y	-
SQLDELIM	-	Y	-	-	-	-	-	-
SUFFIX	Y	Y	-	-	-	-	-	-
SYSTEM	-	-	-	Y	-	-	Y	-
TERM	Y	Y	-	-	Y	Y	-	-
TIME	-	Y	-	-	-	-	-	-
VALIDATE	-	-	-	Y	-	-	-	-
WORKUNIT	Y	Y	Y	-	Y	-	-	-
WSECSPAC	Y	Y	Y	-	Y	-	-	-
WSPACE	Y	Y	Y	-	Y	-	-	-
XLIB	-	-	-	-	-	Y	-	-
XREF	Y	Y	-	-	Y	Y	-	-

* The option is not an input parameter for this step, but indirectly affects the step.

Parameter Descriptions

The only parameter required on the DSNH statement is INPUT; the others are optional. In the alphabetic list of parameters that follows, default values, if any, are underlined.

In general, parameter values do NOT need to be enclosed between apostrophes. But there are these exceptions:

- If the value is a list of tokens with separators, it must be enclosed between apostrophes.
- If the value is a data set name, it will normally have your user identifier prefixed to it. To avoid the prefix, you must enclose the name between sets of 3 apostrophes.

Data Set Names: In most cases, parameter values that are data set names (*dsname*) cannot include member names. Exceptions are noted in the parameter descriptions that follow.

Note: When DB2 was installed, the defaults may have been changed to names specific to your site.

ACQUIRE

Tells the BIND process when to acquire DB2 resources.

(USE)

Acquires resources when they are first used. The **default** is ACQUIRE(USE).

(ALLOCATE)

Acquires resources when the plan is allocated. Do not use ACQUIRE(ALLOCATE) with RELEASE(COMMIT).

ACTION

Tells whether the application plan produced by the BIND process is new or is a replacement for an existing application plan.

(REPLACE)

Specifies that the plan is to replace an existing application plan, if one exists with the same name, or to add a new plan. The **default** is ACTION(REPLACE).

(ADD)

Specifies a new application plan.

ASMHLIB(*dsname*)

Names a data set to be used as the standard MACLIB for Assembler H. The **default** is ASMHLIB('SYS1.MACLIB').

ASMHLOAD(*dsname*)

Names a data set that contains the Assembler H load module. *dsname* may include a member name. The **default** is ASMHLOAD('SYS1.LINKLIB(IEV90)').

ASMLIB(*dsname*)

Names a data set to be used as the standard MACLIB for Assembler. The **default** is ASMLIB('SYS1.MACLIB').

ASMLOAD(*dsname*)

Names a data set that contains the Assembler load module. *dsname* may include a member name. The **default** is ASMLOAD('SYS1.LINKLIB(IEUASM)').

BIND

Tells whether to invoke the BIND process if the previous steps completed successfully.

Use (YES) or (NO). The **default** is **BIND(YES)**.

BLIB(dsname)

BnLIB(dsname)

Names a data set that contains DBRMs to be included by the BIND process. The parameters BnLIB (where n may be 2, 3, 4, 5, 6, 7, or 8) are extensions of BLIB, used to simplify passing a list of data set names.

Use (NONE) to name no data set. The **defaults** are **BLIB(NONE)** and **BnLIB(NONE)**.

BMEM(member-list)

Gives a list of additional DBRM member names to be included by the BIND process.

Use (NONE) to give no additional member names. The **default** is **BMEM(NONE)**.

CCLLIB(dsname)

Names the linkage editor include library to be used for C routines. The **default** is **CCLLIB("EDC.CBASE")**.

CCLOAD(dsname)

Names a data set that contains the C compiler load module. The **default** is **CCLOAD("EDC.VSCC.LOAD(EDCCOMP)")**.

CCMSGSG(dsname)

Names a data set that contains the C messages. The **default** is **CCMSGSG("EDC.VSCC.CCCMSGSG")**.

CCPLIB(dsname)

Names a data set that contains linkage editor include modules for the linkedit step. The **default** is **CCPLIB("EDC.PLIBASE")**.

CCSLIB(dsname)

Names the data set that contains the C compiler headers. The **default** is **CCSLIB("EDC.VSCC.HDRS")**.

CCULIB(dsname)

Names a data set that contains C source statements to be included in the program during the compilation step. The **default** is **CCULIB(NONE)**.

CICSOPT(option-list)

Gives a list of additional CICS translator options. See *CICS/VS Application Programmer's Reference Manual (Command Level)* for information about translator options.

Use (NONE) to give no additional options. The **default** is **CICSOPT(NONE)**.

CICSPRE(prefix)

Gives the prefix for the CICS libraries. The library names will be:

Name	Library
prefix.LOADLIB	translators
prefix.PL1LIBn	PL/I include library
prefix.COBLIB	COBOL include library

The **default** is **CICSPRE(CICS170)**.

CICSXLAT

Tells whether to invoke the CICS command translator. The option is effective only if you use RUN(CICS). You may not use the option with the MARGINS option of the translator.

Use (YES) or (NO). The default is **CICSXLAT(YES)**.

Note: The DB2I panel default is **CICSXLAT(NO)**.

CLIB(dsname)**CnLIB(dsname)**

May name a data set that contains host language source statements to be included by the compiler or assembler. The parameters CnLIB (where *n* may be 2, 3, or 4) are extensions of CLIB, used to simplify passing a list of data set names.

Use (NONE) to name no data set. The defaults are **CLIB(NONE)** and **CnLIB(NONE)**.

COB2CICS(dsname)

Names the linkage editor include library to be used for COBOL II CICS routines. The default is **COB2CICS("SYS1.COB2CICS")**.

COB2LIB(dsname)

Names the linkage editor include library to be used for COBOL II routines. The default is **COB2LIB("SYS1.COB2LIB")**.

COB2LOAD(dsname)

Names a data set that contains the VS COBOL II compiler load module. *dsname* may include a member name. The default is **COB2LOAD("SYS1.COB2COMP(IGYCRCTL)")**.

COMPILE

Tells whether to invoke the compiler or assembler if the precompile step is successful.

Use YES) or (NO). The default is **COMPILE(YES)**.

CONTROL

Tells whether to trace the CLIST execution This can be useful for problem diagnosis.

(NONE) Omits tracing. The default is **CONTROL(NONE)**.

(LIST) Displays TSO commands after substitution for symbols and before command execution.

(CONLIST) Displays CLIST commands after substitution for symbols and before command execution.

(SYMLIST) Displays all executable statements (TSO commands and CLIST statements) before substitution for symbols.

COPTION(string)

Gives a list of compiler or assembler options. For more information, refer to the manual that describes the compiler or assembler options for the specific language you are using. For a list of restrictions on some options, see "COBOL options" on page 84.

Use (NONE) to give no options. The default is **COPTION(NONE)**.

DATE(*format*)

Specifies the format of DATE values that are output from DB2, overriding the format specified when DB2 is installed. *format* is the DATE output format, which may be ISO, JIS, USA, EUR, or LOCAL. The **default** is the value supplied when DB2 was installed, and is written in the data-only load module, DSNHDECP.

DBRMLIB(*dsname*)

Names the partitioned data set that will contain the DBRM as a member. *dsname* may not be DEFAULT; it may include a member name. The **default** is *outname*.DBRM, where *outname* is given by the OUTNAME parameter or its default (TEMP).

DECIMAL

Gives the decimal point indicator for decimal and floating point literals. DECIMAL is valid only for COBOL programs; DECIMAL(PERIOD) is forced for all other programs. The **default** is the value of the DECIMAL POINT field, set on the "DB2 INSTALLATION APPLICATION PROGRAMMING DEFAULTS" panel during installation.

(PERIOD)

Makes the indicator a period.

(COMMA)

Makes the indicator a comma.

DELIMIT

Sets the APOST or QUOTE precompiler option, to indicate the string delimiter used within host language statements. DELIMIT is effective only for COBOL programs; DELIMIT(APOST) is forced for all other programs.

(DEFAULT)

Designates the value chosen, during installation, for the STRING DELIMITER field on the APPLICATION PROGRAMMING DEFAULTS panel.

(APOST)

Names the apostrophe as the string delimiter for host language statements.

(QUOTE)

Names a quotation mark as the string delimiter for host language statements.

ENTRY(*entry-name*)

Names the entry point to be assigned by the linkage editor. The **default** depends on the host language and the value of RUN, as shown in the following table:

Language	RUN value	ENTRY value default
PL/I	IMS	PLICALLA
	CICS	NONE
	other	PLISTART

Language	RUN value	ENTRY value default
ASM or ASMH	IMS	DLITASM
COBOL or COBOL II	IMS	DLITCBL
other	any	NONE (no named entry point)

EXPLAIN

Sets the value of the EXPLAIN parameter for the BIND process. For a fuller explanation of EXPLAIN, see "BIND (DSN)" on page 28.

Use YES or NO.

NO

Provides no EXPLAIN information. The default is **EXPLAIN(NO)**.

YES

Inserts information in the table *userid.PLAN_TABLE*, which must have been defined by or for the user before the application program is bound.

FLAG

Tells 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 severe error messages. The default is **FLAG(I)**.

(W) Only warning, error, and severe error messages.

(E) Only error and severe error messages.

(C) Only severe error messages.

FORTLIB(dsname)

Names the linkage editor include library to be used for FORTRAN routines. The default is **FORTLIB("SYS1.VFORTLIB")**.

FORTLOAD(dsname)

Names a data set that contains the VS FORTRAN compiler load module. *dsname* may include a member name. The default is **FORTLOAD("SYS1.LINKLIB.(FORTVS)")**.

GRAPHIC

Specifies the value of the DSNHDECP MIXED option for the precompiler. See Section 2 of *System and Database Administration Guide* for more information about this option.

NONE

Indicates that the default specified during install should be used.

YES

Indicates that all character data may be mixed DBCS.

NO

Indicates that the data is not mixed DBCS.

The default is **GRAPHIC(NONE)**.

HOST

Defines the host language within which SQL statements are embedded. The default is the value of the LANGUAGE DEFAULT field, set on the "DB2 INSTALLATION APPLICATION PROGRAMMING DEFAULTS" panel during installation.

Use...	For...
(COBOL)	OS/VS COBOL
(COB2)	VS COBOL II
(ASMH)	Assembler H
(ASM)	Assembler
(FORTRAN)	VS FORTRAN
(PLI)	OS PL/I Optimizing Compiler
(C)	C

IMSPRE(prefix)

Sets the prefix for RESLIB, used for routines to be included by the linkage editor for IMS/VS. The default is **IMSPRE(IMSVS)**.

INPUT(dsname)

Names the data set that contains the host language source and SQL statements. *dsname* may include a member name. This name might also be used to build other names that you do not specify explicitly (such as the plan name for the PLAN parameter).

ISOLATION

Tells how far the application program is to be isolated from the effects of other executing applications:

(RR)

Repeatable Read. Indicates that database values read or changed by this application cannot be changed by other programs until this application commits or terminates. For more information on ISOLATION, refer to the section on locking in Section 3 of *System and Database Administration Guide*. The default is **ISOLATION(RR)**.

(CS)

Cursor Stability. Indicates that database values read by this application are only protected while they are being used. (Changed values are protected until this application reaches a commit point.) As soon as the program moves from one row to another, other programs may read or change the first row. For more information on ISOLATION, refer to the section on locking in Section 3 of *System and Database Administration Guide*.

LINECOUNT(integer)

Tells how many lines, including headings, are to be printed on each page of printed output. The default is **LINECOUNT(60)**.

LINK

Tells whether to invoke the linkage editor upon successful completion of compilation or assembly.

Use (YES) or (NO). The default is **LINK(YES)**.

LLIB(dsname)**L_nLIB(dsname)**

May name a data set that contains object or load modules to be included by the linkage editor. The parameters L_nLIB (where *n* may be 2, 3, or 4) are extensions of LLIB, used to simplify passing a list of data set names.

Use (NONE) to name no data set. The defaults are **LLIB(NONE)** and **L_nLIB(NONE)**.

LOAD(dsname)

Names a data set that is to contain the output from the linkage editor (that is, the load module). *dsname* may include a member name. The **default** is **LOAD(RUNLIB.LOAD)**.

LOPTION(string)

Gives a list of linkage editor options. For information on options you may use, Please refer to the appropriate MVS/XA or MVS/ESA publication.

Use (NONE) to give no options. The **default** is **LOPTION(NONE)**.

MACRO

Tells whether the macro preprocessor is to be invoked before the precompilation of a PL/I program.

Use (YES) or (NO). The **default** is **MACRO(YES)**.

OPTIONS

Tells whether the options actually used when executing the precompiler or the CICS command translator are to be printed with the output listing.

Use (YES) or (NO). The **default** is **OPTIONS(NO)**.

OUTNAME(string)

Gives a prefix used to form intermediate data set names. The **default** is **OUTNAME(TEMP)**.

string must not be enclosed between apostrophes and **must not have the same initial character as the INPUT dsname**. It may not contain special characters.

PARMS("string")

Gives a parameter string to be passed to the compiled program when it is executed.

Use (NONE) to pass no parameter string. The **default** is **PARMS(NONE)**.

PASS

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.

Use (**ONE**), (1), (**TWO**), or (2). The **default** is **PASS(ONE)** for PL/I and **PASS(TWO)** for Assembler. PASS has no effect for COBOL or FORTRAN: PASS(ONE) is forced.

PCLOAD(dsname)

Names the precompiler load module. *dsname* may include a member name. The **default** is **PCLOAD("DSN210.DSNLOAD(DSNHPC)")**.

PLAN(plan-name)

Names the application plan to be created by the BIND process. *plan-name* must not be DEFAULT.

The **default** is the first of these choices that is available:

- The member name of the INPUT data set
- The leftmost qualifier of the INPUT data set

PLIB(dsname)**PnLIB(dsname)**

Names the data set that contains host language source or SQL statements to be included by the SQL INCLUDE statement during precompilation. The parameters PnLIB (where *n* may be 2, 3, or 4) are extensions of PLIB, used to simplify passing a list of data set names.

Use (NONE) to name no data set. The **defaults** are **PLIB(NONE)** and **PnLIB(NONE)**.

PLILIB(dsname)

Names the linkage editor include library to be used for PL/I routines. The **default** is **PLILIB("SYS1.PLIBASE")**.

PLILOAD(dsname)

Names a data set that contains the PL/I optimizing compiler load module. *dsname* may include a member name. The **default** is **PLILOAD("SYS1.LINKLIB(IELOAA)")**.

PRECOMP

Tells whether to precompile.

Use (YES) or (NO). The **default** is **PRECOMP(YES)**.

PRINT

Tells were to send printed output, including the lists of options, source, cross-reference, error, and summary information.

(NONE)

Omits printed output. The **default** is **PRINT(NONE)**.

(TERM)

id = I\$82102.TERM option of DSNH CLIST

Sends output to the terminal.

(LEAVE)

Sends output to the specified print data set. You may allocate the print data set:

- Dynamically
- In the JCL used to run the DSNH CLIST (if in batch mode)
- With the TSO ALLOCATE command (before running DSNH).

(dsname)

Names a data set to be used for the output. The name may not contain special characters. Do not enclose *dsname* between apostrophes. The current user profile is prefixed to *dsname*. The following suffixes are also added:

Suffix...	Is used for...
LIST	PL/I macro listings (these listings are overwritten by the compiler listings)
PCLIST	Precompiler listings
CXLIST	CICS command translator listings
LIST	Compiler listings
LINKLIST	Link-edit listings

PSECSPAC(integer)

Tells the amount of secondary space to allocate for print data sets, in the units given by SPACEUN. The **default** is **PSECSPAC(20)**.

PSPACE(nn)

Tells the size of the print file in the units given by SPACEUN. The **default** is **PSPACE(20)**.

RCTERM(integer)

Gives the least value of the return code (from precompile, compile, linkedit, or bind) that prevents execution of later steps. The **default** is **RCTERM(8)**.

RELEASE

Tells the BIND process when to release DB2 resources. If you use ACQUIRE(USE), the **default** is **RELEASE(COMMIT)**; if you use ACQUIRE(ALLOCATE) the only allowed value (and the default) is RELEASE(DEALLOCATE).

(COMMIT)

Releases resources at each commit point.

(DEALLOCATE)

Releases resources when the plan terminates.

RETAIN

Tells whether BIND and EXECUTE privileges are to continue when an application plan is replaced. If they do not continue, only the invoker of the DSNH procedure will have BIND and EXECUTE privileges on the plan. (RETAIN has no effect with ACTION(ADD)).

Use (YES) or (NO). The **default** is **RETAIN(NO)**.

RUN

Tells whether to invoke the compiled program if the previous steps are successful, and, if so, in which environment it will execute. Your choice for the RUN parameter might affect your choice for LLIB.

(YES)**(TSO)**

Invokes the compiled program and includes the TSO attachment facility language interface with the load module. The **default** is **RUN(YES)**.

(NO)**(BATCH)**

Includes the TSO attachment facility language interface with the load module, but does not invoke the compiled program.

(CICS)

Includes the CICS attachment facility language interface with the load module, but does not invoke the compiled program. (CICS applications cannot run in TSO.)

(IMS)

Includes the IMS/VS attachment facility language interface with the load module, but does not invoke the compiled program. (IMS/VS applications cannot run in TSO.)

RUNIN

Tells where to obtain input for the RUN step.

(TERM)

Obtains input from the terminal. The **default** is **RUNIN(TERM)**.

(LEAVE)

Obtains input from SYSIN if the only steps taken are LINK and RUN.

(LEAVE) obtains input from FT05F001 if the language is FORTRAN. Do not use (LEAVE) for any other cases.

(dsname)

Names a data set to be used for the input.

(NONE)

Allocates no input file.

RUNOUT

Tells where to send output from the RUN step.

(TERM)

Sends output to the terminal. The **default** is **RUNOUT(TERM)**.

(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.

(dsname)

Names a data set to receive output.

(NONE)

Allocates no output file for the RUN step.

SOURCE

Tells whether the source code and diagnostics are to be printed with output from the precompiler, CICS command translator, and compiler. Use (YES) or (NO). The **default** is **SOURCE(NO)**.

SPACEUN

Specifies the unit of space for PSPACE and WSPACE.

(TRACK)

Makes the space unit one track. The **default** is **SPACEUN(TRACK)**.

(CYLINDER)

Makes the space unit one cylinder.

SQLDELIM

Sets the APOSTSQL or QUOTESQL precompiler option, to name the SQL string delimiter and, by implication, the SQL escape character. Whichever character is chosen to be the string delimiter, the other is used for the SQL escape character.

This parameter is effective *only for COBOL*. For PL/I, FORTRAN, and assembler language programs, the precompiler forces the APOSTSQL option.

(DEFAULT)

Designates the value chosen, during installation, for the SQL STRING DELIMITER field on the APPLICATION PROGRAMMING DEFAULTS panel.

(APOSTSQL)

Specifies that the string delimiter is the apostrophe (') and the escape character is the quotation mark (").

(QUOTESQL)

Specifies that the string delimiter is the quotation mark (") and the escape character is the apostrophe (').

SUFFIX

Tells whether the TSO standard naming convention should be followed. That convention adds a TSO userid 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). Names become, for example, *userid.name.COBOL*, *userid.name.PL1*, *userid.name.FORTRAN*, or *userid.name.ASM*.

Use (YES) or (NO). The **default** is **SUFFIX(YES)**.

SYSTEM(*subsystem-name*)

Gives the DB2 subsystem name as known to MVS. The **default** is the installation-defined subsystem name (often DSN).

TERM

Tells where to send terminal output, including error information, error statements, and summary information.

(TERM)

Sends output to the terminal. The **default** is **TERM(TERM)**.

(NONE)

Omits terminal output.

(LEAVE)

Sends the output to the current allocation for SYSTEM.

(dsname)

Names a data set to be used for terminal output. Do not enclose *dsname* between apostrophes. The following suffixes are added to *dsname*.

Suffix...	Is used for...
PCTERM	Precompiler output
LIST	Compiler output

TIME(*format*)

Specifies the format for TIME values that are output from DB2, overriding the format specified when DB2 is installed. *format* is the TIME output format, which may be ISO, EUR, USA, JIS, or LOCAL. **There is no default**, since this option overrides the default previously specified.

VALIDATE

Tells BIND when the full validity and authorization of SQL statements are to be verified:

(RUN)

Verifies statements when the program is executed and the application plan is used.

(BIND)

Verifies statements when bind processing occurs.

WORKUNIT(*unit*)

Tells what device to use for temporary and intermediate data sets. *unit* may be a unit name or a device type. The **default** is the UADS unit name for the current TSO user. The default in batch mode is any eligible device.

WSECSPAC(*integer*)

Tells the amount of secondary space to allocate for work data sets, in the units given by SPACEUN. The **default** is **WSECSPAC(20)**.

WSPACE(*integer*)

Tells the size of the temporary files in the units given by SPACEUN. The **default** is **WSPACE(20)**.

XLIB(*dsname*)

Names the linkage editor include library to be used for DB2 routines. The **default** is **XLIB("DSN210.DSNLOAD")**.

XREF

Tells whether a sorted cross-reference listing of symbolic names used in source statements is to be printed with output from the precompiler, compiler, and linkage editor.

Use (YES) or (NO). The default is **XREF(NO)**.

Usage Notes

There are several restrictions on the use of the DSNH CLIST:

SYSPROC: If compilation is done, the SYSPROC data set must include the DB2 CLIST library.

Link-edit restrictions: DSNH cannot process programs that need additional linkedit control statements, and cannot linkedit programs that use the call attachment facility.

You cannot use the NOLOAD and SYNTAX linkedit options.

Library limits: There can be at most 8 bind libraries, 4 precompile libraries, 4 compile libraries, and 4 linkedit libraries.

NONE is a reserved word: NONE may not be the name of an input or a load library, or the value of the string passed with PARMS.

WORKUNIT parameter: You should 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.

COBOL parameters: The BUF and SIZE parameters passed to the COBOL compiler may need to be changed.

COBOL options: The COBOL DYNAM option has several restrictions:

- You cannot use the DYNAM option with CICS.
- You must use the VS COBOL II library.
- To use it with TSO or batch, the DSNLOAD library must precede the IMS/VS RESLIB in the steplib, joblib, or link list concatenations.
- To use the option with IMS/VS, the IMS/VS RESLIB must precede DSNLOAD.

If you have a problem without an obvious cause, use the CONTROL parameter. Most problems involve the allocation of non-existent data sets, and CONTROL will help you trace this. See the *IBM DATABASE 2 Version 2 Application Programming Guide* information about the sequence of program preparation.

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.

NAME option: The NAME option is not supported.

Options requiring additional DD statements: Several options require DD statements that are not provided by the DSNH CLIST, as shown in Figure 9.

Option	Statements Required for...
CDECK	SYPUNCH
COUNT	SYSCOUNT, SYSDBG, SYSDBOUT, SYSUT5, a debug file
DECK	SYPUNCH
DUMP	SYSABEND, SYSDUMP, or SYSUDUMP
FDECK	SYPUNCH
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

Figure 9. COBOL Options that Required Additional DD Statements

FORTRAN and PL/I considerations: Variable format input records are not supported.

SQL host variables: SQL host variables must be explicitly defined.

Restrictions on the 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.

Examples

Example 1: Precompile, bind, compile, linkedit, and run the COBOL program in daa set DSN210.DSNSAMP(DSN8BC4).

- The compiler load module is in SYS1.VSCOLIB(IKFCBL00).
- Additional load modules to be included are in DSN210.RUNLIB.LOAD and DSN210.DSNSAMP.
- The load module will be put into the data set DSN210.RUNLIB.LOAD(DSN8BC4).
- The plan name will be DSN8BC21 for the bind and run.
- DCLGEN data from DSN210.SRCLIB.DATA is required for the precompile.

It is assumed that the DSNH CLIST is in your SYSPROC concatenation.

```
DSNH INPUT(''DSN210.DSNSAMP(DSN8BC4)'' ) -
COBLOAD(''SYS1.VSCOLIB(IKFCBL00)'' ) -
LLIB(''DSN210.RUNLIB.LOAD'' ) -
L2LIB(''DSN210.DSNSAMP'' ) -
LOAD(''DSN210.RUNLIB.LOAD'' ) -
PLAN(DSN8BC21) -
PLIB(''DSN210.SRCLIB.DATA'' )
```

Example 2: Precompile, bind, compile, and linkedit the program in data set DSN210.DSNSAMP.PLI(DSN8BP4).

- The program is written in PL/I; the macro pass is not needed.
- The PL/I compiler options MAP, XREF, and DMP will be used.

DSNH (TSO CLIST)

- Additional load modules to be included are in DSN210.RUNLIB.LOAD and DSN210.DSNSAMP.
- PL/I routines are in the linkage editor include library SYS2.LINKLIB(IEL0AA).
- The DB2 subsystem identifier is SSTR.
- The load module will be put into the data set DSN210.RUNLIB.LOAD(DSN8BC4).
- Printed output is sent to the following data sets:
 - Precompiler listings** DSN210.PROG.PCLIST
 - Compiler listings** DSN210.PROG.LIST
 - Link edit listings** DSN210.PROG.LINKLIST
- The plan name will be DSN8BC21 for the bind and run.
- The DCLGEN data from DSN210.SRCLIB.DATA is required for the precompile.

```
DSNH INPUT(''DSN210.DSNSAMP(DSN8BP4)'' ) -  
HOST(PLI) MACRO(NO) -  
COPTION ('MAP DUMP') -  
LLIB(''DSN210.RUNLIB.LOAD'' ) -  
L2LIB(''DSN210.DSNSAMP'' ) -  
PLILOAD(''SYS2.LINKLIB(IEL0AA)'' ) -  
SYSTEM(SSTR) -  
LOAD(''DSN210.RUNLIB.LOAD'' ) -  
PRINT(PROG) -  
PLAN(DSN8BC21) -  
PLIB(''DSN210.SRCLIB.DATA'' )
```

The COPTION parameters are enclosed between single apostrophes so that they will be passed by TSO as a single parameter. If a single token is being passed as a parameter, no apostrophes are needed. This 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 should not be performed, the data set name must be enclosed between sets of 3 apostrophes if the CLIST is invoked implicitly, and sets of 6 apostrophes if the CLIST is invoked explicitly.

The user prefix for this example was DSN210; 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: Precompile, bind, compile, and linkedit the program in data set DSN210.DSNSAMP(DSN8BD3).

- The C linkage editor include library is EDC.V1R1.LOAD.
- The C compiler load module is EDC.V1R1.LOAD(EDCCOMP).

```
ALLOC      DD(SYSPROC) DSN('DSN210.DSNCLIST') SHR
%DSNH BIND(YES) ACQUIRE(USE) ACTION(REPLACE)-
EXPLAIN(NO) -
CICSXLAT(NO)-
COMPILE(YES) -
CCULIB(''DSN210.DSNSAMP'')-
CCLLIB(''EDC.V1R1.LOAD'')-
CCLOAD(''EDC.V1R1.LOAD(EDCCOMP)'')-
DBRM(''DSN210.DBRMLIB.DATA(DSN8BD3)'')-
DECIMAL(PERIOD) DELIMIT(DEFAULT) FLAG(I)-
HOST(C) ISOLATION(RR)-
INPUT(''DSN210.DSNSAMP(DSN8BD3)'')-
LINECOUNT(60) LINK(YES)-
LLIB(''DSN210.RUNLIB.LOAD'')-
L2LIB(''DSN210.DSNLOAD'')-
LOAD(''DSN210.RUNLIB.LOAD'')-
MACRO(NO)-
OUTNAME(TEMP)-
PLAN(DSN8BD21) PRECOMP(YES)-
PLIB(''DSN210.DSNSAMP'')-
PRINT(TEMP) RCTERM(8)-
RELEASE(COMMIT) RETAIN(YES)-
RUN(NO) RUNIN(TERM)-
RUNOUT(TERM) SOURCE(YES)-
SYSTEM(DSN) SQLDELIM(DEFAULT)-
VALIDATE(RUN)
```

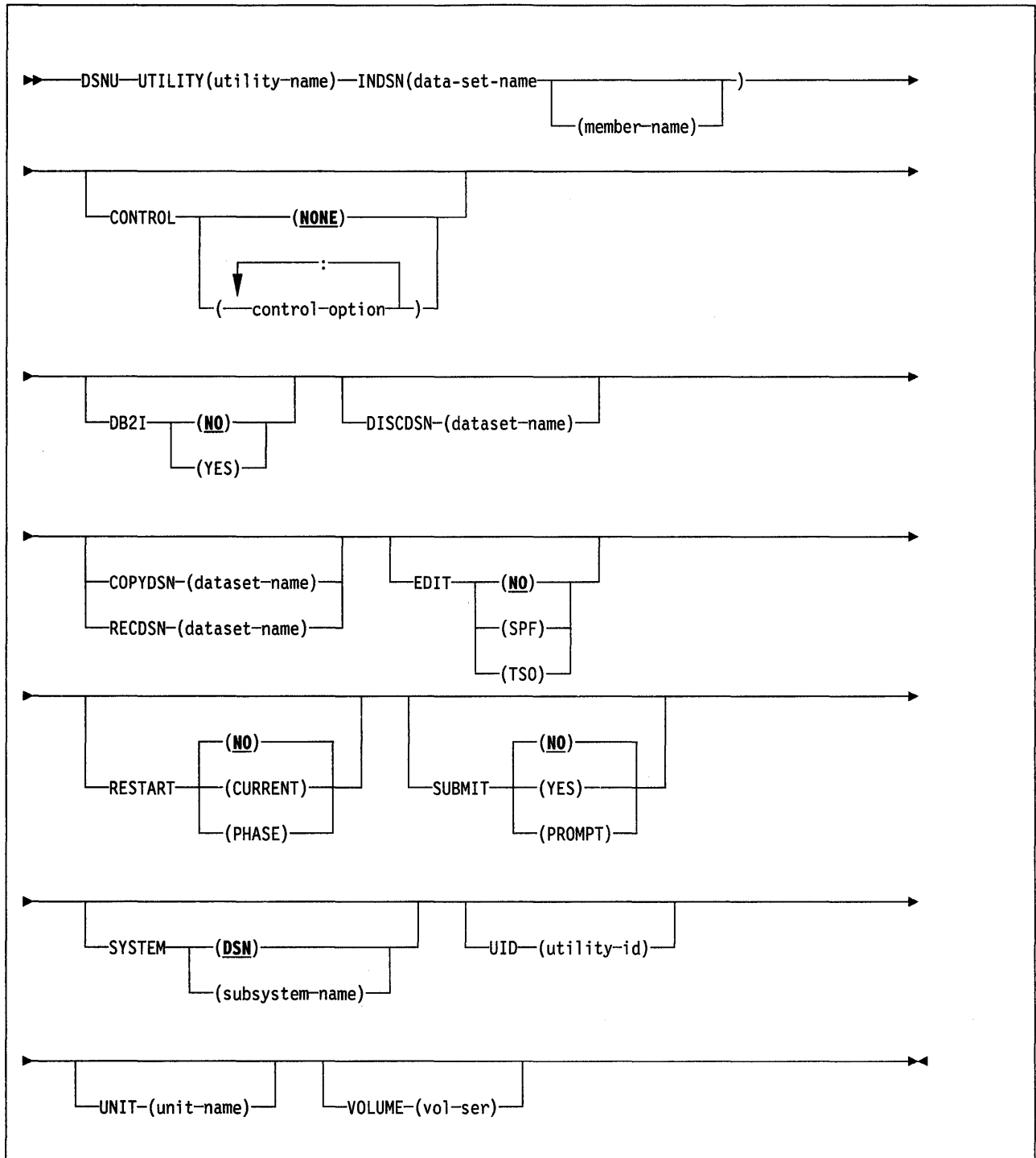
DSNU (TSO CLIST)

The DSNU command procedure (CLIST) generates the JCL needed to invoke the DSNUPROC procedure to execute DB2 utilities as batch jobs. (For a description of DSNUPROC, see “Chapter 3. Running DB2 Utilities” on page 151.)

To use the DSNU CLIST:

1. Create a file containing the required utility statements and control statements. The file is used to create the SYSIN data set in the generated job stream. Note that this file must not include double byte character set (DBCS) data.
2. Ensure that the DB2 CLIST library is allocated to ddname SYSPROC.
3. Invoke the command procedure, using the syntax that follows.

The DSNU CLIST can be invoked using the DB2I utilities panel.



Parameter Descriptions

Note: If you make syntax errors or omit parameter values, TSO will prompt you for correct parameter spelling and omitted values.

The parentheses shown in the following descriptions are required.

%

Identifies DSNU as a member of a command procedure library. Specifying this parameter is optional; however, it does improve performance.

UTILITY *utility-name*

Names the utility you want to invoke. Select the name from the following list.

CHECK	MERGECOPY	REORG	STOSPACE
COPY	MODIFY	REPAIR	
DIAGNOSE	QUIESCE	REPORT	
LOAD	RECOVER	RUNSTATS	

INDSN

Tells what data set contains the utility statements and control statements. Note that the data set you specify must not contain double byte character set (DBCS) data.

(dataset-name)

Is the name of the data set. **Default:** The command procedure will prompt you for the data set name.

(member-name)

Is the member name, and must be given if the data set is partitioned.

CONTROL

Tells whether to trace the CLIST execution.

(NONE)

Omits tracing. The **default** is **CONTROL (NONE)**.

(control-option)

Lists one or more of the options given below. Separate items in the list by colons (:). To abbreviate, code only the first letter of the option.

LIST Displays TSO commands after symbolic substitution and before command execution.

CONLIST Displays CLIST commands after symbolic substitution and before command execution.

SYMLIST Displays all executable statements (TSO commands and CLIST statements) before the scan for symbolic substitution.

NONE Generates a CONTROL statement with the options NOLIST, NOCONLIST, and NOSYMLIST.

DB2I

Is for internal use only. Do not use this parameter.

DISCDSN (*dataset-name*)

Is the cataloged data set name used by LOAD as a discard data set, to hold records not loaded.

COPYDSN (*dataset-name*)

Is the name of the cataloged data set required as a target (output) data set. If you do not supply this information, the CLIST will prompt you for it. It is required for COPY and MERGECOPY only.

RECDSN (*dataset-name*)

Is the cataloged data set name required by LOAD for input or by REORG TABLESPACE as temporary or target data set. If you do not supply this information, the CLIST will prompt you for it. It is required for LOAD and REORG TABLESPACE only.

EDIT

Tells whether to invoke an editor to edit temporary file generated by the CLIST.

(NO)

Does not invoke an editor. The **default** is **EDIT (NO)**.

(SPF)

Invokes the ISPF editor.

(TSO)

Invokes the TSO editor.

RESTART

Tells whether this job restarts a current utility job, and, if so, at what point it is to be restarted.

(NO)

Indicates the utility is a new job, not a restart. There should not be any other utility job step with the same utility identifier (UID).

The **default** is **RESTART (NO)**.

(CURRENT)

Restarts the utility at the last commit point.

(PHASE)

Restarts the utility at the beginning of the current stopped phase. The current stopped phase can be determined by the -DISPLAY UTILITY command.

SUBMIT

Tells whether to submit the generated JCL for processing.

(NO)

Does not submit the JCL for processing. The **default** is **SUBMIT (NO)**.

(YES)

Submits the JCL data set for background processing, using the TSO SUBMIT command.

(PROMPT)

Prompts you, after the data set is processed, to tell whether to submit the JCL data set for batch processing. You cannot use PROMPT when the CLIST itself is executed in the TSO batch environment.

SYSTEM

Names the DB2 subsystem.

(DSN)

Gives DSN as the name of the DB2 subsystem. The **default** is **SYSTEM (DSN)**.

(*subsystem-name*)

Is the one to four character name of the DB2 subsystem.

DSNU (TSO CLIST)

UID (*utility-id*)

Provides a unique identifier for this utility job within DB2.

The **default** is *tso-userid.control-file-name*, where *control-file-name* for each of the utilities is given by the following list:

Utility	control-file-name
CHECK	DSNUCHE
CHECK DATA	DSNUCHD
COPY	DSNUCOP
DIAGNOSE	DSNUDIA
LOAD	DSNULOA
MERGECOPY	DSNUMER
MODIFY	DSNUMOD
QUIESCE	DSNUQUI
RECOVER TABLESPACE	DSNURCT
RECOVER INDEX	DSNURCI
REORG INDEX	DSNURGI
REORG TABLESPACE	DSNURGT
REPAIR	DSNUREP
REPORT	DSNURPT
RUNSTATS	DSNURUN
STOSPACE	DSNUSTO

UNIT (unit-name)

Gives a unit address, a generic device type, or a user-assigned group name for a device on which a new temporary or permanent data set will reside. *unit-name* is placed after the UNIT clause of the generated DD statement.

The default is **UNIT (SYSDA)**.

VOLUME (vol-ser)

Gives the serial number of the volume on which a new temporary or permanent data set will reside. *vol-ser* is placed after the VOL=SER clause of the generated DD statement. If you omit VOLUME, the VOL=SER clause is omitted from the generated DD statement.

Output

DSNU builds a one-step job stream. The job step consists of an EXEC statement that invokes the DB2 utility processor and the required DD statements, including the SYSIN DD * job stream.

JOB Statement: The JOB statement placed at the beginning of the job stream depends on the value of the DB2I parameter. Use the default (NO) if provision has been made to build the JOB statement via exits supplied by your installation and invoked by the TSO SUBMIT command. Alternatively, you can use the EDIT parameter to insert the JOB statement into the generated JCL by means of a text editor.

EXEC Statement: The command procedure builds the EXEC statement, placing in the PARM= field the CLIST parameters that you supply.

DD Statements: The command procedure builds the necessary JCL DD statements. Those statements vary according to the utility being invoked.

- **SYSPRINT DD SYSOUT=A**

Utility messages are sent to SYSPRINT. The generated JCL defines SYSPRINT as SYSOUT=A. You can use the TSO OUTPUT command to control the disposition of the SYSPRINT data set; for example, you can have it sent to your terminal. For further information, see *TSO Extensions Command Language Reference*.

- **UTPRINT DD SYSOUT=A**

DSNU (TSO CLIST)

If either LOAD or REORG builds an index, it invokes DFSORT. Messages from that program are sent to UTPRINT. The generated JCL defines UTPRINT as SYSOUT=A.

- **SYSIN DD ***

To build the SYSIN DD * job stream, DSNU copies the data set named by the INDSN parameter. The INDSN data set is not changed, and you can reuse it when the DSNU procedure has completed.

Editing the Data Set

You may request (through the EDIT parameter on the command procedure) to edit the data set before you submit it. You can use the editor to add a JCL statement to the job stream, change JCL parameters (such as ddnames), or change utility control statements.

If you use a ddname that is not the default on some utility statement, you must change the ddname in the JCL generated by DSNU. For example, in the REORG TABLESPACE utility the default for UNLDDN is SYSREC, and DSNU builds a SYSREC DD statement for REORG TABLESPACE. If you use a different value for UNLDDN, you must edit the JCL data set and change "SYSREC" to the ddname you used.

At the end of the edit, you can either save changes to the data set (by issuing SAVE), or instruct the editor to ignore all changes.

The SUBMIT parameter tells whether the data set is to be submitted as a background job. The temporary data set holding the JCL is reused: If you want to submit more than one job that invokes the same utility, you must rename the JCL data sets and submit them separately.

Example

Example 1: In this example, the CLIST generates a data set called *authorization-id.DSNURGT.CNTL*, containing JCL to invoke the DSNUPROC procedure, which will invoke the REORG TABLESPACE utility. The file MYREOR.DATA will be merged into the JCL data set as SYSIN input. The file MYREOR.WORK is a temporary data set required by REORG TABLESPACE. The TSO editor will be invoked to allow editing of the JCL data set, *authorization-id.DSNURGT.CNTL*. The JCL data set will then be submitted as a batch job. It will not be modified by this CLIST until a new request is made to invoke the REORG TABLESPACE utility.

```
%DSNU UTILITY(REORG TABLESPACE) INDSN(MYREOR.DATA)
      RECDN(MYREOR.WORK) RESTART(NO)
      EDIT(TSO) SUBMIT(YES)
```

END (DSN)

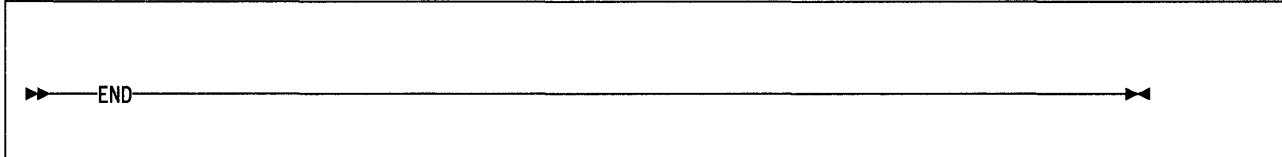
The DSN subcommand END is used to end the DSN session and return to TSO.

Environment

This subcommand originates from a TSO input stream when DSN is running in either background or foreground mode.

Authorization

None is required.



Keyword and Parameter Descriptions

None

Usage Note

In batch, if END is not found in the SYSIN stream, /* or // will end the DSN session. From the foreground, pressing the ATTENTION key twice will end the DSN session.

Example

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 (DSN)

The DSN subcommand FREE deletes application plans from DB2.

The FREE subcommand deletes corresponding table entries from the SYSIBM.SYSPLAN, SYSIBM.SYSDBRM, and SYSIBM.SYSSTMT tables. It deletes from the SYSIBM.SYSPLANDEP catalog table all the application plan dependencies on other structures. All authorization against an application plan name is dropped. The application plan name is then available for use in a BIND subcommand.

The FREE subcommand will not proceed until all currently executing applications using that plan finish executing.

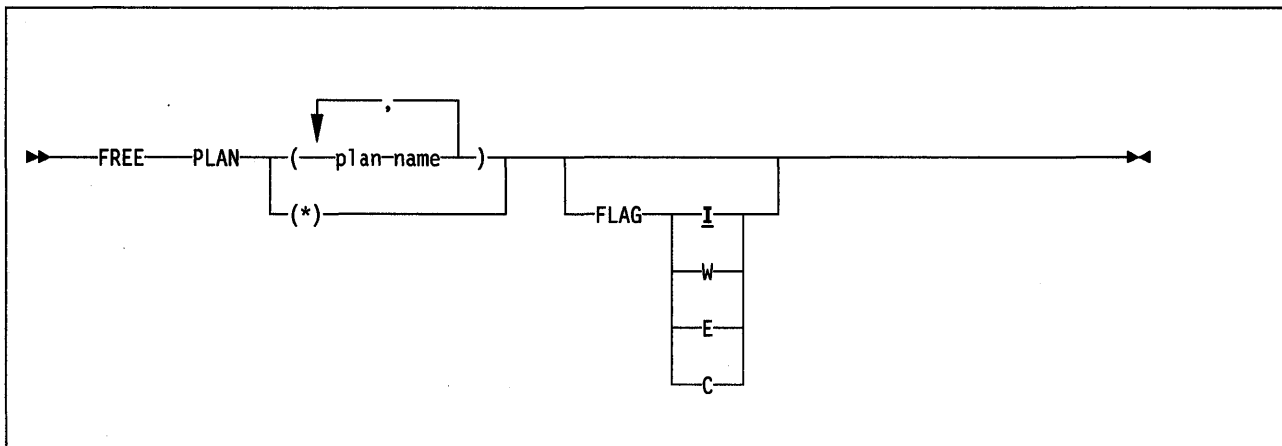
Environment

This subcommand is issued from DSN running in foreground or background, or it can be issued from the DB2I FREE panel. DSN can also be invoked from a CLIST running in either foreground or background mode.

Authorization

To issue the FREE subcommand, the privilege set defined below must include the BIND privilege on all identified plans, or include the SYSADM authority. The privilege set is the union of privileges designated by the authorization IDs of the process.

An authorization ID has the BIND privilege on a plan if the authorization ID is the owner of the plan or has been granted the BIND privilege on that plan.



Keyword and Keyword and Parameter Descriptions

PLAN

Deletes application plans from DB2.

(plan-name)

Lists the names of one or more names of plans you want to free.

(*)

Frees all application plans over which you have BIND authority. Care should be taken in using this form of the command.

FLAG

Tells 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. The **default** is **FLAG I**.
- W** Only warning, error, and completion messages.
- E** Only error and completion messages.
- C** Only completion messages.

Examples

Example: Free plan DSN8BC21. Generate only warning, error, and completion messages (not informational messages).

```
FREE PLAN (DSN8BC21)
FLAG (W)
```

MODIFY irlmproc,ABEND (MVS IRLM)

The MODIFY irlmproc,ABEND command terminates the IRLM abnormally whether or not any IMS/VS subsystems are identified to IRLM.

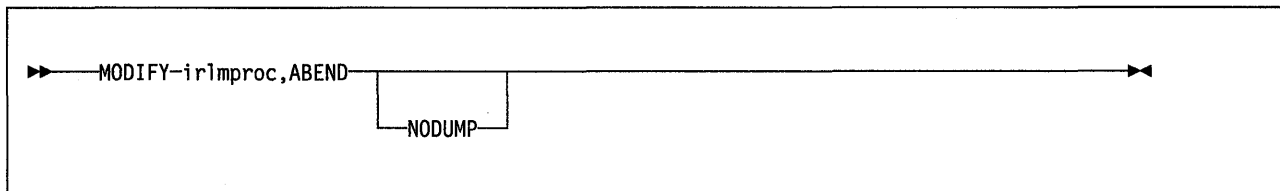
Abbreviation: F

Environment

This command can originate from an MVS console only.

Authorization

The command requires an appropriate level of MVS authority. Please refer to the appropriate MVS/XA or MVS/ESA publication.



Note: Parameters must be separated by commas with no spaces.

Keyword and Parameter Descriptions

irlmproc

Identifies the procedure name of the IRLM that is to be terminated.

NODUMP

Optional. Specifies that IRLM is to terminate without generating a dump. If the NODUMP parameter is not specified, the system dump is taken to the SYS1.DUMPxx data set.

Usage Notes

If the DXR011I irlmproc END-OF-TASK CLEAN-UP SUCCESSFUL message is not received because of outstanding IMS/VS requests in process, the MVS CANCEL command must be used to terminate the irlmproc. The MVS FORCE command should never be used to terminate the irlmproc, because it prevents the irlmproc from completing the cleanup required to permit the subsystem to be restarted.

Examples

Example 1: Enter on OS/VS SYSTEM system console:

```
F KRLM1,ABEND
```

Response on SYSTEM system console:

```
DXR020E KRLM ABENDED VIA MODIFY COMMAND
*IEA911E COMPLETE DUMP ON SYS1.DUMP00
FOR ASID(0004)
ERROR ID = SEQ00001 CPU00 ASID0004 TIME08.34.59.9
DXR011I KRLM END-OF-TASK CLEAN-UP SUCCESSFUL
IEF450I KRLM1 KRLM1 - ABEND S000 U2020
```

MODIFY irlmproc,START trace (MVS IRLM)

The MODIFY irlmproc,START trace command starts the IRLM internal, GTF, or PTB buffer traces.

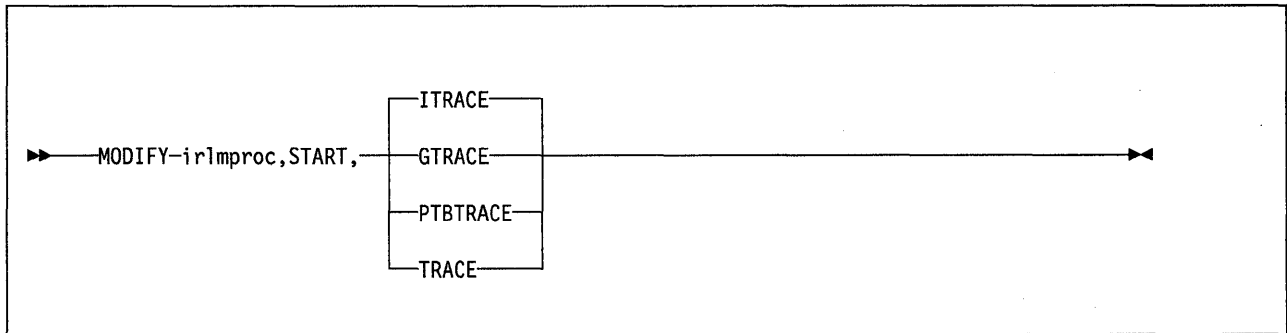
Abbreviation: F

Environment

This command can originate from an MVS console only.

Authorization

The command requires an appropriate level of MVS authority. Please refer to the appropriate MVS/XA or MVS/ESA publication.



Note: Parameters must be separated by commas with no spaces.

Keyword and Parameter Descriptions

irlmproc

Identifies the procedure name of the IRLM that is to be started.

trace

Specifies which of the IRLM traces is to be stopped.

ITRACE

Specifies that only an internal trace is to be started. The internal trace table is a 4K-byte wraparound table.

GTRACE

Specifies that tracing is to be started using the Generalized Trace Facility (GTF), an OS/VS service aid. GTF output is directed to the trace data set SYS1.TRACE and is formatted and printed using the PRDMP service.

GTF should be started with the START GTF command with the USR option before the IRLM trace is initiated. If GTF is not active when the IRLM GTRACE is started, an error message is issued and the records being traced are lost until GTF is started.

PTBTRACE

Specifies that the PTB buffer contents tracing using GTF is to be started. (The IRLM PTB trace is used only in the IMS/VS environment, not in the DB2 environment.)

TRACE

Specifies that internal, GTF, and PTB buffer tracing are to be started.

MODIFY...,START trace (MVS IRLM)

Examples

Example 1: Enter on OS/VS system console:

```
F KRLM1,START,ITRACE
```

Response on system console:

```
DXR004I KRLM ITRACE STARTED
```

Explanation: The operator on the system console has started IRLM internal tracing.

Example 2: Enter on OS/VS system console:

```
F KRLM1,START,GTRACE
```

Response on system console:

```
DXR004I KRLM GTRACE STARTED
```

Explanation: The operator on the system console has started IRLM tracing to GTF.

Example 3: Enter on OS/VS system console:

```
F JRLM2,START,TRACE
```

Response on system console:

```
DXR004I JRLM ITRACE STARTED  
DXR004I JRLM GTRACE STARTED  
DXR004I JRLM PTBTRACE STARTED
```

Explanation: The operator on the system console has STARTED IRLM internal, GTF, and PTB tracing.

MODIFY irlmproc,STATUS (MVS IRLM)

The MODIFY irlmproc,STATUS command displays the status of either this IRLM or the other IRLM.

Abbreviation: F

Environment

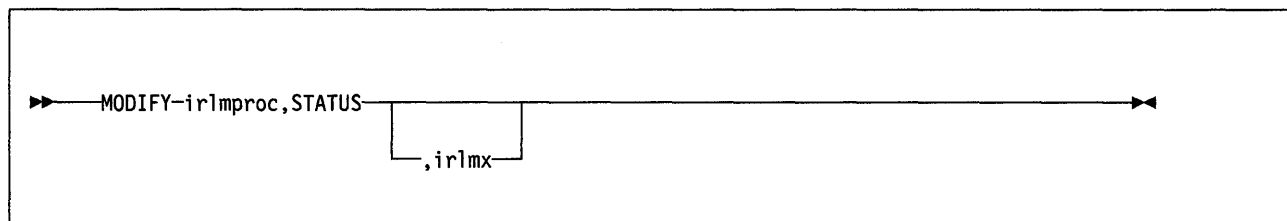
This command can originate from an MVS console only.

Authorization

The command requires an appropriate level of MVS authority. Please refer to the appropriate MVS/XA or MVS/ESA publication.

Keyword and Parameter Descriptions

Please refer to *IMS/VS Version 2 Operator's Reference Manual* for parameter descriptions.



Usage Notes

This command displays information for each subsystem connected to the specified IRLM designated using irlmproc. Each subsystem connected to the specified IRLM is listed, together with work unit and lock information.

If irlmx is not specified, message DXR010I is issued. Message DXR010I is also issued if the IRLM is started SCOPE=LOCAL or if irlmx is the VTAM name of the IRLM specified by the parameter irlmproc.

Message DXR010I is issued if irlmx is the VTAM name of the other IRLM. This message lists each subsystem connected to the IRLM specified by irlmx, with an indication as to whether the connection is active. Message DXR010I also indicates the global sharing state in effect.

Message DXR010I is issued if SESS is specified. This message lists session information.

No communication is involved in the processing of the STATUS command. Therefore, if the sharing state is 'COMM', irlmproc may not have the latest information relating to the subsystems connected to irlmx.

If irlmproc is started specifying SCOPE=GLOBAL, the second line of the display indicates the VTAM names and the IRLMIDs of both IRLMs. If an IRLMID is unknown, a '?' is inserted into the display. This situation occurs if the IRLMs are not now connected and have not previously been connected.

MODIFY...,STATUS (MVS IRLM)

See *IMS/VS Version 2 Messages and Codes Reference Manual* for more information about the messages described above.

Examples

Example 1: Enter on OS/VS SYSTEM 1 system console:

```
F KRLM1,STATUS
```

Response on SYSTEM 1 system console:

```
STC 35 DXR010I KRLM STATUS SCOPE=GLOBAL
STATUS OF SYSTEMS CONNECTED TO KRLM1 (1) (JRLM2 (2)) PT01
NAME STATUS UNITS HELD WAITING
IMS1 UP 0004 0003 0001
IMS5 DOWN ... 0001 ...
```

Explanation: The operator on system 1 has requested information about the IMS/VS systems connected to KRLM1. The second line of the display indicates the VTAM names and the IRLMIDs of both IRLMs. For this and subsequent examples, the VTAM name of the IRLM is the same as the IRLM procedure name.

Example 2: Enter on SYSTEM 1 system console:

```
F KRLM1,STATUS,JRLM2
```

Response on SYSTEM 1 system console:

```
STC 23 DXR010I KRLM STATUS NORMAL
STATUS OF SYSTEMS CONNECTED TO JRLM2 (2) (KRLM1 (2)) PT01
NAME STATUS NAME STATUS NAME STATUS
IMS2 UP IMS3 UP IMS4 DOWN
```

Explanation: The operator on system 1 has requested information from KRLM1 about the subsystems connected to the partner IRLM, JRLM2. Note that JRLM2 is the VTAM name of the partner IRLM, which, for these examples, is the same as the IRLM procedure name. The first line of the display indicates the sharing state in effect.

Example 3: Enter on SYSTEM 1 system console:

```
F KRLM1,STATUS
```

Response on SYSTEM 1 system console:

```
STC 35 DXR010I KRLM STATUS SCOPE=GLOBAL
STATUS OF SYSTEMS CONNECTED TO KRLM1 (1) (JRLM2 (?)) PT01
NAME STATUS UNITS HELD WAITING
IMS1 UP 0004 0003 0001
IMS5 DOWN ... 0001 ...
```

Explanation: The operator on system 1 has requested information about the subsystems connected to KRLM1. This example was created by starting KRLM1, but not starting JRLM2. Thus, KRLM1 does not yet know the IRLMID of JRLM2, and a ? appears in the second line of the display.

MODIFY irImproc,STOP trace (MVS IRLM)

The MODIFY irImproc,STOP trace command stops the IRLM internal, GTF, or PTB traces.

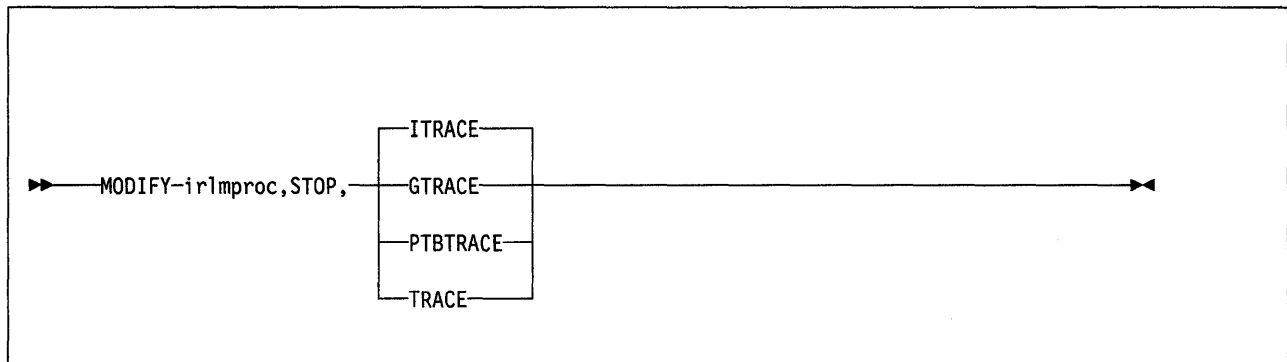
Abbreviation: F

Environment

This command can originate from an MVS console only.

Authorization

The command requires an appropriate level of MVS authority. Please refer to the appropriate MVS/XA or MVS/ESA publication.



Note: Parameters must be separated by commas with no spaces.

Keyword and Parameter Descriptions

irImproc

Identifies the procedure name of the IRLM that is to be terminated.

trace

Specifies which of the IRLM traces is to be stopped.

ITRACE

Specifies that only an internal trace is to be terminated.

GTRACE

Specifies that tracing using the Generalized Trace Facility (GTF) is to be terminated.

PTBTRACE

Specifies that the PTB buffer contents tracing using GTF is to be terminated. (The IRLM PTB trace is used only in the IMS/VS environment, not in the DB2 environment.)

TRACE

Specifies that internal, GTF, and PTB buffer tracing are to be terminated.

MODIFY...,STOP trace (MVS IRLM)

Examples

Example 1: Enter on OS/VS system console:

```
F KRLM1,STOP,ITRACE
```

Response on system console:

```
DXR004I KRLM ITRACE STOPPED
```

Explanation: The operator on the system console has terminated internal IRLM tracing.

Example 2: Enter on OS/VS system console:

```
F KRLM1,STOP,GTRACE
```

Response on system console:

```
DXR004I KRLM GTRACE STOPPED
```

Explanation: The operator on the system console has terminated GTF tracing.

Example 3: Enter on OS/VS system console:

```
F JRLM2,STOP,TRACE
```

Response on system console:

```
DXR004I JRLM ITRACE STOPPED  
DXR004I JRLM GTRACE STOPPED  
DXR004I JRLM PTBTRACE STOPPED
```

Explanation: The operator on the system console has terminated IRLM internal, GTF, and PTB tracing.

-MODIFY TRACE (DB2)

The DB2 command -MODIFY TRACE changes the trace events (IFCIDs) being traced for a particular active trace. -MODIFY TRACE also stops any IFCID previously active for the specified trace.

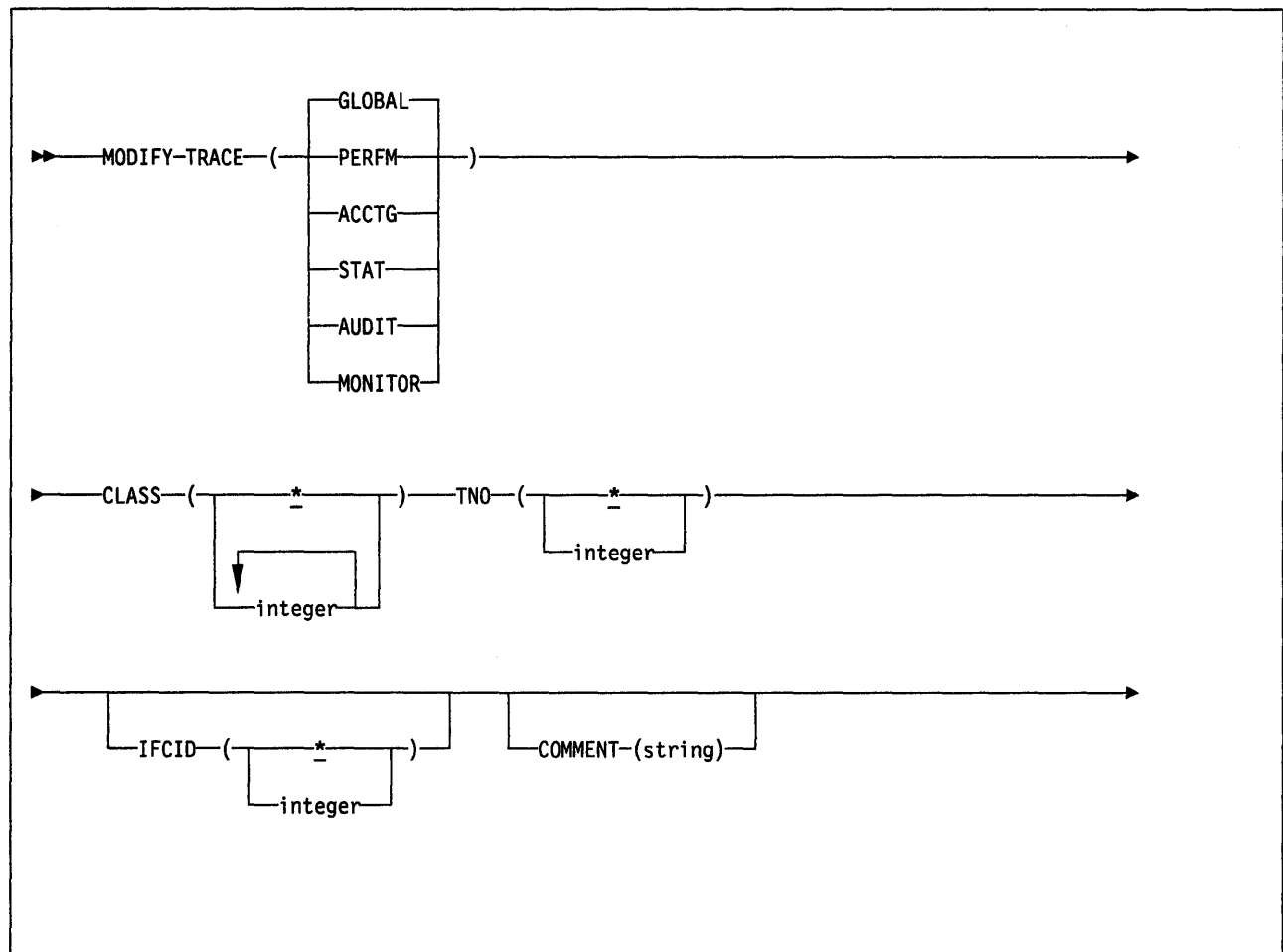
Abbreviation: -MOD TRACE

Environment

This command can be entered from an MVS console, from the DSN command processor, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

To issue this command, the privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, must include the SYSADM or SYSOPR authorities, or the TRACE privilege.



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see “DB2 Command Parsing” on page 11.

The trace type you specify determines which IFCIDs are started. For fuller descriptions of each trace type, see “-START TRACE (DB2)” on page 129.

Type	Description
GLOBAL	Service data from the entire DB2 subsystem Abbreviation: G
PERFM	Performance records of specific events Abbreviation: P
ACCTG	Accounting records for each transaction Abbreviation: A
STAT	Statistical data Abbreviation: S
AUDIT	Audit data Abbreviation: AU
MONITOR	Monitor data Abbreviation: MON

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.

CLASS (*integer*)

Limits the list to IFCIDs started for specified classes. For descriptions of the allowable classes, see “-START TRACE (DB2)” on page 129.

Abbreviation: C

The **default** is **CLASS(*)**, which starts all default IFCID classes.

TNO

Limits the list to a particular trace, identified by its trace number (1 to 32, 01 to 09). You may only specify one trace number.

The **default** is **TNO(*)**, which does not limit the list.

IFCID (*ifcid_nbr*)

Specifies the optional IFCID(s) (trace events) to start. All IFCIDs and classes specified are started for the trace type specified. If you do not specify this parameter, then all IFCIDs contained in the activated trace classes are started. The maximum number of IFCIDs is 64.

The **default** is **IFCID(*)**.

Examples

Example 1: Change trace number 6 so that it collects only statistics and accounting data. You may define CLASS(30) at your site.

```
-MODIFY TRACE(S) IFCID(1,2},3) TNO(6) CLASS(30)  
COMMENT ('STATS AND ACCOUNTING ON')
```

REBIND (DSN)

The REBIND subcommand rebinds an application plan when changes have been made that affect the plan, but the SQL statements in the program have not changed: For example, when authorizations have changed, when a new index has been created that the plan may use, or when RUNSTATS has been used.

REBIND is generally faster and more economical of resources than BIND. But if the SQL statements have changed, or if the program has been recompiled, use BIND (REPLACE).

Environment

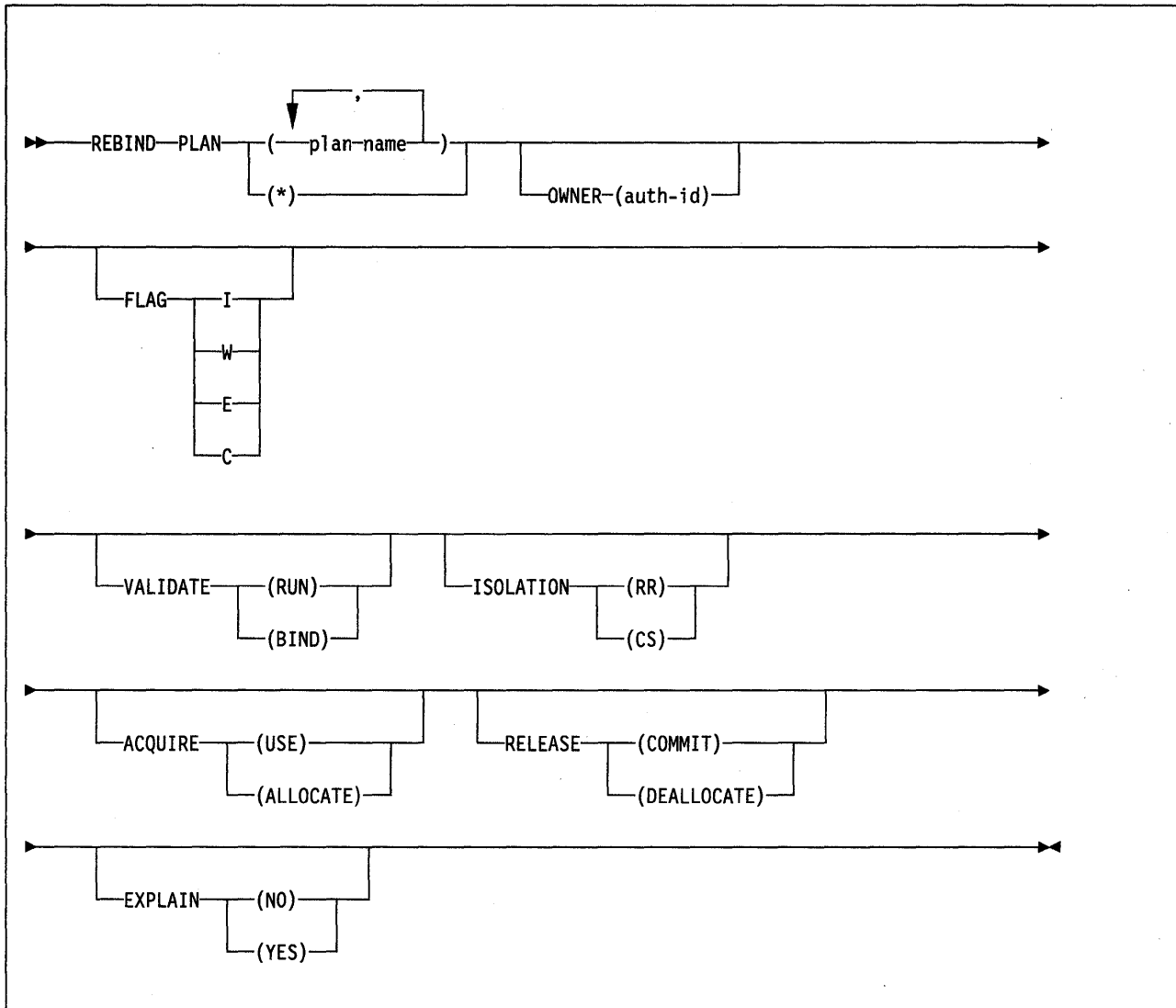
REBIND can be invoked using DB2I, or the REBIND subcommand can be issued through the DSN command processor running in foreground or background.

Authorization

To issue the REBIND subcommand, the privilege set defined below must include the BIND privilege or the SYSADM authority. If the owner of the new plan differs from the owner of the existing plan, the privilege set is the privileges designated by the authorization ID of the new owner of the plan. Otherwise, the privilege set is the union of privileges designated by the primary authorization ID of the process.

An authorization ID has the BIND privilege on a plan if the authorization ID is the owner of the plan or has been granted the BIND privilege on the plan.

REBIND (DSN)



Keyword and Parameter Descriptions

PLAN

Lists the application plans you want to rebind.

(plan-name)

Is the name of an application plan. *plan-name* may have 1 to 8 alphanumeric characters; the first character must be alphabetic.

(*)

Rebinds all application plans for which you have the BIND privilege.

The following parameters are optional.

OWNER (authorization-id)

Designates the authorization ID of the owner of the rebound plan.

If any of the authorization IDs of the process has the SYSADM authority,

authorization-id may be any value. Otherwise, *authorization-id* must be one of the authorization IDs of the process.

Default: If the OWNER keyword is omitted, the authorization ID of the existing plan remains the owner of the plan.

FLAG

Tells 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. The **default** is **FLAG I**.
- W** Only warning, error, and completion messages.
- E** Only error and completion messages.
- C** Only completion messages.

VALIDATE

Tells whether full validity checking occurs during execution of the REBIND subcommand, or whether some checking is deferred until the application plan is actually used. There are two validity checks that may be deferred—checking for table existence and for appropriate access authority.

The **default** is the value used the last time the plan was bound or rebound.

(RUN)

Defers full checking until the plan is used. Thus, any SQL statement that fails validation during the bind process is validated each time it is executed. Only warning messages are produced while the REBIND subcommand executes.

(BIND)

Checks validity during bind processing. Error messages are produced.

ISOLATION

Tells how far an application bound to this plan should be isolated from the effects of other executing applications.

The **default** is the value used the last time the plan was bound.

(RR)

Repeatable Read. Tells that database values read or changed by this application cannot be changed by other programs until this application commits or terminates. For more information about ISOLATION, refer to the section on locking in Section 3 of *System and Database Administration Guide*.

(CS)

Cursor Stability. Tells that database values read by this application are protected only while they are being used. (Changed values are protected until this application reaches a commit point.) As soon as the program moves from one row to another, other programs may read or change the first row. For more information about ISOLATION, refer to the section on locking in Section 3 of *System and Database Administration Guide*.

ACQUIRE

Tells whether resources are acquired when they are first accessed, or when the plan is allocated. The **default** is the value used the last time the plan was bound.

USE

Opens table spaces and acquires locks only when the application program bound to the plan first uses them.

ALLOCATE

Opens all table spaces and acquires all table space locks when the plan is allocated. The value has no effect on dynamic SQL statements; ACQUIRE(USE) is used for those.

RELEASE

Tells whether resources are released at each COMMIT point, or when the application terminates. The **default** is the value used the last time the plan was bound.

COMMIT

Releases resources at each commit point.

But, if you use ACQUIRE (ALLOCATE), you must use RELEASE (DEALLOCATE).

DEALLOCATE

Releases resources only when the application plan terminates. The value has no effect on dynamic SQL statements; RELEASE(COMMIT) is used for those.

EXPLAIN

Obtains information about how SQL statements in the DBRM will be executed, and inserts that information into the *x.PLAN_TABLE* table where *x* is the authorization ID of the owner of the plan. This table must exist before the binding process begins. For a description of the table, see "EXPLAIN (SQL)" in *SQL Reference*.

The value of the REBIND option EXPLAIN can be overridden by an SQL EXPLAIN statement in the program. Otherwise, the value of EXPLAIN in BIND applies to all explainable SQL statements in the program, and to the fullselect portion of any DECLARE CURSOR statements. In all inserts to *plan_owner_authid.PLAN_TABLE*, the value of QUERYNO is the statement number in the program.

The **default** is the value used the last time the plan was bound or rebound.

NO

Provides no EXPLAIN information.

YES

Inserts information in *userid.PLAN_TABLE*, which must have been previously created by or for the user performing the REBIND.

Output

REBIND updates or inserts entries in the following catalog tables:

- SYSIBM.SYSPLAN, which records the parameters used in binding application plans
- SYSIBM.SYSPLANAUTH, which records the privileges held over plans
- SYSIBM.SYSPLANDEP, which records the dependencies of plans on other objects

Usage Notes

Because of changes to DB2 descriptors on which the plan depends, rebinding is required when an application plan is invalidated (changes such as deletion of an index or revocation of authority). But, even when rebinding is not required, it can result in improved program performance (if, for example, an index has been created and a new application plan may make use of the index).

Rebinding of an application plan is not possible if the application is currently executing.

Example

Example: Rebind plan DSN8BC21 to enable DB2 to take advantage of a newly created index. W means that only warning, error, and completion messages will be issued. (No informational messages will be issued.) VALIDATE(BIND) indicates that any error condition should be raised during the bind process. CS indicates that database values being read by this application are protected only while they are being used. (Changed values are protected until the application reaches a commit point.) Since the OWNER keyword is omitted, the authorization ID that owns the plan remains the same.

```
REBIND PLAN (DSN8BC21)
  FLAG (W)
  VALIDATE (BIND)
  ISOLATION (CS)
```

-RECOVER BSDS (DB2)

The DB2 command `-RECOVER BSDS` reestablishes dual bootstrap data sets (BSDS) after one has been disabled by a data set error. Command processing consists of allocating a data set with the same name as the one that encountered the error and copying the contents of the remaining good data set into the new data set.

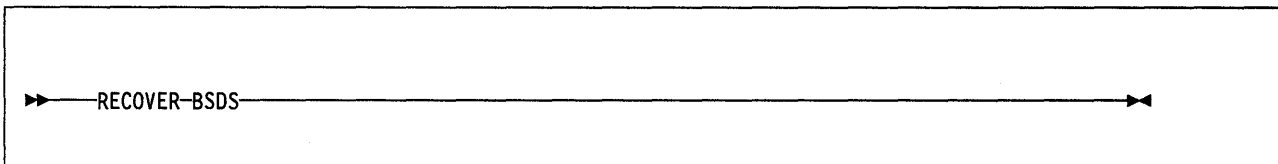
Abbreviation: `-REC BSDS`

Environment

This command can be entered from an MVS console, from the DSN processor, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

To issue this command the privilege set, consisting of the union of all privileges designated by all authorization IDs of the process, must include the `SYSADM` authority or the `BSDS` privilege.



Usage Notes

For a detailed description of steps the installation must take to reestablish dual BSDS mode after a BSDS I/O error occurs, see Section 5 of *System and Database Administration Guide*.

Example

Example: Reestablish dual BSDS mode.

`-RECOVER BSDS`

-RECOVER INDOUBT (DB2)

The DB2 -RECOVER INDOUBT command recovers threads left indoubt because DB2 or a transaction manager could not automatically recover them.

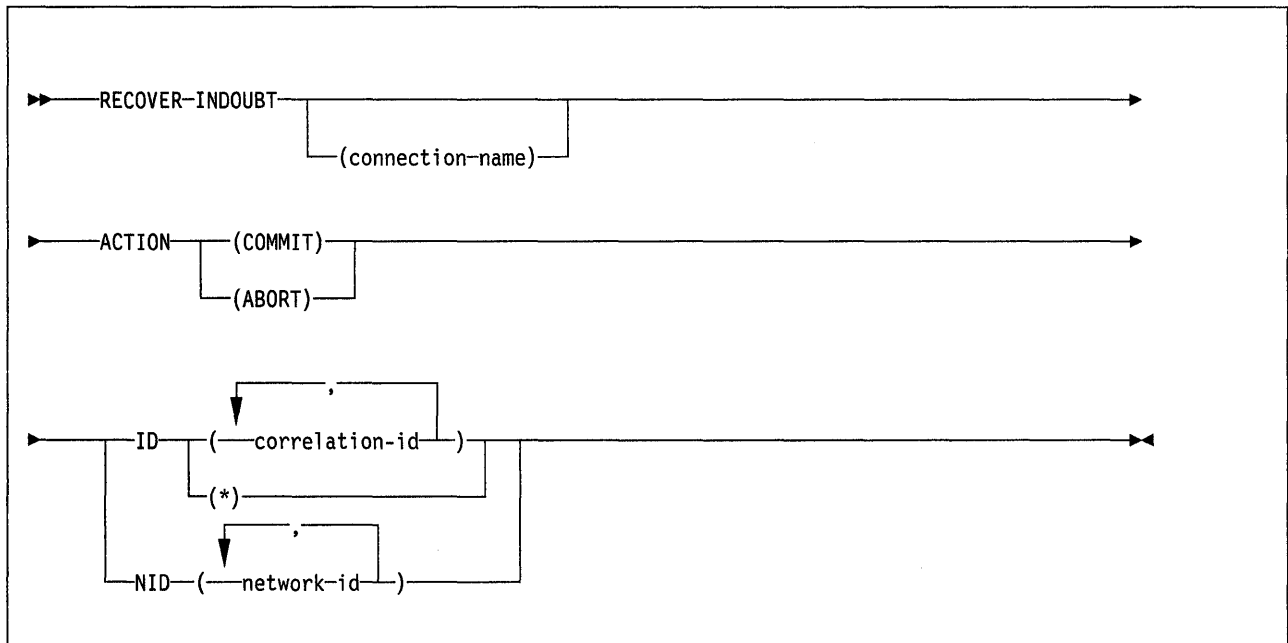
Abbreviation: -REC IND

Environment

This command can be entered from an MVS console, from the DSN command processor, from the DB2 commands panel, or from an IMS/VS or CICS terminal.

Authorization

To issue this command the privilege set, consisting of the union of all privileges designated by all authorization IDs of the process, must include the SYSADM or SYSOPR authority, or the RECOVER privilege. or have been granted the RECOVER privilege.



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see “DB2 Command Parsing” on page 11.

(connection-name)

Is a 1- to 8-character connection name. Threads belonging to that connection name are recovered.

The **default** is the connection name from which you enter the command. If you enter this command from an MVS console, you *must* supply a connection name; no default is available.

ACTION

Tells whether to commit or abort the indoubt thread.

Abbreviation: ACT

-RECOVER INDOUBT (DB2)

(COMMIT)

Commits the thread.

(ABORT)

Aborts the thread.

Restriction: You must use either ID or NID, but not both.

ID

Tells whether to recover a specific thread or all threads.

(*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 keyword.

(*)

Recovers all threads associated with the connection-name. Even threads with more than one associated network-id are resolved.

NID (*network-id*)

Identifies threads by their network-ids. *network-id* is a network-id associated with an individual thread. You may use more than one *network-id* for the same *connection-name*.

A network id is a 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.

The network id appears on the recovery log of the originating node as an identification of the unit of work. In DB2 internal format it uses 16 bytes (an 8-byte node name immediately followed by an 8-byte number).

Usage Notes

network-id is not normally needed, because the *correlation-id* can identify indoubt threads. However, if *correlation-id* is not unique, *network-id* must be used.

Example

Example: Recover indoubt threads. Schedule a commit for all threads associated with the connection name from which the command is entered.

```
-RECOVER INDOUBT  
ACTION (COMMIT)
```

RUN (DSN)

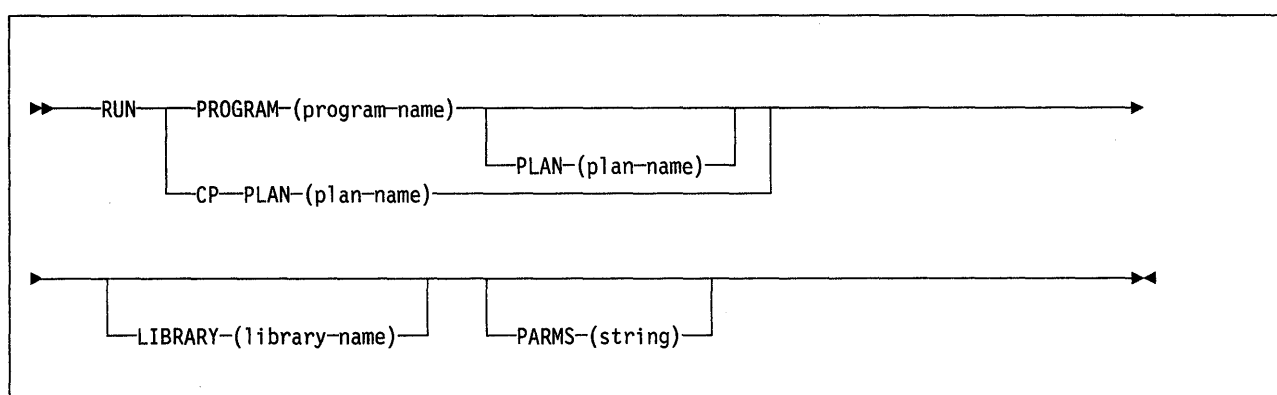
The DSN subcommand RUN invokes an application program, which may contain SQL statements.

Environment

This subcommand is issued under the DSN command processor running in either foreground or background mode, or it can be issued using the DB2I RUN panel.

Authorization

The privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, must include the EXECUTE privilege on the plan. The privilege may have been explicitly granted or may be inherent in another privilege. The EXECUTE privilege on a plan is inherent in SYSADM authority and in ownership of the plan.



Keyword and Parameter Descriptions

Use at least one of the two following clauses, but do not use the same clause twice.

CP

Causes a prompt to be issued: "ENTER TSO COMMAND". This is useful for running command processors and debugging programs (for example, COBTEST).

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*.

PROGRAM (*program-name*)

program-name is the name of the program you want to run.

LIBRARY (*library-name*)

library-name is the name of the data set containing the program to be run. The default is **SYS1.LINKLIB**.

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, repre-

RUN (DSN)

sent each of them by two consecutive apostrophes. The list is passed as a varying-length character string of up to 100 characters.

For PL/I: Use a list of the form 'A/B', where A lists parameters for the PL/I run-time (transient) package, and B, parameters for the application program. If A is not needed, write the list in the form '/B'. 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 COBOL: Use a list of the form 'D/C', where C lists parameters for the COBOL run-time package, and D, parameters for the application program. If D is not needed, write the list in the form '/C'.

For Assembler Language: Use a list of the form 'program parameters'. There are no run-time parameters.

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 'DSN210.RUNLIB.LOAD'.

```
DSN SYSTEM (DSN)
RUN PROGRAM (DSN8BC4) LIB ('DSN210.RUNLIB.LOAD')
```

Example 2: Run application program DSN8BP4. The application plan is DSN8BE21. The program is in library 'DSN210.RUNLIB.LOAD'. Pass the parameter O'TOOLE to the application program, which is written in PL/I. Pass no parameters to the run-time package.

```
DSN SYSTEM (DSN)
RUN PROGRAM (DSN8BP4) PLAN (DSN8BE21) -
LIB ('DSN210.RUNLIB.LOAD') PARS ('/O'TOOLE')
```

SPUFI (DSN)

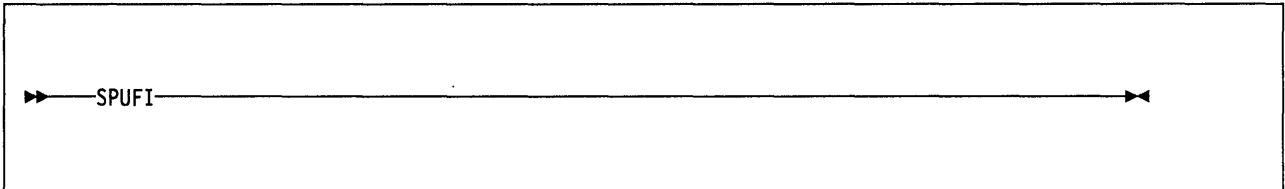
The DSN subcommand SPUFI invokes the SQL Processor Using File Input.

Environment

You can use this subcommand only under ISPF. You can issue it from ISPF option 6, or from a CLIST.

Authorization

None is required.



Usage Notes

- The effect of the SPUFI subcommand is to invoke 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 the *Application Programming 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.
- The SPUFI panel variables you enter after invoking SPUFI directly with the DSN command will not be saved in the same ISPF variable pool used by DB2I. Panel variables will therefore vary depending on whether you invoke the facility directly, or through DB2I.

/SSR (IMS/VS)

The IMS/VS command /SSR allows the IMS/VS operator to enter an external subsystem command.

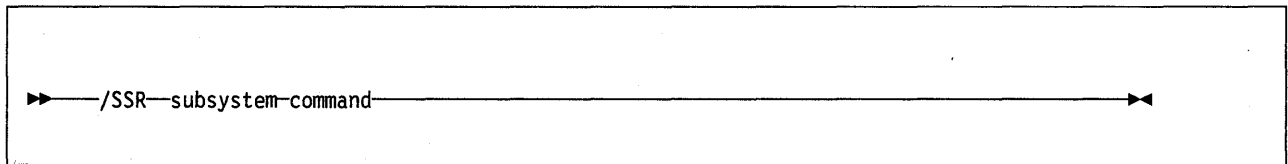
Environment

This command originates from an IMS/VS terminal.

Authorization

This command requires an appropriate level of IMS/VS authority, as described in the *IMS/VS Version 2 System Administration Guide*.

In addition, the privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, 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 command's description.



Keyword and Parameter Descriptions

subsystem-command

For *subsystem-command*, substitute 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).¹

Usage Notes

IMS/VS uses the command recognition character (CRC) to determine which external subsystem (that is, DB2) will receive the command. The only action taken by IMS/VS is to route the command to the appropriate subsystem.

¹ The subsystem recognition character is defined in the IMS/VS SSM member for the external subsystem.

/START (IMS/VS)

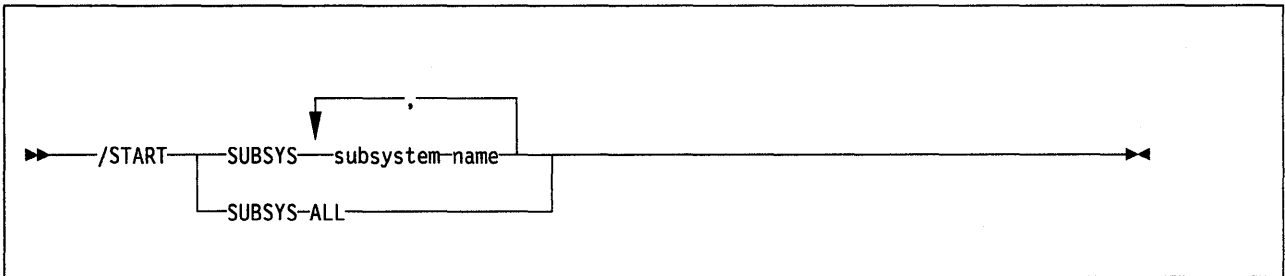
The IMS/VS command /START (with the SUBSYS parameter) makes available the connection between IMS/VS and the specified external subsystem. Establishing the connection allows application programs to access resources managed by the external subsystem.

Environment

This command originates from an IMS/VS terminal.

Authorization

This command requires an appropriate level of IMS/VS authority, as described in the *IMS/VS Version 2 System Administration Guide*.



Keyword and Parameter Descriptions

SUBSYS

Specifies one or more names of external subsystems to be connected to IMS/VS, or all external subsystems.

subsystem-name

For *subsystem-name*, substitute one or more names of external subsystems to be connected to IMS/VS.

ALL

Indicates that all external subsystems are to be connected to IMS/VS.

Usage Notes

The copy in main storage of the external subsystem PROCLIB entry will be 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/VS.

The preceding is only a partial description of the /START command. For a complete description, refer to *IMS/VS Operator's Reference Manual*.

-START DATABASE (DB2)

The `-START DATABASE` command is typically used after a previous `-STOP DATABASE` command, or after a table space or index has been placed in deferred restart status by DB2. It makes the named 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.

If you use the `SPACENAM` keyword, table spaces and indexes you list are made available for application programs.

Abbreviation: `-STA DB`

Environment

This command can be entered from an MVS console, from the DSN command processor, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

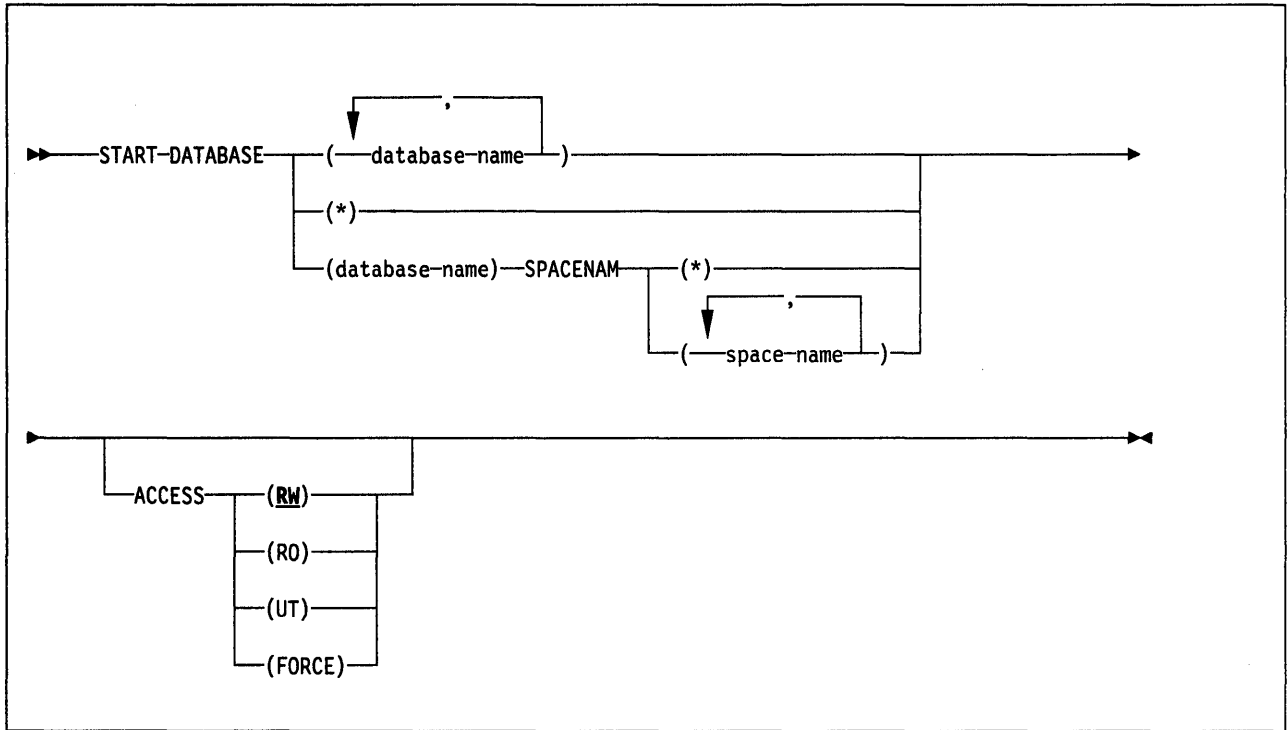
The privilege to start all databases requires no privilege. All databases for which the privilege set defined below has the `STARTDB` privilege are started.

The privilege to start a list of databases requires no privilege. All databases for which the privilege set has the `STARTDB` privilege are started. Error messages are produced for those specified databases for which the privilege set does not have the `STARTDB` privilege.

The `STARTDB` privilege may have been explicitly granted, or may be inherent in another privilege; it is inherent in the following authorities:

- `SYSADM` authority
- `DBMAINT` authority for the database.

To execute this command when `DSNDB06` or `DSNDB07` is stopped, the privilege set must include `Install SYSADM` authority.



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see "DB2 Command Parsing" on page 11.

(database-name)

Is the name of a database 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 has at least DBMAINT authority (except databases already started). You cannot use "*" with ACCESS(FORCE).

SPACENAM

Tells what particular table spaces or indexes within the data base are to be started. You cannot use this keyword if you have used more than one data-base name. If you use ACCESS(FORCE), you must use SPACENAM with a list of table space and index names.

Abbreviation: SPACE

space-name

Is the name of a table space or index space to be started.

(*)

Starts all table spaces and index spaces in the named database. You cannot use "*" with ACCESS(FORCE).

You may use a list of several names of table spaces and index spaces. Separate names in the list by commas. All table spaces and index spaces must be in the single named database.

ACCESS

Tells whether databases can be read from and written to, read from only, or accessed by utilities only.

-START DATABASE (DB2)

Abbreviation: ACC

(RW)

Allows programs to read from or write to the named databases. The default is **ACCESS(RW)**.

(RO)

Allows programs only to read the named databases. Any programs attempting to change data will not succeed.

(UT)

Allows only DB2 utilities to access the named databases.

(FORCE)

Resets any indications that a table space or index is unavailable because of pending deferred restarts, write error ranges, read-only accesses, or utility controls. Also resets the check pending, copy pending, and recovery pending states. Full access to the data is forced.

With ACCESS(FORCE) you must use a single database name, the SPACENAM keyword, and an explicit list of table space and index names. You may not use DATABASE * or SPACENAM *.

CAUTION:

A table space or index space started with ACCESS(FORCE) may be in an inconsistent state.

See "Usage Notes" for further instructions.

Usage Notes

Data sets off line: It is not necessary for every disk pack containing partitions, table spaces, or indexes to be on line when a database is started. Packs must, however, be on line when partitions, table spaces, or indexes are first referred to. If they are not, an error in opening will occur.

Table spaces and indexes explicitly stopped: If table spaces and indexes are stopped explicitly (using the -STOP DATABASE command with the SPACENAM parameter), they must be started explicitly. Starting the database will not start table spaces or indexes that have been explicitly stopped.

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 non-DB2 program, 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 "DSN1COPY (Service Aid)" on page 200.

Normal Completion Message: The message DSN9022I will be issued to indicate that processing is complete. At this point, any recovery that was in process will be done. To ensure that no work is pending, issue the -DISPLAY DATABASE command.

Examples

Example 1: Start table space DSN8S21E in database DSN8D21A.

```
-START DATABASE (DSN8D21A) SPACENAM (DSN8S21E)
```

Example 2: Start all databases for which you have authority.

```
-START DATABASE (*)
```

-START DB2 (DB2)

The DB2 -START DB2 command 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/VS, CICS, and TSO).

The effect of restarting the system can be controlled by a “conditional restart control record,” which you create by the CHANGE LOG INVENTORY utility. For a description of the effects, see “CHANGE LOG INVENTORY (DSNJU003) (Utility)” on page 170 and “Usage Notes” on page 124.

Abbreviation: -STA DB2

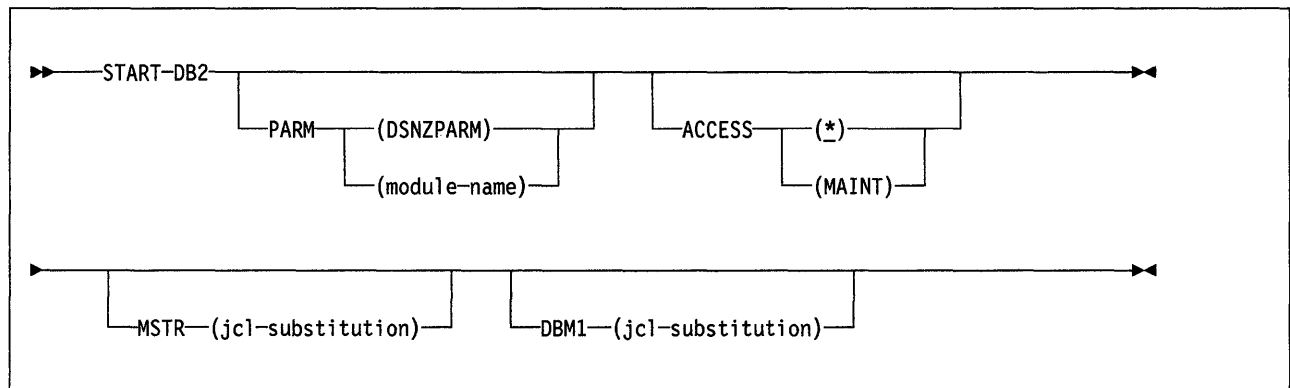
Environment

This command can only be entered from an MVS console. The name of the DB2 subsystem is determined by the subsystem recognition character. For example, -START indicates the DB2 subsystem to be started is the one with '-' as the subsystem recognition character.

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.

Authorization

None is required. However, the command can be executed only from an MVS console with the START command capability. Please refer to the appropriate MVS/XA or MVS/ESA publication.



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see “DB2 Command Parsing” on page 11.

The following parameters are optional.

PARM (*member-name*)

Names the load module that contains the DB2 subsystem initialization parameters. *member-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 may connect to DB2. The **default** is **ACCESS(*)**.

(MAINT)

Prohibits access to the any authorization IDs other than Install SYSADM and Install SYSOPR.

To issue the this command with the MAINT option, the privilege set must include Install SYSADM or Install SYSOPR.

MSTR *jcl-substitution*

Gives parameters and values to be substituted in the EXEC statement of the JCL that invokes the startup procedure for the system services address space.

jcl-substitution is 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.

We recommend that you omit the keyword, and use the parameters provided in the startup procedure.

DBM1 *jcl-substitution*

Gives parameters and values to be substituted in the EXEC statement of the JCL that invokes the startup procedure for the database services address space.

jcl-substitution is 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.

We recommend that you omit the keyword, and use the parameters provided in the startup procedure.

Usage Notes

Subsystem recognition character: If your installation has more than one DB2 subsystem, you must define more than one subsystem recognition character.

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:

1. Log records are processed to the extent determined by the conditional restart control record.
2. 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 recovery elements (RUDT).
3. 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.

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 invokes the startup procedure for the system services address space uses the symbol &RGN, provide a value for that symbol.

```
-START DB2 MSTR('RGN=6000K')
```

START irlmproc (MVS IRLM)

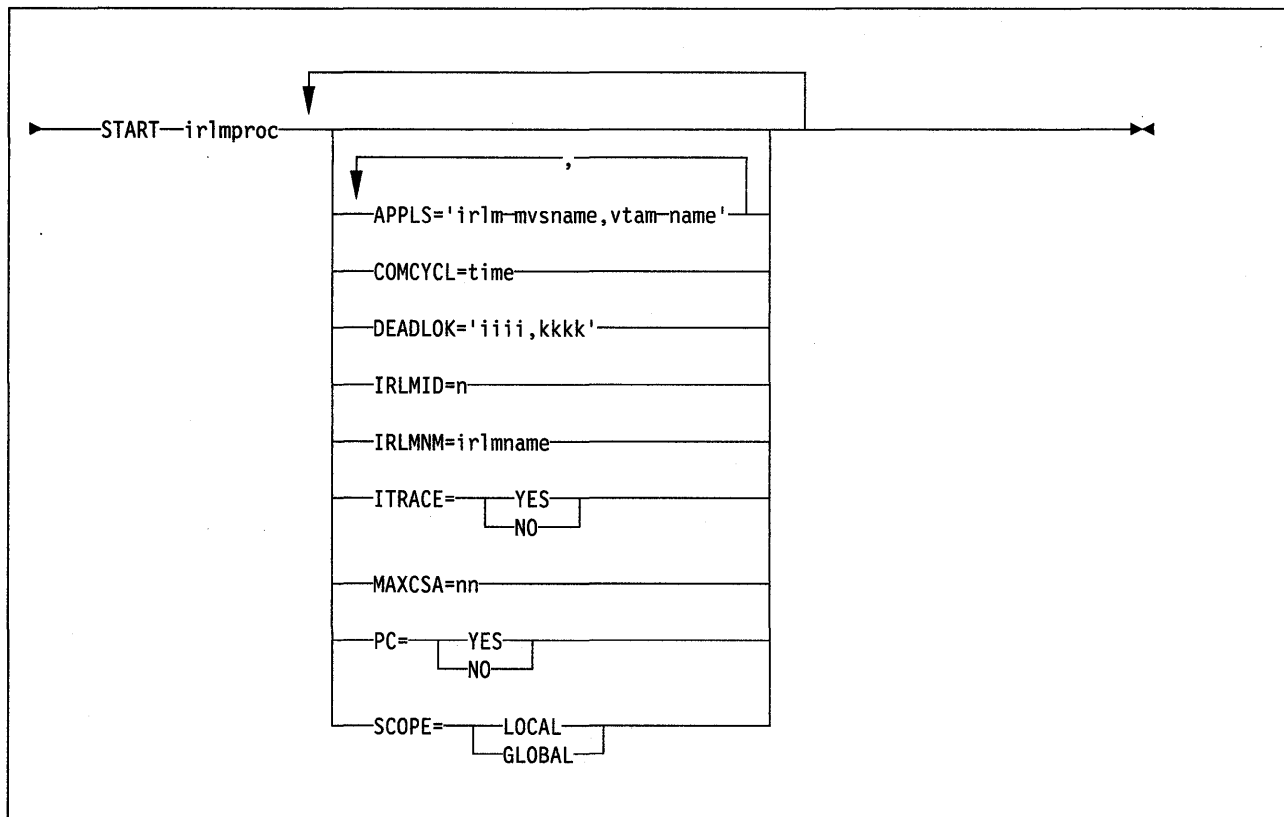
The START irlmproc command starts an IRLM component with a procedure put in place by the installation. Symbolic parameters in the procedure may be overridden on the START irlmproc command.

Environment

This command can originate from an MVS console only.

Authorization

The command requires an appropriate level of MVS authority. Please refer to the appropriate MVS/XA or MVS/ESA publication.



Note: Parameters must be separated by commas with no spaces.

Keyword and Parameter Descriptions

irlmproc

Is the procedure name of the IRLM to be started.

The following parameters are optional.

DEADLOK = 'iiii,kkkk'

Gives 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 1- to 4-digit number from 1 to 9999. It tells the interval (in seconds) between IRLM local deadlock-detection cycles.

kkkk

Is a 1- to 4-digit number from 1 to 9999. It tells the number of local deadlock cycles that must expire before global deadlock-detection is performed.

IRLMID = *n*

Gives a decimal number from 1 to 8 that is used to distinguish between multiple IRLMs. *n* must be a single digit.

IRLMNM = *irlmname*

Gives the 1- to 4-byte MVS subsystem name assigned to this IRLM.

ITRACE =

Tells whether the IRLM is to automatically start the request handler internal trace.

YES Starts the trace.**NO** Does not start the trace.**MAXCSA = *nn***

Specifies the maximum amount of CSA the IRLM can use for its dynamic control block structure. *nn* must be a 1 to 2 digit decimal number from 1 to 99. The number indicates what multiple of 100K bytes of CSA storage IRLM will use. If IRLM is using Extended CSA support and running on a MVS/XA system then the number indicates multiples of 1 meg of Extended CSA storage. For example,

MAXCSA=3

allows the IRLM to use 300K bytes of CSA or 3 MEG bytes of ECSA if executing with Extended CSA support on a MVS/XA system.

PC =

Tells whether the IRLM is to use the MVS cross-memory services.

YES Uses the cross-memory services.**NO** Does not use the cross-memory services.**SCOPE =**

Tells whether intersystem sharing is to be performed.

LOCAL

Limits the sharing to intrasystem; VTAM is neither required nor used.

Example

Example: Enter on OS/VS system console:

```
S KRLM1,DEADLOK='100,200'
```

-START RLIMIT (DB2)

The -START RLIMIT command starts the resource limit facility (governor) and specifies a resource limit specification table for the facility to use.

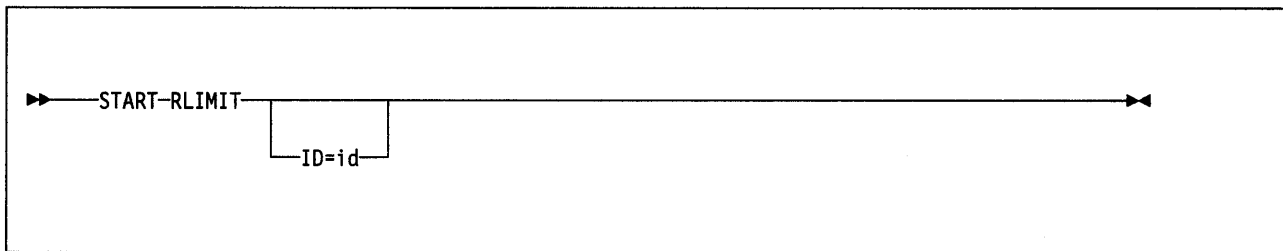
Abbreviation: START RLIM

Environment

This command can be entered from an MVS console, from the DSN command processor, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

To issue this command, the privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, must include the SYSADM or SYSOPR authorities, or the TRACE privilege.



Keyword and Parameter Descriptions

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 2 of *System and Database Administration Guide* for more information about creating a resource limit specification table.

The **default** ID is the one specified in the RLFTBL parameter when DB2 was installed.

Example

Example:

```
START RLIMIT ID=01
```

-START TRACE (DB2)

The DB2 command -START TRACE starts DB2 traces. For more information about the trace facility, see *Diagnosis Guide and Reference* and Section 6 of *System and Database Administration Guide*.

Abbreviation: -STA TRACE

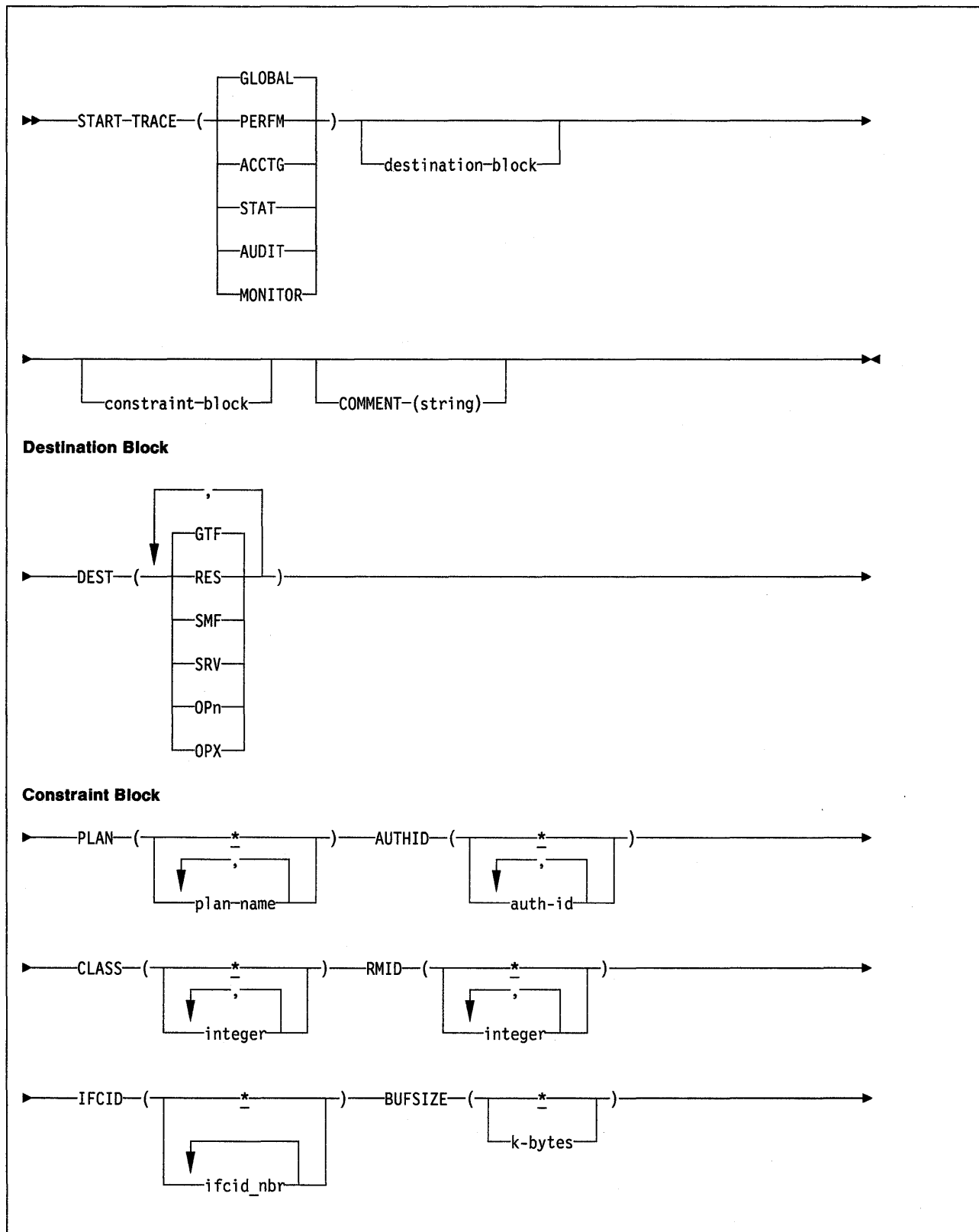
Environment

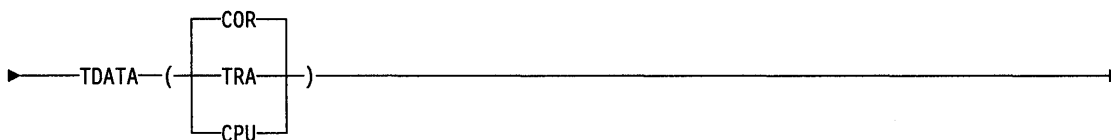
This command can be entered from an MVS console, from the DSN command processor, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

To execute this command, the privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, must include the SYSADM or SYSOPR authority, or the TRACE privilege.

-START TRACE (DB2)



Constraint Block (continued)**Keyword and Parameter Descriptions**

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see “DB2 Command Parsing” on page 11.

You must specify a trace type.

The parameters GLOBAL, PERFM, ACCTG, STAT, AUDIT, and MONITOR identify the *type* of trace started.

(GLOBAL)

Is intended for servicing DB2 and includes data from the entire DB2 subsystem.

Abbreviation: G

(PERFM)

Is intended for performance analysis and tuning, and includes records of specific events in the system.

Abbreviation: P

(ACCTG)

Includes records written for each thread, intended to be used in accounting for a particular program or authorization ID.

Abbreviation: A

(STAT)

Collects statistical data broadcast by various components of DB2, at time intervals that can be chosen during installation.

Abbreviation: S

(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 may 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.

-START TRACE (DB2)

The Destination Block

DEST

Tells where the trace output is to be recorded. You may use more than one value, but do not use the same value twice. If you do not specify a value, the list is not limited.

Abbreviation: D

The allowable values, and the default value, depend on the type of trace started, as shown in the following table:

Type	GTF	RES	SMF	SRV	OPn	OPX
GLOBAL	Allowed	Default	Allowed	Allowed	Allowed	Allowed
PERFM	Allowed	NO	Default	Allowed	Allowed	Allowed
ACCTG	Allowed	NO	Default	Allowed	Allowed	Allowed
STAT	Allowed	NO	Default	Allowed	Allowed	Allowed
AUDIT	Allowed	NO	Default	Allowed	Allowed	Allowed
MONITOR	Allowed	NO	Allowed	Allowed	Allowed	Default

Figure 10. Allowable Destinations for Each Trace Type

The meaning of each value is as follows:

Value Trace output is recorded by ...

GTF The MVS Generalized Trace Facility. If used, the GTF must be started and accepting user (USR) records before the -START TRACE command is given. The record identifier for records from DB2 is X'0FB9'.

RES A wrap-around table residing in CSA storage.

SMF The System Management Facility. If used, the SMF must be functioning before the -START TRACE command is given. The types of records written depend on the trace type, as shown in the following list:

Trace type	Record type
GLOBAL	102
PERFM	102
ACCTG	101
STAT	100
AUDIT	102
MONITOR	102

SRV A serviceability routine reserved for use by IBM only; not for general use.

OPn A specific destination. *n* can be an integer from 1 to 8.

OPX A generic destination which uses the first free OPn slot.

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:

Type	PLAN	AUTHID	CLASS	RMID
GLOBAL	Allowed	Allowed	Allowed	Allowed
PERFM	Allowed	Allowed	Allowed	Allowed
ACCTG	Allowed	Allowed	Allowed	NO
STAT	NO	NO	Allowed	NO
AUDIT	Allowed	Allowed	Allowed	NO
MONITOR	Allowed	Allowed	Allowed	Allowed

Figure 11. Allowable Constraints for Each Trace Type

The meaning of each option is as follows:

PLAN

Introduces a list of specific plans for which trace information is gathered. You cannot use this option for a STAT or AUDIT trace.

(*)

Starts a trace for all plans. The **default** is **PLAN (*)**.

plan-name

Is the name of an application plan. You may use up to 8 names; a separate trace is started for each name. If you use more than one name, you may use only one value for AUTHID.

AUTHID

Introduces a list of specific authorization IDs for which trace information is gathered. You cannot use this option for a STAT or AUDIT trace.

(*)

Starts a trace for all authorization IDs. The **default** is **AUTHID (*)**.

authorization-id

Names an authorization ID. You may use up to 8 identifiers; a separate trace is started for each identifier. If you use more than one identifier, you may use only one value for PLAN.

CLASS

Introduces a list of classes of data gathered. What classes are allowable, and their meaning, depends on the type of trace started.

Abbreviation: C

(*)

Starts a trace for all classes of data.

integer

Is any number in the list that follows. You may use any number of classes that are allowed for the type of trace started. The **default** classes for each trace type are marked in the list by asterisks (*).

-START TRACE (DB2)

Type	Class	Data Collected
GLOBAL	1 *	Reserved for major functions and exceptions
	2 *	Reserved for subcomponents
	3 *	All other module entries and exits
	4	Index logging
	5	Work file information and SQL parsing
	6	Installation-defined serviceability record (IFCID 0156)
	7 - 29	Reserved
	30 - 32	Available for installation-defined classes
	PERFM	1 *
2 *		Subsystem-related events
3 *		SQL-related events
4		Buffer Manager I/O and EDM pool requests
5		Log Manager
6		Summary lock information
7		Detailed lock information
8		Data manager detail
9		Sort detail
10		BIND, commands, and utilities
11		Dispatching
12		Storage manager
13		Edit and validation exits
14		In and out of DB2
15		Installation-defined performance record (IFCID 0154)
16 - 29	Reserved	
30 - 32	Available for installation-defined classes	
ACCTG	1 *	Accounting data
	2	In DB2 time
	3	Wait time
	4	Installation-defined accounting record (IFCID 0151)
	5 - 29	Reserved
30 - 32	Available for installation-defined classes	
STAT	1 *	Statistical data
	2	Installation-defined statistics record (IFCID 0152)
	3 - 29	Reserved
	30 - 32	Available for installation-defined classes
AUDIT	1 *	Authorization failures
	2	Explicit GRANT and REVOKE
	3	CREATE, DROP, and ALTER operations against audited tables
	4	First change of audited object
	5	First read of audited object
	6	SQL statement at bind
	7	Change in authorization for audited object
	8	Utility access to any object
	9	Installation-defined audit record (IFCID 0146)
	10 - 29	Reserved
30 - 32	Available for installation-defined classes	
MONITOR	1 *	Activate the READS IFCIDs
	2	Time in DB2 (CPU and elapsed)
	3	Wait time in DB2 for I/O, locks, and latches; also, resource usage information
	4	Installation-defined monitor record (IFCID 0155)

Figure 12 (Part 1 of 2). Allowable Classes for Each Trace Type

Type	Class	Data Collected
	5 - 29	Reserved
	30 - 32	Available for installation-defined classes

Figure 12 (Part 2 of 2). Allowable Classes for Each Trace Type

A more detailed description of the contents of each class is given in Section 6 of *System and Database Administration Guide*.

RMID

Introduces a list of specific resource managers for which trace information is gathered. You cannot use this option for a STAT, ACCTG, or AUDIT trace.

(*)

Starts a trace for all resource managers. The **default** is **RMID (*)**.

integer

Is the identifying number of any resource manager in the table that follows. You may use up to 8 allowable resource manager identifiers; do not use the same one twice.

RMID	Resource manager
1	Initialization procedures
2	Agent services management
3	Recovery management
4	Recovery log management
6	Storage management
7	Subsystem support for allied memories
8	Subsystem support for SSI functions
10	Buffer management
12	System parameter management
14	Data manager
15	Index manager
16	Instrumentation commands, trace, and dump services
18	Data space management (in the database address space)
19	Data space management (VSAM Access Method Services in the system services address space)
20	Service controller
21	Data base utilities
22	Relational database support
23	General command processing
24	Message generator
26	Instrumentation accounting and statistics

Figure 13. Resource Manager Identifiers

IFCID (*ifcid_nbr*)

Specifies the optional IFCID(s) (trace events) to start. All specified IFCIDs and their associated classes are started. If you do not specify this parameter, then all IFCIDs contained in the activated trace classes are started. The maximum number of IFCIDs is 64.

The **default** is **IFCID(*)**.

BUFSIZE (*k_bytes*)

Specifies the size of an IFC managed buffer that will receive the trace data. You can only specify this parameter if you specified an *OPn* destination.

-START TRACE (DB2)

(*k_bytes*) can range from 8 to 128K in 4K increments. If you specify a value outside of this range, then the range limit closest to the specified value is used.

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.

CORRELATION

Places a correlation header on the record.

Abbreviation: COR

TRACE

Places a trace header of the record.

Abbreviation: TRA

CPU

Places a CPU header on the record. The CPU header contains the current CPU time for the MVS TCB or SRB executing.

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.

Usage Notes

Number of traces: If you use one value, or no values, for PLANID and AUTHID, the -START TRACE command starts a single trace. If you use multiple values for either PLANID or AUTHID, the command starts a trace for each plan or authorization ID. There may be up to 32 traces going at one time.

Examples

Example 1: Start the broadest possible trace for everything. Write records to the Generalized Trace Facility. Include a comment to identify the trace.

```
-START TRACE (GLOBAL)
  DEST (GTF)
  COMMENT ('GLOBAL TRACE FOR SYSTEM TEST')
```

Example 2: Start an accounting trace for plan DSN8BC21. Write records to SMF (that will happen by default). Include a comment to identify the trace.

```
-START TRACE (ACCTG)
  PLAN (DSN8BC21)
  COMMENT ('ACCTG TRACE FOR DSN8BC21')
```

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/VS)

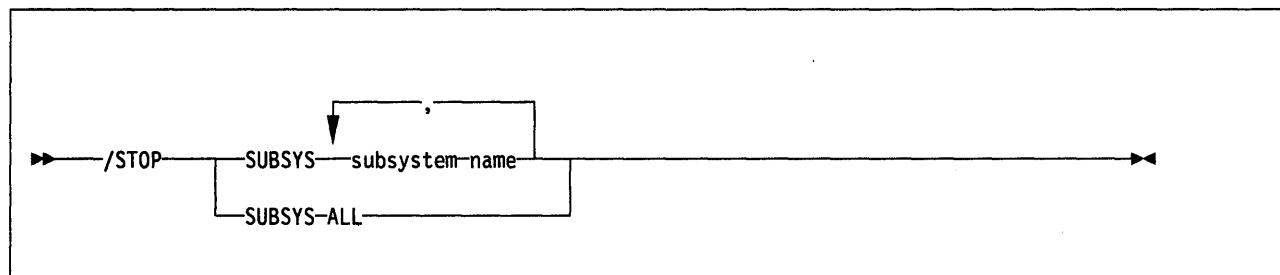
The IMS/VS command /STOP (with the SUBSYS parameter) prevents application programs from accessing external subsystem resources.

Environment

This command originates from an IMS/VS terminal.

Authorization

This command requires an appropriate level of IMS/VS authority, as described in the *IMS/VS Version 2 System Administration Guide*.



Keyword and Parameter Descriptions

SUBSYS

Specifies whether connection is to be stopped for one or more names of external subsystems presently connected to IMS/VS, or for all of them.

subsystem-name

For *subsystem-name*, substitute one or more names of external subsystems whose connection to IMS/VS is to be stopped.

ALL

Indicates connection is to be stopped for all external subsystems presently connected to IMS/VS.

Usage Notes

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 then 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 will then refresh the copy in main storage of the PROCLIB entry with the modified entry.

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

-STOP DATABASE (DB2)

The -STOP DATABASE command makes the specified databases unavailable for applications, and closes their data sets.

Abbreviation: -STO DB

Environment

This command can be entered from an MVS console, from the DSN processor under TSO, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

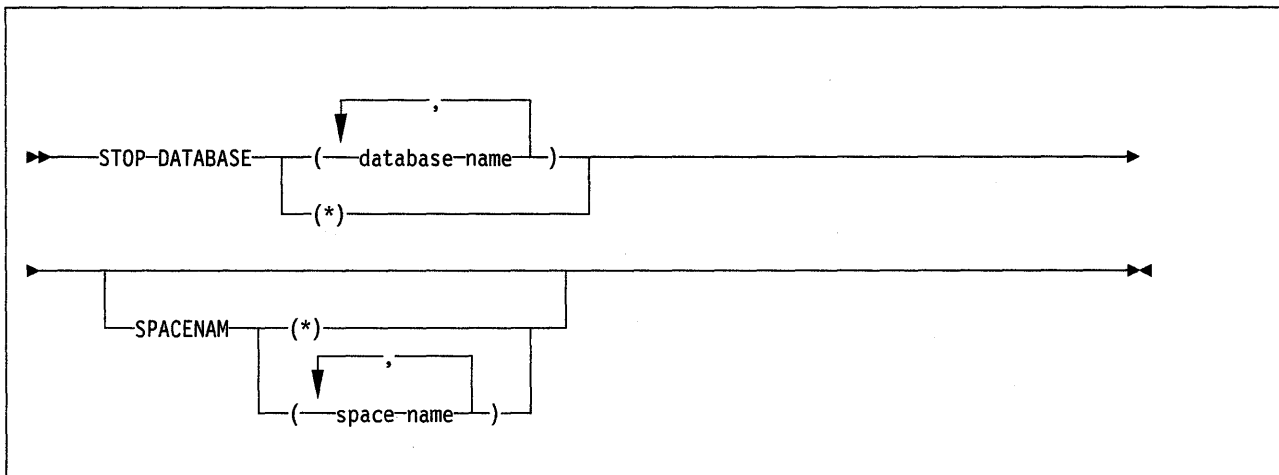
Authorization

The privilege to stop all databases requires no privilege. All databases for which the privilege set defined below has the STOPDB privilege are stopped.

The privilege to stop a list of databases requires no privilege. All databases for which the privilege set has the STOPDB privilege are started. Error messages are produced for those specified databases for which the privilege set does not have the STOPDB privilege.

The STOPDB privilege may have been explicitly granted, or may be inherent in another privilege; it is inherent in the following authorities:

- SYSADM authority
- DBMAINT authority for the database.



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see "DB2 Command Parsing" on page 11.

One of the two following parameters is required.

(database-name)

Is the name of a database to be stopped. If you use more than one name, separate names in the list by commas.

(*) Stops all databases for which the privilege set has at least DBMAINT authority.

SPACENAM

Indicates one or more, or all, names of table spaces or indexes within the specified database that are to be stopped.

Abbreviation: SPACE

(*) Stops all table spaces and indexes of the named database. You cannot use this parameter with more than one database name.

(*space-name*)

Is the name of a table space or index space to be stopped. See "Usage Notes," below, for instructions on how to start them again.

You cannot use this parameter if you use more than one database name, or if you use *.

Usage Notes

If table spaces and indexes are stopped explicitly (using the -STOP DATABASE command with the SPACENAM parameter), they must be started explicitly when the -START DATABASE command is issued. Starting the database will not start table spaces or indexes that have been stopped explicitly.

The message DSN9022I will be issued to indicate that processing is complete. To ensure that all databases have been successfully stopped, issue the -DISPLAY DATABASE command.

Examples

Example 1: Stop table space DSN8S21E in database DSN8D21A.

```
-STOP DATABASE (DSN8D21A)  
  SPACENAM (DSN8S21E)
```

Example 2: Stop all databases (except DSNDDB01, DSNDDB06, and DSNDDB07).

```
-STOP DATABASE (*)
```

-STOP DB2 (DB2)

The DB2 -STOP DB2 command stops the DB2 subsystem.

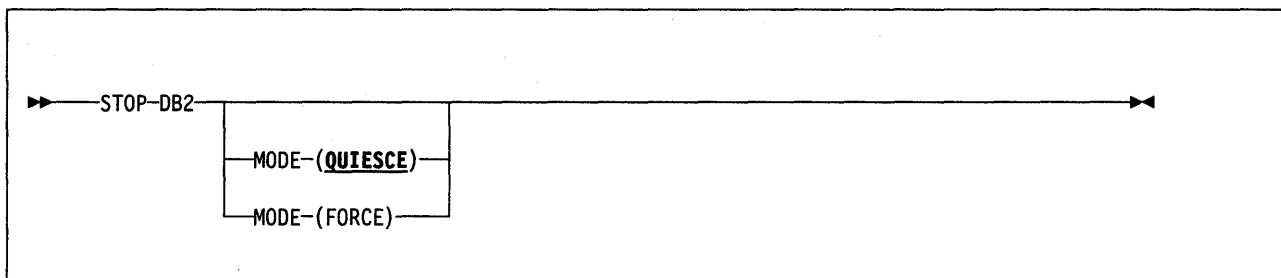
Abbreviation: -STO DB2

Environment

This command can be entered from an MVS console, from the DSN command processor, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

To issue this command, the privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, must include the SYSADM or SYSOPR authorities, or the STOPALL privilege.



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see "DB2 Command Parsing" on page 11.

MODE

Tells whether currently executing programs will be allowed to complete.

(QUIESCE)

Allows currently executing programs to complete processing. No new program is allowed to start. The **default** is **MODE (QUIESCE)**.

(FORCE)

Terminates currently executing programs, including utilities. No new program is allowed to start. MODE FORCE will probably cause indoubt situations.

Usage Notes

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 MVS commands.

Example

Example: Stop the DB2 subsystem. Allow currently active programs to complete. Do not allow new programs to identify to DB2.

```
-STOP DB2 MODE (QUIESCE)
```

STOP irImproc (MVS IRLM)

The STOP irImproc command shuts IRLM down normally. The command is rejected if any active IMS/VS subsystems are currently identified to IRLM.

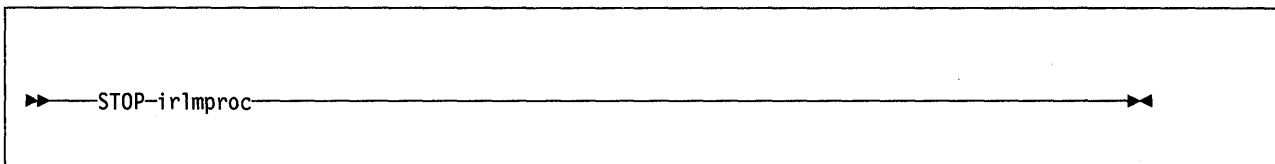
Abbreviation: P

Environment

This command can originate from an MVS console only.

Authorization

The command requires an appropriate level of MVS authority. Please refer to the appropriate MVS/XA or MVS/ESA publication.



Parameter Description

irImproc

Identifies the procedure name for the IRLM to be stopped.

Example

Example: Enter on OS/VS SYSTEM 1 system console:

```
P KRLM1
```

Response on SYSTEM 1 system console:

```
DXR032I KRLM STOP COMMAND ACCEPTED  
DXR011I KRLM END-OF-TASK CLEAN-UP SUCCESSFUL
```

Response on SYSTEM 2 system console:

```
DXR025I JRLM SESSION LOST, SHARING STATE IS IRLM FAILED
```

Explanation: 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.

-STOP RLIMIT (DB2)

The -STOP RLIMIT command 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.

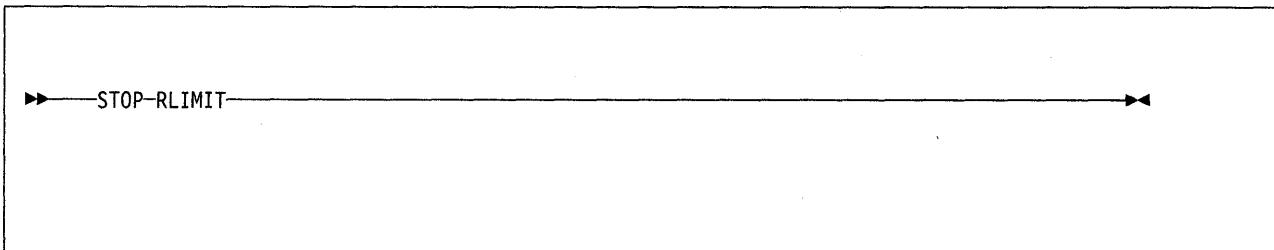
Abbreviation: STOP RLIM

Environment

This command can be entered from an MVS console, from the DSN command processor, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

To issue this command, the privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, must include the SYSADM or SYSOPR authorities, or the TRACE privilege.



-STOP TRACE (DB2)

The DB2 -STOP TRACE command stops tracing. For more information about this trace facility, refer to *Diagnosis Guide and Reference*.

Abbreviation: -STO TRACE

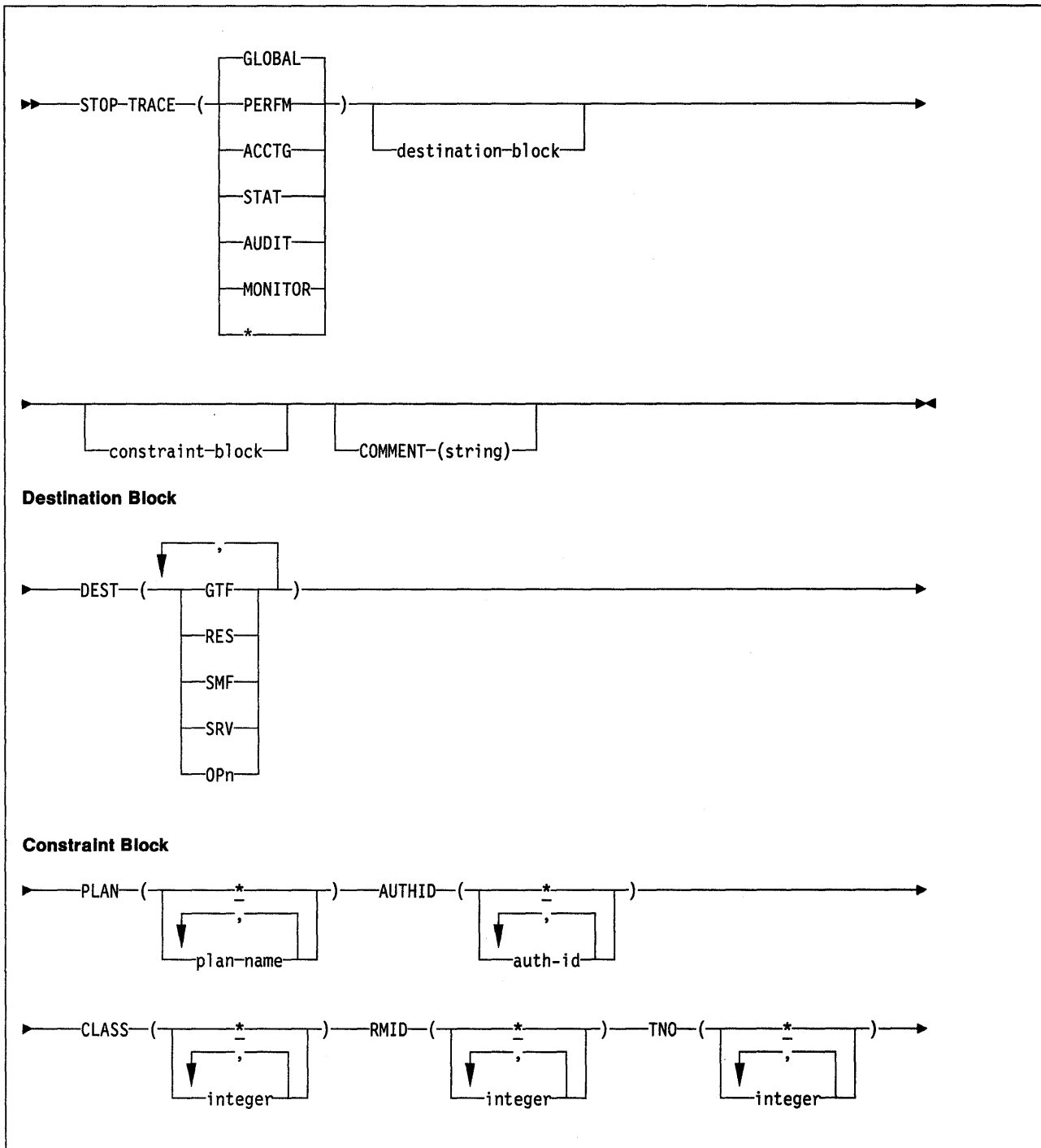
Environment

This command can be entered from an MVS console, from the DSN processor, from the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

To execute this command, the privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, must include the SYSADM or SYSOPR authority, or the TRACE privilege.

-STOP TRACE (DB2)



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see "DB2 Command Parsing" on page 11.

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 command

`-STOP TRACE (PERFM) CLASS (1,2)`

stops only the active traces that were started using the options PERFM and CLASS (1,2); it does *not* stop, for example, any trace started using CLASS(1).

You must specify a trace type or an asterisk. `-STOP TRACE (*)` stops all active traces.

Each of the following keywords limits the command to stopping traces of the corresponding type. For fuller descriptions of each type, see "-START TRACE (DB2)" on page 129.

Type	Description
GLOBAL	Service data from the entire DB2 subsystem Abbreviation: G
PERFM	Performance records of specific events Abbreviation: P
ACCTG	Accounting records for each transaction Abbreviation: A
STAT	Statistical data Abbreviation: S
AUDIT	Audit data Abbreviation: AU
MONITOR	Monitor data Abbreviation: 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 may use more than one value, but do not use the same value twice. If you do not specify a value, the list is not limited.

Abbreviation: D

Possible values and their meanings are:

Value	Trace destination
GTF	The Generalized Trace Facility
RES	A wrap-around table residing in CSA storage
SMF	The System Management Facility
SRV	A serviceability routine reserved for problem diagnosis
OP <i>n</i>	A specific destination. <i>n</i> can be a value from 1 to 8.

-STOP TRACE (DB2)

See “-START TRACE (DB2)” on page 129 for a list of allowable destinations for each trace type.

PLAN (*plan-name*)

Limits stopping to traces started for particular application plans. You may use up to 8 plan names. If you use more than one name, you may use only one value for AUTHID. Do not use this option with STAT or AUDIT.

The **default** is **PLAN(*)**, which does not limit the command.

AUTHID (*authorization-id*)

Limits stopping to traces started for particular authorization identifiers. You may use up to 8 identifiers. If you use more than one identifier, you may use only one value for PLAN. Do not use this option with STAT or AUDIT.

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 129. You may not specify a class if you did not specify a trace type.

Abbreviation: C

The **default** is **CLASS(*)**, which does not limit the command.

RMID (*integer*)

Limits stopping to traces started for particular resource managers. For descriptions of the allowable classes resource manager identifiers, see “-START TRACE (DB2)” on page 129. Do not use this option with STAT, ACCTG, or AUDIT.

The **default** is **RMID(*)**, which does not limit the command.

TNO

Limits stopping to particular traces, identified by their trace numbers (1 to 32, 01 to 09). You may use up to 8 trace numbers. If you use more than one number, you may use only one value each for PLANID and AUTHID.

The **default** is **TNO(*)**, which does not limit the list.

Examples

Example 1: Stop all traces that have the Generalized Trace Facility as their only destination.

```
-STOP TRACE (*) DEST (GTF)
```

Example 2: Stop trace number 4.

```
-STOP TRACE (P) TNO(4)
```

Example 3: Stop all active traces of any kind.

```
-STOP TRACE (*)
```

Example 4: Stop all performance traces.

```
-STOP TRACE=P
```

Example 4: Stop all monitor tracing.

```
-STOP TRACE(MON)
```

-TERM UTILITY (DB2)

The DB2 -TERM UTILITY command terminates execution of a utility job step, but it does not release all resources associated with the step. If the utility is executing, it terminates at the next commit point. If the utility is stopped, some of its resources are released by -TERM.

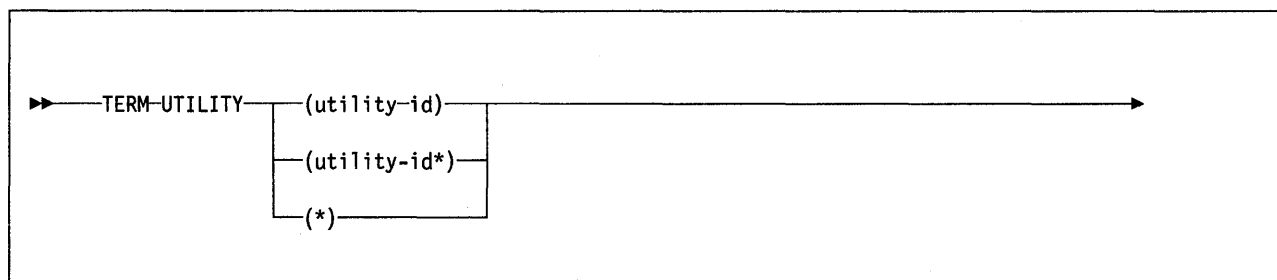
Abbreviation: -TERM UTIL

Environment

This command can be entered from an MVS console, from the DSN command processor, from the DB2I utilities panel, the DB2I commands panel, or from an IMS/VS or CICS terminal.

Authorization

To execute this command, the privilege set defined below must include SYSADM or SYSOPR authority, or include the primary authorization ID of the process that originally submitted the utility job for termination.



Keyword and Parameter Descriptions

For a description of how commands are parsed, and alternatives that may be used in writing keywords and parameters, see "DB2 Command Parsing" on page 11.

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*. *control-file-name* is formed of the letters DSNU, followed by the first three letters of the name of the utility.

If the *utility-id* was created by default by the EXEC statement invoking DSNU TLIB, 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.

(*)

Terminates every utility job step known to DB2.

Usage Notes

Restartability: A terminated utility job step cannot be restarted by any of the methods described in "Chapter 3. Running DB2 Utilities" on page 151. You must resubmit the step as a new utility job.

What Happens to Particular Utilities: In some cases, terminating a utility job may leave work in an undesirable state, requiring special processing before the job can be resubmitted. The following list tells the effects of -TERM on jobs for each of the utilities:

Utility	Special Effects of -TERM
CHECK	None.
COPY	Deletes the output data set and turns on the copy pending restriction. Run COPY again, making a full image copy.
DIAGNOSE	None.
LOAD	See Figure 33 on page 243 for a table of the effect of -TERM on LOAD phases.
MERGECOPY	None.
MODIFY	None.
QUIESCE	None.
RECOVER	Places the object being recovered in recovery pending state.
REORG	See Figure 34 on page 271, for a table of the effect of -TERM on REORG 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/VS)

The IMS/VS /TRACE command directs and controls the IMS/VS capabilities for tracing internal IMS/VS events. It also starts, stops, and defines the activity to be monitored by the IMS/VS DC Monitor.

Abbreviation: /TRA

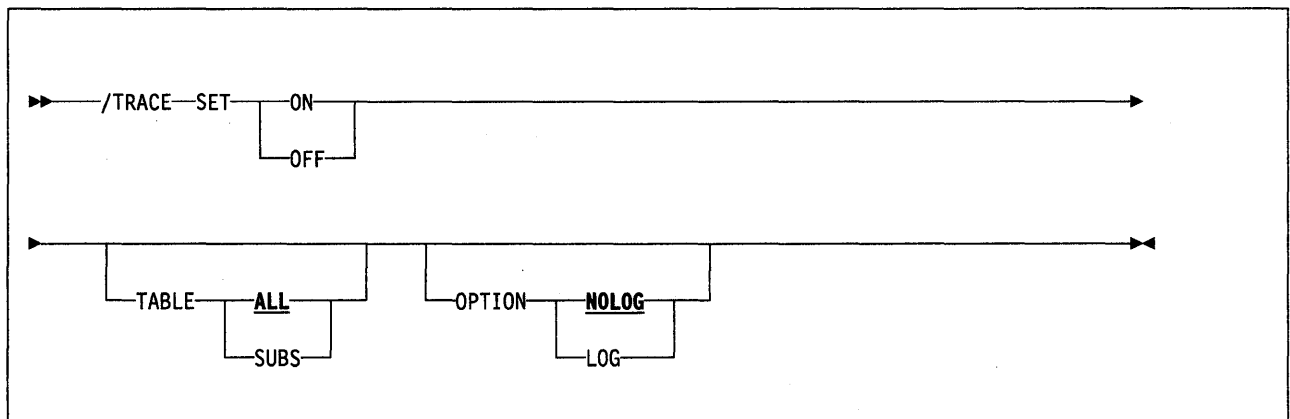
Environment

This command originates from an IMS/VS terminal.

Authorization

To enter this command, users must have passed the IMS/VS security check, as described in *IMS/VS Version 2 System Administration Guide*.

The syntax diagram below includes only those parameters that DB2 users need to know. For a diagram with the complete syntax of this command, refer to *IMS/VS Operator's Reference Manual*.



Keyword and Parameter Descriptions

The parameter descriptions for the /TRACE command are described in *IMS/VS Operator's Reference Manual*; however, this section provides information about the two parameters that are especially important for DB2 users.

SUBS

This parameter follows the TABLE keyword parameter. It 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 parameter, then the default is ALL; ALL includes SUBS, as well as other trace tables.

LOG

This parameter follows the OPTION keyword (which follows the TABLE keyword parameter) to specify that traced data should be written to the IMS/VS system log. Because IMS/VS has a tracing mechanism that writes trace entries to the IMS/VS system log, it is important that DB2 users specify SET ON and TABLE OPTION LOG. Otherwise, the trace information that IMS/VS provides will not be available unless a control region dump occurs.

Examples

Example 1: This command starts IMS/VS tracing and:

- Enables the DB2 trace.
- Writes IMS/VS trace tables to the IMS/VS log before they wrap.

`/TRACE SET ON TABLE SUBS OPTION LOG`

Example 2: This command starts IMS/VS tracing and:

- Enables all trace tables (including DB2's). (ALL is the default parameter for the TABLE keyword.)
- Writes IMS/VS trace tables to the IMS/VS log before they wrap.

`/TRACE SET ON TABLE ALL OPTION LOG`

Chapter 3. Running DB2 Utilities

Description of Utilities	153
Stand-alone Utilities	155
Service Aids	155
Data Sets Used by Utilities	156
Invoking Utilities	158
Using the DB2 Utilities Panel in DB2I	158
Using the DSNU CLIST in TSO	160
Using the Supplied JCL Procedure	163
Creating the JCL Yourself	166
Monitoring and Controlling Utilities	167
Monitoring Utility Status	167
Failure to Complete Execution	168
Restarting a Utility	168
Terminating a Utility	169
CHANGE LOG INVENTORY (DSNJU003) (Utility)	170
CHECK (Utility)	178
COPY (Utility)	185
DIAGNOSE (Utility)	190
DSN1CHKR (Service Aid)	194
DSN1COPY (Service Aid)	200
DSN1LOGP (Service Aid)	209
DSN1PRNT (Service Aid)	221
LOAD (Utility)	224
MERGECOPY (Utility)	247
MODIFY (Utility)	251
PRINT LOG MAP (DSNJU004) (Utility)	254
QUIESCE (Utility)	255
RECOVER (Utility)	257
REORG (Utility)	266
REPAIR (Utility)	273
REPORT (Utility)	285
RUNSTATS (Utility)	288
STOSPACE (Utility)	295

This section contains procedures and guidelines for running, terminating, and displaying the status of DB2 utilities. It is designed as an overview and presupposes that you are familiar with DB2 functions and facilities.

This section describes each DB2 utility and details requirements that must be met before running them. It outlines four different methods of invoking the utilities and explains how to monitor and control utilities.

Description of Utilities

DB2 utilities run as standard MVS batch jobs. They require DB2 to be running. They do not run under control of the terminal monitor program (TMP), but have their own attach mechanisms. They invoke DB2 control facility services (such as the log manager) directly.

Utility	Description
CHECK	<p>The CHECK INDEX option tests whether indexes are consistent with the data they index, and issues warning messages when it finds an inconsistency.</p> <p>The CHECK DATA option checks table spaces for violations of referential constraints, and reports information about each violation it detects.</p>
COPY	<p>Creates an image copy of a table space or a data set within the table space. There are two types of image copies:</p> <ul style="list-style-type: none">• A full image copy is a copy of all pages in a table space or data set.• An incremental image copy is a copy only of pages that have been modified since the last use of the COPY utility.
DIAGNOSE	<p>Generates information useful in diagnosing problems. DIAGNOSE is intended for use after direction by your IBM Support Center.</p>
LOAD	<p>Loads data into one or more tables in a table space or partition. It may also be used to replace the contents of a single partition or an entire table space. The LOAD DATA statement describes the data to be loaded and provides information needed for allocating resources. The loaded data is processed by any edit or validation routine associated with the table, and any field procedure associated with any column of the table.</p>
MERGECOPY	<p>Merges several incremental copies of a table space to make one incremental copy, and merges incremental copies with a full image copy to make a new full image copy.</p>
MODIFY	<p>Deletes records of unwanted copies from the SYSIBM.SYSCOPY catalog table and records of related log records from the SYSIBM.SYSLGRNG directory. You can remove records that were written before a specific date or you can remove records of a specific age, or all records including those created on the current day.</p> <p>You can access database tables while MODIFY is running, but not SYSIBM.SYSCOPY or SYSIBM.SYSLGRNG.</p>

QUIESCE	Establishes a quiesce point (the current log RBA) for a table space, or list of table spaces, and records it in the SYSIBM.SYSCOPY catalog table.
RECOVER	<p>Recovers data to the current state. You can also RECOVER to a prior copy or a specified log RBA. The largest unit of data recovery is the table space; the smallest is the page. You can recover one table space, a list of table spaces, a data set, pages within an error range, or a single page. Data is recovered from image copies of a table space and database log change records. If the most recent full image copy data set is unusable, and there are previous image copy data sets existing in the system, RECOVER uses the previous image copy data sets.</p> <p>It also recovers multiple indexes over tables that reside in a common table space. Indexes are recovered in their entirety with a single invocation of RECOVER.</p> <p>For recovery of an error range, only pages on the same track as the damaged pages are unavailable while RECOVER is running.</p>
REORG	Reorganizes a table space to improve access performance and reorganizes indexes so that they are more efficiently clustered. If you specify REORG UNLOAD ONLY, the REORG utility unloads data in a format acceptable to the LOAD utility of the same DB2 subsystem.
REPAIR	<p>Replaces invalid data with valid data. The data may be your own data, or data you would not normally access, such as space map pages and index entries.</p> <p>Be extremely careful in using REPAIR: Improper use can damage the data even further.</p> <p>The REPAIR utility is discussed under "REPAIR (Utility)" on page 273 and in Section 1 of <i>Diagnosis Guide and Reference</i>.</p>
REPORT	<p>The REPORT utility reports information necessary for recovering a table space. REPORT reports:</p> <ul style="list-style-type: none"> • recovery history from the SYSIBM.SYSCOPY catalog table • log ranges from SYSIBM.SYSLGRNG • log data sets from the bootstrap data set (BSDS) • names of all table spaces and tables in a table space set.
RUNSTATS	<p>Scans a table space or indexes to gather information about</p> <ul style="list-style-type: none"> • Utilization of space • Efficiency of indexes <p>The information is recorded in the DB2 catalog, and is used by the SQL optimizer to select access paths to data during the bind process. It is available to the database administrator for evaluating database design, and determining when table spaces or indexes need to be reorganized.</p>
STOSPACE	Updates DB2 catalog columns that indicate how much space is allocated for storage groups and related table spaces and indexes.

Stand-alone Utilities

The following utilities are stand-alone utilities that can be invoked only by means of MVS JCL. Print Log Map can be used while DB2 is running; Change Log Inventory can only be used while DB2 is not running.

Utility	Description
Change Log Inventory	Changes the bootstrap data sets (BSDSs). You can use Change Log Inventory to add or delete active or archive log data sets or to create a conditional restart control record to control the next start of the DB2 subsystem. You may not run this utility when DB2 is running. For instructions on using Change Log Inventory, refer to "CHANGE LOG INVENTORY (DSNJU003) (Utility)" on page 170.
Print Log Map	Prints the log data set name and log RBA association for both copy 1 and copy 2 of all active and archive log data sets. Print Log Map prints the passwords for those data sets, if provided, and the active log data sets available for new log data. It also prints the status of all conditional restart control records in the bootstrap data set and the contents of the queue of checkpoint records in the bootstrap data set. For instructions on using the Print Log Map, refer to "PRINT LOG MAP (DSNJU004) (Utility)" on page 254.

Service Aids

The following service aids are stand-alone aids and can be invoked only by means of MVS JCL. DSN1CHKR, DSN1COPY, and DSN1PRNT can be used while DB2 is running; DSN1LOGP can only be used while DB2 is not running.

Service Aid	Description
DSN1CHKR	Scans a specified table space for broken links, hash chains, and orphans. You can print table space pages in formatted or dump form, or print each record individually as DSN1CHKR follows its chain through the table space.
DSN1COPY	Copies DB2 VSAM data sets that contain table space or index spaces. Other data sets you can copy are: <ul style="list-style-type: none">• VSAM to sequential• VSAM to VSAM• sequential to VSAM• sequential to sequential You can also copy hexadecimal dumps of DB2 data sets and databases, perform validity checking on data or index pages, and perform OBID translation to enable moving data sets between different systems.
DSN1LOGP	Reads the contents of the recovery log and formats the contents for display. The two formats are: <ul style="list-style-type: none">• A <i>detail report</i> that formats and displays individual log records. This information is used by IBM Support Center personnel to resolve problems that

require extensive analysis of the recovery log contents.

- A *summary report* that summarizes the information contained in the log to help you perform a conditional restart.

You may not use DSN1LOGP while DB2 is running.

DSN1PRNT

Prints DB2 VSAM data sets that contain table spaces or index spaces. It is especially useful when you want to examine the contents of a table or index.

You can print hexadecimal dumps of DB2 data sets and databases using this service aid. You can also print:

- Image copy data sets
- Sequential data sets

Data Sets Used by Utilities

A SYSIN DD statement is needed in every utility job to describe an input data set; some utilities also need other data sets. Figure 14 lists the name of each DD statement that may be needed, the utilities that require it, and the purpose of the corresponding data sets. If an alternate DD name is allowed, you give it as a parameter in a utility option; the figure also lists the option keywords used.

DD Name	Used by By	Purpose	Option Keyword
SORTLIB	defined locally	Contains the DFSORT load module. If you omit this and sorting is needed, SYS1.SORTLIB is dynamically allocated.	-
SORTOUT	LOAD, REORG (required for tables with indexes)	Holds sorted keys (sort output) to allow SORT phase to be restartable.	WORKDDN
	CHECK DATA (required)	Holds sorted keys (sort output).	
SORTWKnn	CHECK INDEX, LOAD, REORG, RECOVER INDEX	Work data set for sorting indexes. nn is a 2-digit number; you may use several data sets. To estimate the size of the data set needed, see "Work Data Set" on page 240.	-
SYSCOPY	COPY	Output data set for copies.	COPYDDN
SYSDISC	LOAD	Discarded records (optional).	DISCARDN
SYSERR	LOAD, CHECK DATA (required)	Contains information about errors encountered during processing.	ERRDDN

Figure 14 (Part 1 of 2). Data Sets Used by Utilities

DD Name	Used by By	Purpose	Option Keyword
SYSIN	All	Input data set for utility statements.	-
SYSMAP	LOAD	Contains information about where input data set records are loaded.	MAPDDN
SYSPRINT	All	Messages and printed output. (Usually SYSOUT.)	-
SYSREC	LOAD	LOAD input data set.	INDDN
	REORG	Unloaded records for REORG.	UNLDDN
SYSUT1	CHECK INDEX, LOAD (required for tables with indexes), REORG, RECOVER INDEX	Temporary work data set that holds sorted keys on (sort output) if SORTOUT data set is not provided.	WORKDDN
	MERGE	Temporary work data set that holds intermediate merged output.	WORKDDN
UTPRINT	CHECK INDEX, LOAD, REORG, RECOVER INDEX	Messages from DFSORT. (Usually, SYSOUT or DUMMY.)	-

Figure 14 (Part 2 of 2). Data Sets Used by Utilities

To prevent unauthorized access to data sets (for example, image copies), you can protect the data sets with passwords or by using the Resource Access Control Facility (RACF). To use a utility with a data set protected by RACF, you must be authorized to access the data set.

Controlling Data Set Disposition

Most data sets need to exist only during utility execution (for example, during reorganization). However, several data sets should be kept in certain circumstances.

- Retain COPYDDN until it is no longer needed for recovery.
- Retain UNLDDN if UNLOAD(PAUSE) or UNLOAD(ONLY) has been specified for the REORG utility.
- Retain DISCARDN until the contents are no longer needed for subsequent loads.

Because you may need to restart a utility, take the following precautions when defining the disposition of data sets:

- Use DISP = (MOD,KEEP,CATLG) or DISP = (MOD,CATLG) for data sets you want to retain.

- Use DISP= (MOD,DELETE,CATLG) for data sets that will be discarded after utility execution.
- Use DISP= (NEW,DELETE) for DFSORT SORTWKnn data sets, or refer to *DFSORT Application Programming: Guide* for alternatives.
- Do NOT use temporary data set names, such as &&WORK.

Invoking Utilities

DB2 provides four ways to invoke the utilities: each is described separately in one of the following sections.

- “Using the DB2 Utilities Panel in DB2I”
- “Using the DSNU CLIST in TSO” on page 160
- “Using the Supplied JCL Procedure” on page 163
- “Creating the JCL Yourself” on page 166

For the least involvement with JCL, choose one of the first two options.

In using either of these options, we recommend that you edit the generated JCL to alter or add necessary fields on the JOB or ROUTE cards before submitting the job. You can run the utilities by supplying the necessary parameters and data set names. Both of these methods require TSO, and the first also requires access to the DB2 Utilities Panel in DB2 Interactive (DB2I).

If you want to interact closely with JCL, or supply your own JCL, choose one of the last two methods.

Using the DB2 Utilities Panel in DB2I

This method of invoking DB2 utilities requires the least knowledge of JCL.

If your site does not have default Job and Route cards, you must edit the JCL to define them. If you edit the utility job before submitting it, you must use the ISPF editor, and must submit your job directly from the editor.

1. Create the utility statement for the utility you intend to invoke, and save it in a sequential or partitioned data set.

For example, the following utility statement makes an incremental image copy of table space DSN8D21A.DSN8S21D with a SHRLEVEL value of CHANGE.

```
COPY TABLESPACE DSN8D21A.DSN8S21D
  FULL NO
  SHRLEVEL CHANGE
  DEVT SYSDA
```

For the rest of this example, suppose that you save the statement in the default data set, UTIL.

2. Select the DB2I menu from the ISPF Primary Option Menu.
3. Select the UTILITIES option on the DB2I Menu. This panel is shown in Figure 15 on page 159. Items you must enter are highlighted.

```

DSNEUP01                                DB2 UTILITIES
====>

Select from the following:

1 FUNCTION ====> EDITJCL  (SUBMIT job, EDITJCL, DISPLAY, TERMINATE)
2 JOB ID   ====> TEMP    (A unique job identifier string)
3 UTILITY  ====> COPY    (CHECK, CHECK DATA, COPY, DIAGNOSE, LOAD,
                        MERGE, MODIFY, QUIESCE,
                        RECOVER INDEX, RECOVER TABLESPACE,
                        REORG INDEX, REORG TABLESPACE
                        REPORT, REPAIR, RUNSTATS, STOSPACE)

4 CONTROL CARDS DATA SET  ====> UTIL
5 RECDN (LOAD,              ====>
        REORG TABLESPACE)
6 DISCDN (LOAD)            ====>
7 COPYDSN (COPY, MERGECOPY) ====> 'MYCOPIES.DSN8D21A.JAN1'

To RESTART a utility, specify starting point, otherwise enter NO.
8 RESTART  ====> NO      (NO, At CURRENT position, or beginning of PHASE)

PRESS: ENTER to process   END to exit   HELP for more information

```

Figure 15. DB2 UTILITIES Panel

4. Fill in field 1 with the function you want to execute. In this example, you want to submit the utility job, but you want to edit the JCL first. After you have edited the utility job, enter SUBMIT on the editor command line.
5. Field 2 should be a unique identifier for your utility job. When the panel appears, it contains the default value TEMP. In this example, that value is satisfactory; leave it as is.
6. Fill in field 3 with the utility you want to run. In this example, enter COPY.
7. Fill in field 4 if you want to use an input data set other than the default data set. Unless you enclose the data set name between apostrophes, TSO adds your user identifier as a prefix. In this example, enter UTIL, which is the default data set.
8. Fill in field 5 if you are running LOAD, REORG, or RECOVER INDEX. If you are running LOAD, you must indicate the data set name that contains records to be loaded. If you are running REORG or RECOVER INDEX you must indicate the unload data set. In this example, you do not have to fill in field 5.
9. Fill in field 6 if you are running LOAD with discard processing; you must indicate a discard data set. In this example, you do not have to fill in field 6.
10. Fill in field 7 with an output data set name if you are running COPY or MERGECOPY. In this example, if you want the output copies in 'MYCOPIES.DSN8D21A.JAN1', fill in field 7 (you are using the COPY utility) with 'MYCOPIES.DSN8D21A.JAN1'.
11. Change field 8 only if this job restarts a stopped utility. In this example, leave the default value, NO.
12. Press ENTER.

If you need help while using the DB2 Utilities panel, press the HELP PF key. There are HELP panels to explain the parameters on the DB2 Utilities panel, and HELP panels that give descriptions, syntax, and examples for each of the DB2 utility statements.

Restarting the Utility, Using DB2I

To restart the utility, if it is stopped:

1. Access the DB2 UTILITIES Panel.
2. Fill in the panel fields as shown above, except field 8.
3. Change field 8. Use:
CURRENT to restart at the last commit point
PHASE to restart at the beginning of the phase executed last
4. Press ENTER.

Using the DSNU CLIST in TSO

You can also initiate a DB2 utility by invoking the DSNU CLIST under TSO. The CLIST generates the JCL needed to invoke the DSNUPROC procedure and execute DB2 utilities as batch jobs. When you use the CLIST, you need not be concerned with details of the JCL.

The CLIST creates a job that performs only one utility operation. However, you can invoke the CLIST for each utility operation you need, and then edit and merge the outputs into one job or step.

To use the CLIST:

1. Enter the utility statement for the utility you want to invoke in a sequential or partitioned data set.
2. Invoke the CLIST in order to create the JCL to run the utility. Supply the name of the utility you need and the name of the data set in which you stored the utility control statements.

Figure 16 is an example of how to invoke the CLIST for the COPY utility. The list following the example describes the parameters used. For descriptions of other parameters you may use, refer to "DSNU (TSO CLIST)" on page 88.

```
%DSNU
UTILITY (COPY)
INDSN ('MYCOPY (STATEMNT)')
COPYDSN ('MYCOPIES.DSN8D21A.JAN1')
EDIT (TSO)
SUBMIT (YES)
UID (TEMP)
RESTART (NO)
```

Figure 16. Invoking the DSNU CLIST.

UTILITY (COPY)

Names the utility you want to use: COPY. The CLIST generates JCL in a data set named DSNUCOP.CNTL. DSNUCOP.CNTL contains the statements necessary to invoke the DSNUPROC procedure which, in turn, invokes the COPY utility.

In general, DB2 places the JCL in a data set named DSNUxxx.CNTL, where xxx is the first three letters of the utility name. If you submit another job with the same utility name, the first job is deleted.

INDSN ('MYCOPY(STATEMNT)')

Merges the data set 'MYCOPY.STATEMNT', which contains the COPY utility control statement, into the JCL as SYSIN input.

COPYDSN ('MYCOPIES.DSN8D21AJAN1')

Supplies the name of the cataloged data set into which the copies are placed.

If you omit a value for UTILITY, INDSN, or COPYDSN, the CLIST prompts you for it. Other utilities prompt for other required data set names; for example, LOAD requires, and prompts for, a RECDSN name.

EDIT (TSO)

Invokes the TSO editor to allow you to edit DSNUCOP.CNTL before the utility job is executed.

You can also specify EDIT (ISPF) if you want to use the ISPF editor, or EDIT (NO) if you don't want to edit the data. EDIT (NO) is the default.

SUBMIT (YES)

Submits DSNUCOP.CNTL as a background job after you end the editing session. The file MYCOPY(STATEMNT) is merged into DSNUCOP.CNTL as SYSIN input.

SUBMIT (NO) is the default.

UID (TEMP)

Provides a unique identifier for your job; we have chosen TEMP because it is the default provided by DB2I. The identifier is a way to refer to your job in the -DISPLAY UTILITY or -TERM UTILITY command. Enclose the value between apostrophes if it contains special characters.

The default provided by the DSNU CLIST is *userid*.DSNUCOP (in general, "DSNU" followed by the first three letters of the utility name). We recommend, though, using a value for the UID option.

RESTART (NO)

Specifies that this is a new utility job. There should be no other utility job step with the identifier TEMP.

3. Edit the generated JCL (in DSNUCOP.CNTL) to alter or add DD statements as needed.

This last step is optional. Suppose, however, that you want to change the output destination, and put the copy in 'MYCOPIES.DSN8D21A.JAN8'. The following section explains the steps you should take.

Editing the Generated JCL

The generated JCL consists of a JOB statement, an EXEC statement, and certain DD statements, as shown in Figure 17 on page 162. Any of these statements can be edited and changed.

```

//DSNUCOP JOB your-job-statement-parameters
//          USER=userid,PASSWORD=userword
//*ROUTE PRINT routing-information
//UTIL     EXEC  DSNUPROC,SYSTEM=DSN,UID=TEMP,UTPROC=''
//SYSCOPY  DD    DSN=MYCOPIES.DSN8D21A.JAN1,DISP=(MOD,KEEP,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN    DD    *
            COPY TABLESPACE DSN8D21A.DSN8S21D
            FULL NO
            SHRLEVEL CHANGE
            DEVT SYSDA

/*

```

Figure 17. DSNUCOP.CNTL. This is the JCL before editing.

The following list describes the required statements:

Statement	Description
JOB	The CLIST uses any JOB statements that you had saved when using DB2I. If there are no existing JOB statements, DB2 produces a skeleton JOB statement that you can modify after the JCL is complete. The <i>jobname</i> is DSNU, followed by the first three letters of the name of the utility you are using.
EXEC	The CLIST builds the EXEC statement. The values you specified for SYSTEM (DSN, by default), UID (TEMP), and RESTART (none) become the values of SYSTEM, UID, and UTPROC for DSNUPROC.

After the CLIST builds the EXEC statement, it generates the following DD statements.

SYSCOPY DD Defines the data set to which copies are to be sent. In the example, this is the line you want to edit. Change it to read:

```

//SYSCOPY DD    DSN=MYCOPIES.DSN8D21A.JAN8,DISP=(MOD,KEEP,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(1,1))

```

SYSIN DD Defines the utility input statement data set. This data set can reside on a tape, direct-access device, or system reader. The data set may be in any format readable by BSAM.

Additional Data Sets for Some Utilities

For the CHECK, LOAD, MERGE, or REORG utilities, you may also need DD statements for other data sets. Data sets that may be required are listed under "Data Sets Used by Utilities" on page 156.

Restarting the Utility Using the DSNU CLIST

To restart the utility, if it is stopped, invoke the DSNU CLIST as shown in Figure 16 on page 160, but change the value of the RESTART parameter. Use:

RESTART(CURRENT) to restart at the last commit point
RESTART(PHASE) to restart at the beginning of the first incomplete phase

Using the Supplied JCL Procedure

Another way to initiate a DB2 utility is by using the supplied JCL procedure, DSNUPROC, shown in Figure 19 on page 165. This procedure uses the parameters that you supply to build an appropriate EXEC statement to invoke a DB2 utility.

To invoke the DSNUPROC procedure, you must write and submit JCL like that built by the DSNU CLIST, and shown in Figure 17 on page 162. In your JCL, the EXEC statement invokes the DSNUPROC procedure. The parameters you can supply to that procedure, and their syntax, are explained in the following section.

Syntax of DSNUPROC

Use the syntax in the following figure to write the EXEC DSNUPROC statement in the JCL.

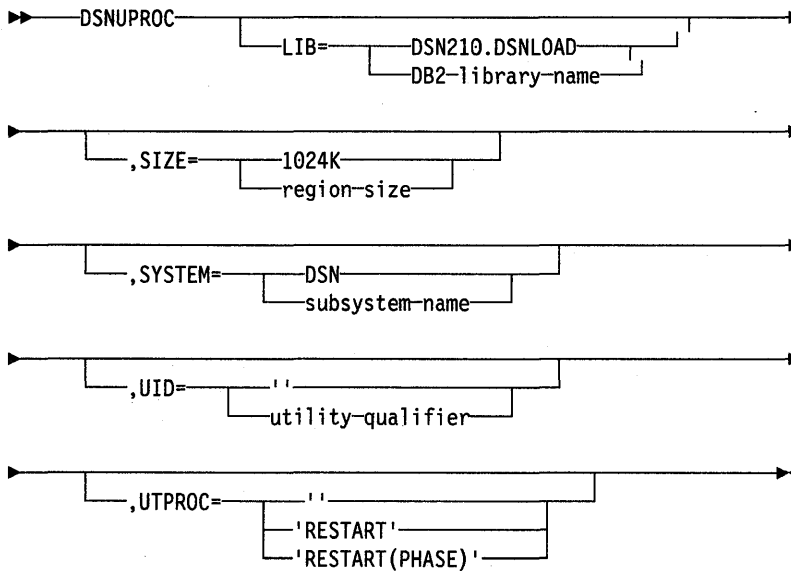


Figure 18. Syntax of DSNUPROC

Keyword and Parameter Descriptions

The following list describes all the parameters. For the example, you need to use only one, UID = TEMP. For all others, you may use the defaults.

LIB =

Specifies the data set name of the DB2 subsystem library. The **default** is **DSN210.DSNLOAD**.

SIZE =

Specifies the region size of the utility execution area; that is, the number of bytes of virtual storage allocated to this utility job. The **default** is **1024K**.

SYSTEM =

Specifies the DB2 subsystem. The **default** is **DSN**.

UID =

Specifies the unique identifier for your utility job. The name may use up to 16 characters. If the name contains special characters, enclose it between apostrophes: for example, 'PETERS.JOB'.

The **default** is an empty string.

UTPROC =

Controls restart processing. The **default** is an empty string, used when you are not restarting a stopped job.

To restart the utility, use:

'RESTART' to restart at the last commit point.

'RESTART(PHASE)' to restart at the beginning of the phase executed last.

The procedure provides the SYSPRINT and UTPRINT DD statements for printed output. You must provide DD statements for SYSIN and whatever other data sets are needed by your job. See "Data Sets Used by Utilities" on page 156 for a description of data sets that may be needed.

The following example is the DSNUPROC procedure invoked by the JCL example in Figure 17 on page 162.


```

//DSNUPROC PROC LIB='DSN210.DSNLOAD',
//      SYSTEM=DSN,
//      SIZE=1024K,UID='',UTPROC=''
//*
//*****
//*
//* PROCEDURE-NAME:      DSNUPROC
//*
//* DESCRIPTIVE-NAME:   UTILITY PROCEDURE
//*
//* COPYRIGHT = 5740-XYR (C) COPYRIGHT IBM CORP 1982, 1987
//* REFER TO COPYRIGHT INSTRUCTIONS FORM NUMBER G120-2083
//*
//* STATUS:              VERSION 2, RELEASE 1
//*
//* FUNCTION: THIS PROCEDURE INVOKES THE ADMF UTILITIES IN THE
//*           BATCH ENVIRONMENT
//*
//* PROCEDURE-OWNER:    UTILITY COMPONENT
//*
//* COMPONENT-INVOKED:  DB2 UTILITIES (ENTRY POINT DSNUTILB).
//*
//* ENVIRONMENT:        TSO BATCH
//*
//* INPUT:
//*   PARAMETERS:
//*     LIB = THE DATA SET NAME OF THE DB2 PROGRAM LIBRARY.
//*           THE DEFAULT LIBRARY NAME IS PREFIX.DSNLOAD,
//*           WITH PREFIX SET DURING INSTALLATION.
//*     SIZE = THE REGION SIZE OF THE UTILITIES EXECUTION AREA.
//*           THE DEFAULT REGION SIZE IS 1024K.
//*     SYSTEM = THE SUBSYSTEM NAME USED TO IDENTIFY THIS JOB
//*             TO DB2. THE DEFAULT IS "DSN".
//*     UID = THE IDENTIFIER WHICH WILL DEFINE THIS UTILITY
//*           JOB TO DB2. IF THE PARAMETER IS DEFAULTED OR
//*           SET TO A NULL STRING, THE UTILITY FUNCTION WILL
//*           USE ITS DEFAULT, USERID.JOBNAME. EACH UTILITY
//*           WHICH HAS STARTED AND IS NOT YET TERMINATED
//*           (MAY NOT BE RUNNING) MUST HAVE A UNIQUE UID.
//*     UTPROC = AN OPTIONAL INDICATOR USED TO DETERMINE WHETHER
//*             THE USER WISHES TO INITIALLY START THE REQUESTED
//*             UTILITY OR TO RESTART A PREVIOUS EXECUTION OF
//*             THE UTILITY. IF OMITTED, THE UTILITY WILL
//*             BE INITIALLY STARTED. OTHERWISE, THE UTILITY
//*             WILL BE RESTARTED BY ENTERING THE FOLLOWING
//*             VALUES:
//*       RESTART(PHASE) = RESTART THE UTILITY AT THE
//*                       BEGINNING OF THE PHASE EXECUTED
//*                       LAST.
//*       RESTART = RESTART THE UTILITY AT THE LAST
//*                 OR CURRENT COMMIT POINT.

```

Figure 19 (Part 1 of 2). Sample Listing of Supplied JCL Procedure (DSNUPROC)

```

/**                                     *
/** OUTPUT: NONE.                       *
/**                                     *
/** EXTERNAL-REFERENCES: NONE.         *
/**                                     *
/** CHANGE-ACTIVITY:                   *
/**                                     *
/**                                     *
/*******
/**
/**DSNUPROC EXEC PGM=DSNUTILB,REGION=&SIZE,
/**      PARM=(&SYSTEM,'&UID','&UTPROC')
/**STEPLIB DD  DSN=&LIB,DISP=SHR
/**
/*******
/**                                     *
/*** THE FOLLOWING DEFINE THE UTILITIES' PRINT DATA SETS
/***                                     *
/*******
/**
/**SYSPRINT DD  SYSOUT=*
/**UTPRINT  DD  SYSOUT=*
/**SYSUDUMP DD  SYSOUT=*
/***DSNUPROC PEND      REMOVE * FOR USE AS INSTREAM PROCEDURE

```

Figure 19 (Part 2 of 2). Sample Listing of Supplied JCL Procedure (DSNUPROC)

Creating the JCL Yourself

DB2 utilities execute as standard OS/VS jobs. In order to execute the utility, you must supply the JOB statement required by your installation and the JOBLIB or STEPLIB DD statements required to access DB2. You must also use an EXEC statement and a set of DD statements. The EXEC statement is described in the following section; for a description of the data sets you may need DD statements for, see "Data Sets Used by Utilities" on page 156.

EXEC Statement

The EXEC statement can be a procedure that contains the required JCL, or it can be of the form:

```
//stepname EXEC PGM=DSNUTILB,PARM='system,[uid],[utpr:]'
```

In this statement, brackets, [], indicate optional parameters. The parameters have the following meanings:

DSNUTILB

Names the utility control program. The program must reside in an APF-authorized library.

system

Names the DB2 subsystem.

uid

Is the unique identifier for your utility job.

utproc

Is the value of the UTPROC parameter in DSNUPROC. Use it only when restarting the utility job. Use:

- 'RESTART' To restart at the last commit point.
- 'RESTART(PHASE)' To restart at the beginning of the phase executed last.

In our example, you would write:

```
//stepname EXEC PGM=DSNUTILB,PARM='DSN,TEMP'
```

Monitoring and Controlling Utilities

The following sections describe how to monitor utility status and how to terminate a utility.

Monitoring Utility Status

You can learn the current status of utilities by issuing the DB2 -DISPLAY UTILITY command. DB2 returns a message indicating the utility name, identifier, phase, and state. The message also indicates the number of pages or records processed by the utility.

Utility States

A DB2 utility may be in one of these states:

- Active: The utility has started execution and has been granted access to the required table spaces or index spaces.
- Stopped: The utility has stopped executing, but data that has been changed remains unavailable to you. To make that data available again, you must either:
 - Correct the condition that stopped the utility, and restart the utility so that it runs to termination; or
 - Terminate the utility with the DB2 -TERM UTILITY command (see “Terminating a Utility” on page 169).
- Terminated: The utility has been requested to terminate by the DB2 -TERM UTILITY command. If the utility has terminated, there is no message.

Utility Phases

The functions performed by a utility are divided into phases, which indicate progress of the utility; some utilities can be restarted at the beginning of the last executed phase. Output from -DISPLAY UTILITY shows the currently executing phase.

The phases are named as follows:

- UTILINIT Perform initialization and setup for the utility. If you restart a utility in this phase, it executes as a new utility.
- utility-name For many utilities, this is the only processing phase, and bears the name of the utility: COPY, MERGECOPY, MODIFY, RUNSTATS, REPAIR, or STOSPACE.
- various CHECK, LOAD, RECOVER, and REORG have more than three phases. For their descriptions, see the sections on the several utilities in this publication.
- UTILTERM Perform cleanup for the utility. A utility that has stopped in this phase has already completed most of its processing.

Failure to Complete Execution

If a utility job completes normally, it issues return code 0. If it completes, but with warning messages, it issues return code 4. Return code 8 means failure to complete.

During execution of the utility, any of these problems can cause a failure:

- **Problem:** DB2 terminates the utility job step and any later utility steps.
Solution: Submit a new utility job to execute the terminated steps; use the same utility identifier for the new job, to ensure that there is no duplicate utility running.
- **Problem:** The particular utility function is not executed by DB2, but prior utility functions are executed.
Solution: Submit a new utility step to execute the function.
- **Problem:** The utility function is placed in the stopped state by DB2.
Solution: Restart the utility job step at either the last commit point or the beginning of the phase, using the same utility identifier. Alternately, terminate the job step (using `-TERM UTILITY(uid)`) and resubmit it.

Restarting a Utility

When you restart a utility job step, do not delete the utility statements that appear before the stopped step. Also, do not change the utility function that is currently stopped and the DB2 objects upon which it is operating. However, you can change other parameters relating to the stopped step and later utility steps.

The utility can be restarted at the last commit point or at the beginning of the current (incomplete) phase. See “Invoking Utilities” on page 158 for a more detailed discussion of invoking utilities.

Do not specify MVS automatic step restart.

When restarting a utility, do not specify `VOL=SER=`. Let DB2 determine the data sets from the system catalog.

There are special considerations when restarting a utility that received an out of space condition on the work data set (for example, ABEND37 on SYSUT1). See Figure 14 on page 156 for a list of utilities that use a work data set.

Before restarting a utility:

1. Copy the work data set.
2. Redefine the data set with additional space, using the same VOLSER, the same DSNAME, and the same DCB parameters.
3. Copy the data back into the work data set.

Note: The data set name and first VOLSER are used to keep track of utility restart information. They must be the same as the original utility run.

Terminating a Utility

Use the `-TERM UTILITY` command to terminate the execution of an active utility or release the resources associated with a stopped utility.

Caution: After you issue the `-TERM UTILITY` command, the terminated utility cannot be restarted. In addition, the objects on which the utility was operating may be left in an indeterminate state. In many cases you cannot rerun the same utility without first recovering the objects on which the utility was operating. The situation varies, depending on the utility and the phase that was in process when you issued the command. These considerations are particularly important when terminating the `COPY`, `LOAD`, and `REORG` utilities.

Using `-TERM UTILITY`

If you cannot restart a utility, you may have to terminate it in order to make the data available to other applications. To terminate, issue the `DB2 -TERM UTILITY` command. Use the command only if you must start the utility from the beginning.

If the utility is active, `-TERM UTILITY` terminates it at the next commit point. It then performs any necessary cleanup operations.

You may choose to put the `-TERM UTILITY` command in a conditionally executed job step; for example, if you never want to restart certain utility jobs. The following shows a sample job stream for our `COPY` example:

```
//TERM EXEC PGM=IKJEFT01,COND((8,GT,S1),EVEN)
//*
//*****
//* IF THE PREVIOUS UTILITY STEP, S1, TERMINATED WITH A
//* RETURN CODE >= 8, OR IF IT ABENDED, ISSUE A TERMINATE
//* COMMAND FOR IT. IT WILL NOT BE RESTARTED.
//*****
//*
//SYSPRINT DD SYSOUT=A
//SYSTSPRT DD SYSOUT=A
//SYSOUT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSTSIN DD *
DSN S(DB2)
-TERM UTILITY(TEMP)
END
/*
```

CHANGE LOG INVENTORY (DSNJU003) (Utility)

The CHANGE LOG INVENTORY utility changes the bootstrap data sets (BSDSs).
The utility can be used to:

- Add or delete active or archive log data sets
- Supply passwords for archive logs and DB2 system databases
- Create a conditional restart control record to control the next start of the DB2 subsystem
- Change the VSAM catalog name entry in the BSDS.

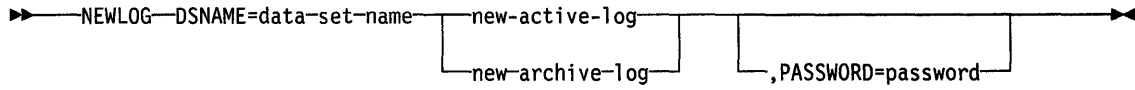
Environment

The utility can be executed only as a batch job and only when DB2 is not running.

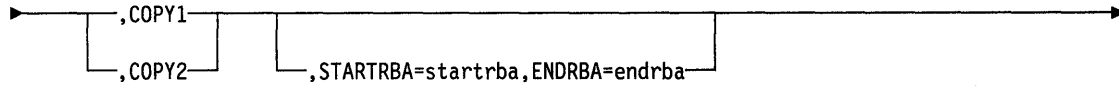
Authorization

To use this utility, the primary authorization ID designated by the process must have either the requisite RACF authorization or, if the BSDS is password protected, the appropriate VSAM password for the data set.

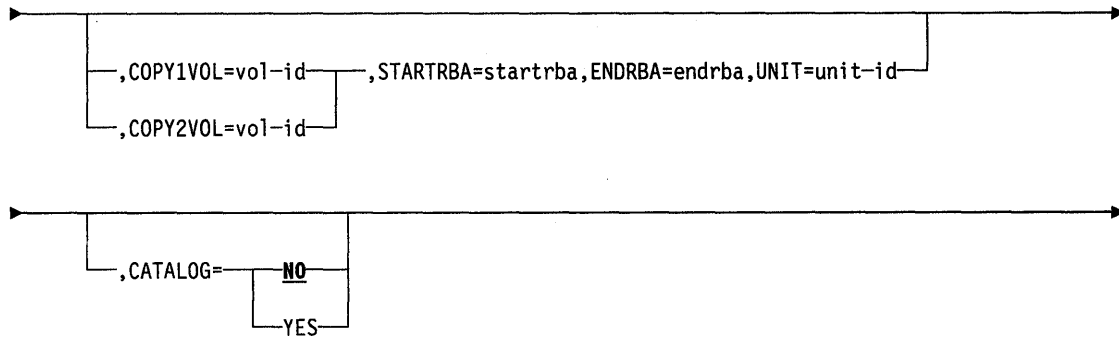
NEWLOG Statement



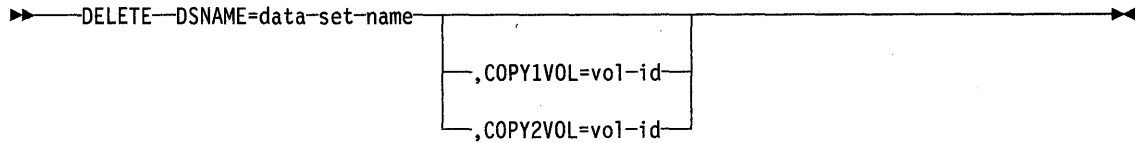
New-Active-Log



New-Archive-Log



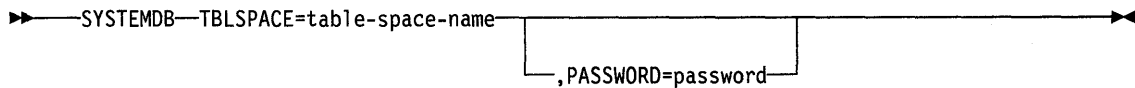
DELETE Statement

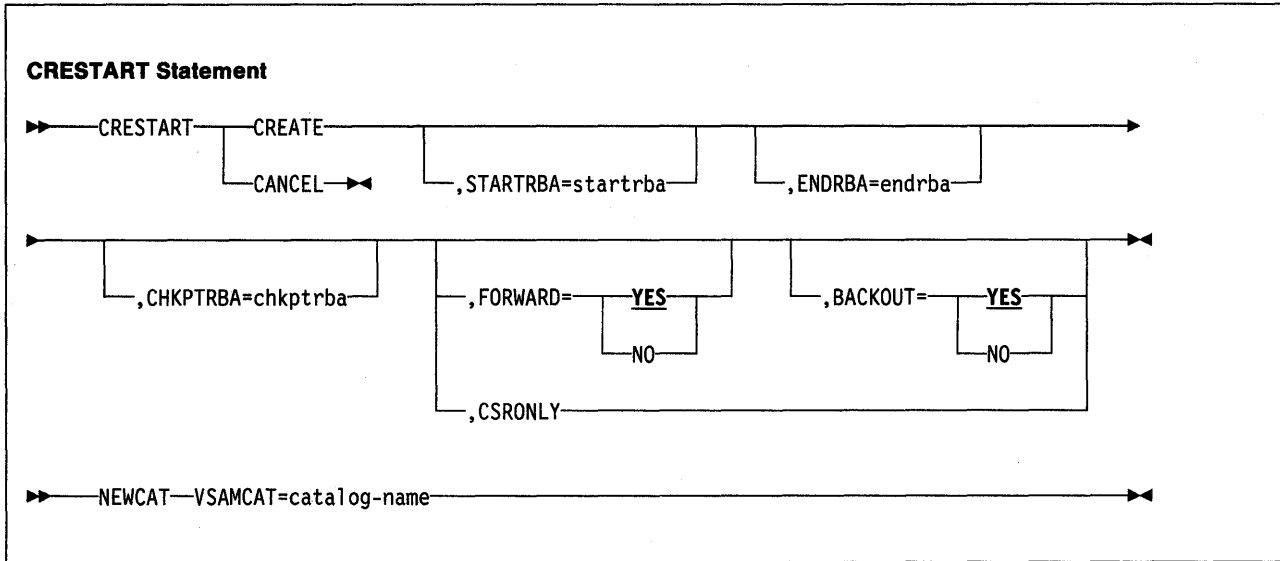


ARCHIVE Statement



SYSTEMDB Statement





Change Log Inventory Statements

The following statements tell what operations are to be performed. Their keywords and parameters describe the options to be used.

NEWLOG

Declares one of the following data sets:

- A VSAM data set that is available for use as an active log data set.
Use only the keywords DSNAME = , COPY1, COPY2, and PASSWORD = .
- An active log data set that is replacing one that encountered an I/O error.
Use only the keywords DSNAME = , COPY1, COPY2, STARTRBA = , ENDRBA = , and PASSWORD = .
- An archive log data set volume.
Use only the keywords DSNAME = , COPY1VOL = , COPY2VOL = , STARTRBA = , ENDRBA = , UNIT = , CATALOG = , and PASSWORD = .

DELETE

Deletes all information about the specified log data set or data set volume from the bootstrap data sets.

ARCHIVE

Gives a 1- to 8-character password for *all* archive data sets created after this operation. The password is added to the installation's MVS password data set every time a new archive log data set is created. The NOPASSWD parameter will remove the password protection for all archives created after this operation.

SYSTEMDB

Gives VSAM passwords for the data sets that support the following DB2 databases:

- DSNDB01 (the DB2 directory)
- DSNDB06 (the DB2 catalog)
- DSNDB04 (the default database).

CRESTART

Controls the next restart of DB2, either by creating a new conditional restart control record or by canceling the one currently active.

CAUTION:

This statement can override DB2's efforts to maintain data in a consistent state.

Do not use the statement without understanding "conditional restart," which is described in Section 5 of *System and Database Administration Guide*.

NEWCAT

Changes the VSAM catalog name in the BSDS.

Keyword and Parameter Descriptions**DSNAME = *dsname***

Names a log data set. *dsname* may be up to 44 characters long.

COPY1

Makes the data set an active log copy-1 data set.

COPY2

Makes the data set an active log copy-2 data set.

STARTRBA = *startrba*

On the NEWLOG statement, gives the log RBA (relative byte address within the log) of the beginning of the replacement active log data set or the archive log data set volume specified by DSNAME. *startrba* is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added. The RBA can be obtained from messages or by printing the log map.

On the CRESTART statement, *startrba* is the earliest RBA of the log to be used during restart. If you omit the option, DB2 determines the beginning of the log range.

ENDRBA = *endrba*

On the NEWLOG statement, gives the log RBA (relative byte address within the log) of the end of the replacement active log data set or the archive log data set volume specified by DSNAME. *endrba* is a hexadecimal number of up to 12 digits. If you use fewer than 12 digits, leading zeros are added.

On the CRESTART statement, *endrba* is the last RBA of the log to be used during restart, and the starting RBA of the next active log written after restart. Any log information in the bootstrap data set and the active logs, with an RBA greater than *endrba*, is discarded. If you omit ENDRBA, DB2 determines the end of the log range.

The value of ENDRBA must be a multiple of 4096. (The hexadecimal value must end in 000.) Also, the value must be greater than or equal to the value of STARTRBA. If STARTRBA and ENDRBA are equal, the next restart is a "cold start"; that is, no log records are processed during restart.

COPY1VOL = *vol-id*

vol-id is the volume serial of the copy-1 archive log data set named after DSNAME.

COPY2VOL = *vol-id*

vol-id is the volume serial of the copy-2 archive log data set named after DSNAME.

CHANGE LOG INVENT. (Utility)

UNIT = *unit-id*

unit-id is the device type of the archive log data set named after DSNAME.

CATALOG

Tells whether the archive log data set is cataloged.

NO

Tells that the archive log data set is not cataloged. All subsequent allocations of the data set are made using the unit and volume information specified on the statement. The **default** is **NO**.

YES

Tells that the archive log data set is cataloged. A flag is set in the BSDS indicating this, and all subsequent allocations of the data set are made using the catalog.

PASSWORD = *password*

Assigns a password to the data set. *password* is also stored in the BSDS and used in any subsequent access to archive log data sets.

TBLSPACE = *tablespace*

Gives the table space or index space qualifier for the data set to be assigned a VSAM password or from which the current VSAM password is to be removed. The qualifier is found in the data set name, which has this format:

catname.DSNDBx.dbname.tsname.I0001.An

where:

catname = VSAM catalog name or alias

x = C or D

dbname = database name

tsname = table space name

n = data set integer

Use the SYSTEMDB statement only if the database name (*dbname*) is DSNDB01 or DSNDB04. Then use the table space name (*tsname*) for *tablespace*.

CREATE

Creates a new conditional restart control record. When the new record is created, the previous control record becomes inactive.

CANCEL

Makes the currently active conditional restart control record inactive. The record remains in the BSDS as historical information.

No other keyword may be used with CANCEL.

CHKPTRBA = *chkptrba*

Is the log RBA (relative byte address within the log) of the start of the checkpoint record to be used during restart.

If you use STARTRBA or ENDRBA, and do not use CHKPTRBA, the CHANGE LOG INVENTORY utility selects the RBA of an appropriate checkpoint record. If you do use CHKPTRBA, you override the value selected by the utility. However, *chkptrba* must be in the range determined by *startrba* and *endrba* or their default values. If possible, don't use CHKPTRBA; let the utility determine the RBA of the checkpoint record.

CHKPTRBA=0 overrides any selection by the utility; at restart, DB2 attempts to use the most recent checkpoint record taken.

FORWARD =

Tells whether to use the “forward log recovery” phase of DB2 restart, which reads the log forward to recover any units of recovery that were in one of the following two states when DB2 was last stopped:

- Indoubt (had finished the first phase of commit, but not started the second phase)
- In-commit (had started but not finished the second phase of commit).

YES

Allows forward log recovery. The **default** is **FORWARD = YES**.

But, if you specify a “cold start” (by using the same value for STARTRBA and ENDRBA), no recovery processing is performed.

NO

Terminates forward log recovery before log records are processed.

BACKOUT =

Tells whether to use the “backward log recovery” phase of DB2 restart, which rolls back any units of recovery that were in one of the following two states when DB2 was last stopped:

- Inflight (did not complete the first phase of commit)
- In-abort (had started but not finished an abort).

YES

Allows backward log recovery. The **default** is **BACKOUT = YES**.

But, if you specify a “cold start” (by using the same value for STARTRBA and ENDRBA), no recovery processing is performed.

NO

Terminates backward log recovery before log records are processed.

CSRONLY

Performs only the first and second phases of restart processing (“log initialization” and “current status rebuild”). After these phases, the system status is displayed and restart terminates.

When DB2 is restarted with this option of effect, the conditional restart control record is not made inactive. To prevent the control record from remaining active, use the Change Log Inventory utility again with CRESTART CANCEL, or with CRESTART CREATE to create a new active control record.

NOPASSWD

Removes the archive password protection for all archives created after this operation. No other keyword may be used with NOPASSWD.

VSAMCAT = *catname*

Changes the VSAM catalog name entry in the BSDS. *catname* may be up to 8 characters long. The first character must be alphabetic, while the remaining characters may be alphameric.

SYSIN Stream Parsing

You may use more than one statement of each type. In each statement, separate the operation name (NEWLOG, DELETE, ARCHIVE, SYSTEMDB, or CRESTART) from the first parameter by one or more blanks. You may use parameters in any order; separate them by commas with no blanks. Do not split a parameter description across two SYSIN records.

Comments: A statement containing an asterisk in column 1 is considered a comment, and is ignored. However, it appears in the output listing. To include a comment or sequence number in a SYSIN record, separate it from the last comma by a blank. When a blank is encountered following a comma, the rest of the record is ignored.

Usage Notes

Utility Invocation: The following statement invoking the utility can be included only in a batch job:

```
//EXEC PGM=DSNJU003
```

Data Definition (DD) Statements: Change Log Inventory recognizes DD statements with the following ddnames:

JOB CAT

STEP CAT

Names the catalog in which the bootstrap data sets (BSDSs) are cataloged. The statement is optional. Typically, the high-level qualifier of the BSDS name will point to the ICF catalog that contains an entry for the BSDS.

SYSUT1

Is required, to name and allocate the bootstrap data set.

SYSUT2

Names and allocates a second copy of the bootstrap data set. The statement is required if you use dual BSDSs.

SYS PRINT

Is required, to name a data set for print output.

SYS IN

Is required, to name the input data set for statements.

Deleting Log Data Sets with Errors: If an active log data set has encountered an I/O error, you must:

1. Be sure that the data is saved. If you have been using dual active log data sets, DB2 can use the other active log. Otherwise, you may:
 - a. Check to see if the data set has been offloaded. For example, check the list of archive log data sets to see if one has the same RBA range as the active log data set.
 - b. If the data set has not been offloaded, copy the data to a new data set. Then, update the bootstrap data set; to do that:
 - 1) Use DELETE to remove information about the bad data set.
 - 2) Use NEWLOG to name the new data set as the new active log.

The DELETE and NEWLOG operations can be performed by the same job step (the DELETE statement precedes the NEWLOG statement in the SYSIN input data set).

2. Delete the bad data set, using VSAM Access Method Services.

Changing the Password on an Archive Log Data Set: When you change the password on an archive log data set, you must change the entry in the bootstrap data set as follows:

1. Use DELETE to remove the entry for the data set whose password changed.
2. Use NEWLOG to name the same data set as a new archive log, giving the new password.

No such action is needed when you change the password on an *active* log data set, because its password is not recorded in the bootstrap data set.

Conditional Restart Control Record: An existing conditional restart control record governs any START DB2 operation until one of these events occurs:

- A restart operation completes.
- A CRESTART CANCEL statement is issued.
- A new conditional restart control record is created.

Examples

Example 1: In the following example, the first statement adds a new archive log data set. The second statement establishes a new password for subsequent archive data sets.

```
NEWLOG DSNAME=DSNREPAL.A0001187,COPY1VOL=DSNV04,UNIT=SYSDA,
STARTRBA=3A190000,ENDRBA=3A1F0000,CATALOG=NO,PASSWORD=SYSNKZX
ARCHIVE PASSWORD=SYSNLZX
```

Example 2: The following statement deletes a data set.

```
DELETE DSNAME=DSNREPAL.A0001187,COPY1VOL=DSNV04
```

Example 3: The following statement creates a new conditional restart control record, specifying no backward log recovery and "log truncation" (a new relative byte address for the end of the log).

```
CRESTART CREATE,BACKOUT=NO,ENDRBA=0000000010000
```

CHECK (Utility)

The CHECK utility tests whether indexes are consistent with the data they index, and issues warning messages when it finds an inconsistency. CHECK also checks table spaces for violations of referential constraints, and reports information about violations detected. CHECK optionally deletes rows in violation of referential constraints, and copies them to an exception table.

Environment

See “Chapter 3. Running DB2 Utilities” on page 151, for an explanation of ways to invoke DB2 utilities.

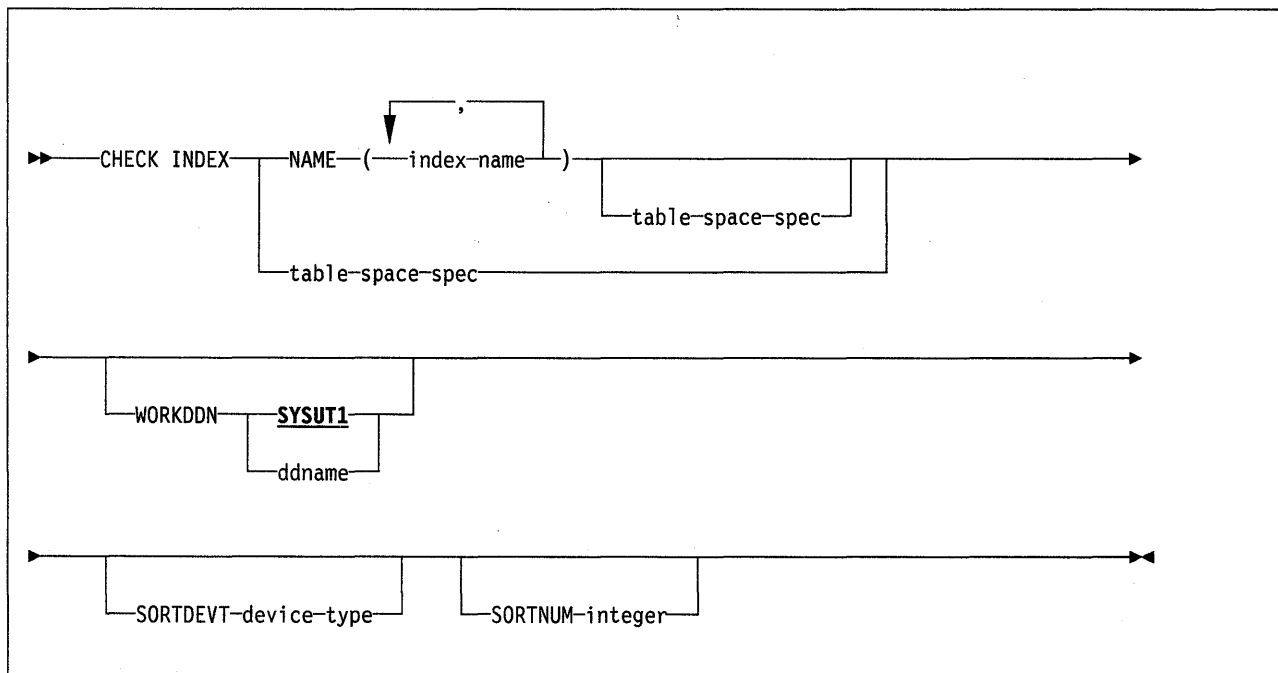
Authorization

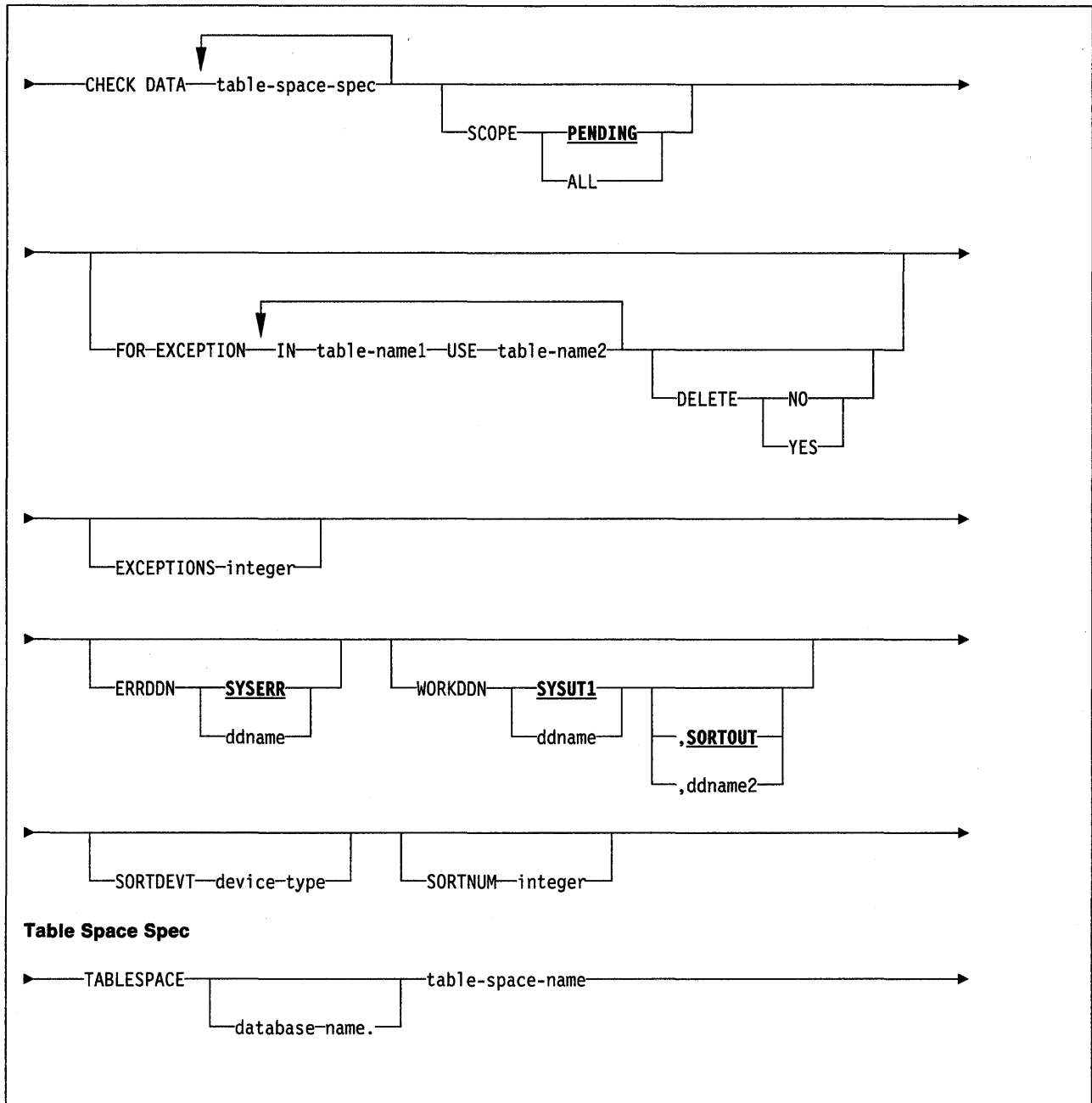
To execute this utility, the privilege set defined below must include the CHECK privilege for the database containing the named table space. The CHECK privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following privileges:

- SYSADM authority
- DBADM, DBCTRL, or DBMAINT authority for the database.

To execute the CHECK INDEX utility on a table space in database in database DSNDB01 or DSNDB06, the privilege set must include the SYSADM or SYSOPR authorities defined when DB2 was installed. To execute all other utilities against a table space in database DSNDB01 or DSNDB06, the privilege set must include the SYSADM authority defined when DB2 was installed.

The authorization required to invoke a utility job step is discussed under “Authorization” in the description of each utility.





Keyword and Parameter Descriptions

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see "Utility Statement Parsing" on page 15.

CHECK INDEX

INDEX

Indicates that you are checking for index consistency.

CHECK (Utility)

NAME (*index-name*)

Names the indexes that are to be checked. All indexes must belong to tables in the same table space. If you omit the NAME option, you must use the TABLESPACE option, and then CHECK checks all indexes on all tables in the table space you name.

index-name

Is the name of an index, in the form *creator-id.name*. If you omit the qualifier *creator-id.*, the userid for the utility job is used. If you use a list of names, separate items in the list by commas. Parentheses are required around a name or list of names.

TABLESPACE *dbname.tsname*

Names the table space the indexes belong to. If you omit the NAME option and use TABLESPACE, CHECK checks all indexes on all tables in the table space you name. If you use NAME with TABLESPACE, all indexes you name must be on tables in the table space you name.

dbname is the name of the database and is optional. The default is **DSNDB04**.

tsname is the name of the table space.

WORKDDN *ddname*

Names a DD statement for a temporary work file.

ddname is the DD name. The default is **SYSUT1**.

SORTDEVT *device-type*

Names the device type for temporary data sets to be dynamically allocated by DFSORT. It can be any device type acceptable to the DYNALLOC parameter of the SORT or OPTION control statement for DFSORT, as described in *DFSORT Application Programming: Guide*.

device-type is the device type. If you omit SORTDEVT and a sort is required, you must provide the DD statements that the sort program needs for the temporary data sets.

SORTNUM *integer*

Tells the number of temporary data sets to be dynamically allocated by the sort program.

integer is the number of temporary data sets.

If you omit SORTDEVT, SORTNUM is ignored.

If you use SORTDEVT and omit SORTNUM, no value will be passed to DFSORT; it will be allowed to take its own default.

CHECK DATA

DATA

Indicates that you are checking referential constraints.

SCOPE

Specifies which rows in the table space are to be checked.

PENDING

Indicates that only those rows that need to be checked in table spaces, partitions, or tables that are in "check pending" status are to be checked.

If you specify this option for a table space that is *not* in check pending status, the table space will be ignored.

ALL

Indicates that all dependent tables in the specified table spaces are to be checked.

FOR EXCEPTION

Indicates that any row in violation of referential constraints is copied to an exception table.

IN *table1*

Specifies the table (in the table space specified on the TABLESPACE keyword) from which rows are to be copied.

table1 is the name of the table.

USE *table2*

Specifies the exception table into which error rows are to be copied.

table2 is the name of the exception table.

DELETE

Indicates whether or not rows in violation of referential constraints are deleted from the table space after they are copied into an exception table. You can only use this option if you have used the FOR EXCEPTION keyword.

NO

Indicates that error rows remain in the table space.

YES

Indicates that error rows are deleted from the table space.

EXCEPTIONS *integer*

Specifies the maximum number of exceptions reported. CHECK terminates when it reaches the number of exceptions specified. If you specify zero, all exceptions are reported.

Note: Only records containing *primary* referential integrity errors are applied toward the discard limit. There is no limit on the number of records containing secondary errors.

The **default** is zero.

ERRDDN *ddname*

Names a DD statement for an error processing data set.

ddname is the DD name. The **default** is **SYSERR**.

WORKDDN (*ddname1,ddname2*)

Names the DD statements for the temporary work file for sort input and the temporary work file for sort output. A temporary work file for sort output is *required*.

ddname1 is the DD name of the temporary work file for sort input. The **default** is **SYSUT1**.

ddname2 is the DD name of the temporary work file for sort output. The **default** is **SORTOUT**.

SORTDEVT *device-type*

Names the device type for temporary data sets to be dynamically allocated by DFSORT. It can be any device type acceptable to the DYNALLOC parameter of the SORT or OPTION control statement for DFSORT, as described in *DFSORT Application Programming: Guide*.

CHECK (Utility)

device-type is the device type. If you omit SORTDEVT and a sort is required, you must provide the DD statements that the sort program needs for the temporary data sets.

SORTNUM *integer*

Tells the number of temporary data sets to be dynamically allocated by the sort program.

integer is the number of temporary data sets.

If you omit SORTDEVT, SORTNUM is ignored.

If you use SORTDEVT and omit SORTNUM, no value will be passed to DFSORT; it will be allowed to take its own default.

Output

Output from the CHECK utility consists entirely of messages. See Section 3 of *Messages and Codes* for more information about these messages.

If FOR EXCEPTION is specified, output consists of error rows copied to exception tables.

If DELETE YES is specified, output consists of error rows deleted from dependent tables; in addition, all descendent rows of error rows are copied to exception tables and deleted from their respective source tables.

Usage Notes

The work data set for CHECK INDEX: A single sequential data set, described by the DD statement named in the WORKDDN option, is needed during execution of CHECK with the INDEX option.

To find the approximate size of the WORKDDN data set, in bytes:

1. For each table, multiply the number of records in the table by the number of nonclustering indexes defined on the table.
2. Add the products obtained in step 1.
3. Multiply that sum by the largest key length plus 6.

Another way to estimate the size of the WORKDDN data set is to obtain the high-used relative byte address (RBA) for each index from a VSAM catalog listing. Then sum the RBAs.

The work data sets for CHECK DATA: Three sequential data sets, described by the DD statements named in the WORKDDN and ERRDDN options, are needed during execution of CHECK with the DATA option.

To find the approximate size of the WORKDDN data set, in bytes:

1. Add 12 to the length of the longest foreign key.
2. Multiply the sum by the number of keys checked.

Create the ERRDDN data set so that it is large enough to accommodate 1 error entry (length=60 bytes) per defect detected by CHECK.

Restarting: You cannot restart a CHECK utility job. If necessary, terminate the job by using -TERM UTILITY, and run the job again.

When you terminate CHECK DATA, table spaces remain in the check pending status in which they were when the utility was -TERMed. The CHECKDAT phase places the table space in the check pending status when an error is detected; at the end of the phase, the check pending status is reset if no errors were detected. The REPORTCK phase resets the check pending status if the DELETE YES option was specified.

Phases of execution: Though you cannot restart CHECK, one of the following phases may be identified if the job terminates.

For CHECK with the INDEX option, the utility signals successful completion by issuing summary messages at the end of phase CHECKIDX. The phases for CHECK INDEX are:

- UTILINIT: initialization and setup
- UNLOAD: unloading of index entries
- SORT: sorting of unloaded index entries
- CHECKIDX: scanning of data to validate index entries
- UTILTERM: cleanup

The phases for CHECK DATA are:

- UTILINIT: initialization.
- SCANTAB: extract foreign keys. Use foreign key index if it exists, else scan table.
- SORT: sort foreign keys if not extracted from foreign key index.
- CHECKDAT: look in primary indexes for foreign key parents, and issue messages to report errors detected
- REPORTCK: copy error rows into exception tables, and delete them from source table if DELETE YES is specified.
- UTILTERM: cleanup.

Exception Tables: See Chapter 5 of *SQL Reference* for information about creating exception tables.

Examples

Example 1: Check the project-number index (DSN8210.XPROJ1) on the sample project table.

```
CHECK INDEX NAME (DSN8210.XPROJ1)
SORTDEVT SYSDA
```

Example 2: Check all indexes on the employee-table table space (DSN8S21E).

```
CHECK INDEX TABLESPACE DSN8S21E
SORTDEVT 3330
```

Example 3: Check the indexes DSN8210.XEMPRAC1 and DSN8210.XEMPRAC2 on the employee-to-project-activity sample table.

```
CHECK INDEX NAME (DSN8210.XEMPRAC1, DSN8210.XEMPRAC2)
```

Example 4: Check and correct all constraint violations in table spaces DSN8D21A.N8S21D and DSN8D21A.DSN8S21E.

CHECK (Utility)

```
CHECK DATA TABLESPACE DSN8D21A.DSN8S21D
TABLESPACE DSN8D21A.DSN8S21E
FOR EXCEPTION IN DSN8210.DEPT USE DSN8210.EDEPT
                IN DSN8210.EMP USE DSN8210.EEMP
                IN DSN8210.PROJ USE DSN8210.EPROJ
                IN DSN8210.PROJACT USE DSN8210.EPROJACT
                IN DSN8210.EMPPROJACT USE DSN8210.EEPA
DELETE YES
```

COPY (Utility)

The image copy utility (COPY) creates an image copy of a table space or a data set within a table space. There are two types of image copies:

1. A **full image copy** is a copy of all pages in a table space or data set.
2. An **incremental image copy** is a copy only of pages that have been modified since the last use of the COPY utility.

Environment

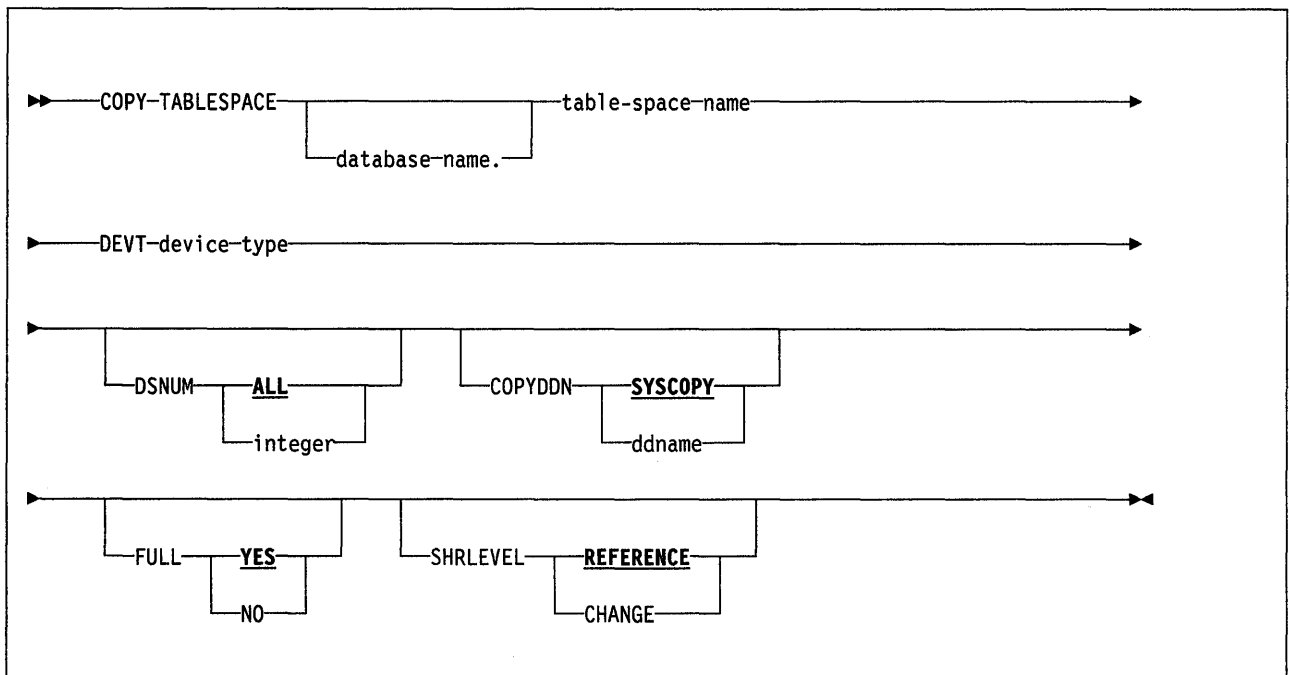
See “Chapter 3. Running DB2 Utilities” on page 151, for an explanation of ways to invoke DB2 utilities.

Authorization

To execute this utility, the privilege set defined below must include the IMAGCOPY privilege for the database containing the named table space. The IMAGCOPY privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following privileges:

- SYSADM authority
- DBADM, DBCTRL, DBMAINT authorities for a database.

If the utility is being executed on a table space in database DSNDB01 or DSNDB06, the privilege set must include either the SYSADM or SYSOPR authority defined when DB2 was installed.



Keyword and Parameter Descriptions

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see "Utility Statement Parsing" on page 15.

TABLESPACE

Names the table space (and, optionally, the data base it belongs to) that is to be copied.

database-name

Is the name of the database the table space belongs to. The **default** is **DSNDB04**.

tablespace-name

Is the name of the table space to be copied.

DEVT

Names the device type for the image copy data set. *device-type* must match the COPYDDN *ddname* on the DD statement specification for UNIT.

If you use tape output, do not put an incremental image copy on the same tape with either the full image copy or another incremental image copy of the same table space; to do so would cause a later RECOVER operation to fail. If you attempt it while using COPY, error messages are issued and COPY terminates.

The following identifiers are optional.

DSNUM

Identifies a partition or data set, within the table space, that is to be copied; or it copies the entire table space.

ALL

Copies the entire table space. The **default** is **ALL**.

integer

Is the number of a partition or data set to be copied.

For a partitioned table space, the integer is its partition number.

For a nonpartitioned table space, the total number of data sets can increase dynamically. Find the data set integer at the end of the data set name in the VSAM catalog. The data set name has this format:

catname.DSNDB*x*.*dbname*.*tsname*.I0001.*An*

where:

catname = VSAM catalog name or alias
x = C or D
dbname = database name
tsname = tablespace name
n = data set integer

Note: If image copies are taken by data set (rather than by table space), then RECOVER or MERGECOPY must use the copies by data set.

COPYDDN *ddname*

Names the DD statement for the output data set for the image copy. *ddname* is the DD name. The **default** is **SYSCOPY**.

You cannot have duplicate image copy data sets. If the DD statement identifies a non-cataloged data set with the same name, volume serial, and file sequence

number as one already recorded in SYSIBM.SYSCOPY, a message is issued and no copy is made. If it identifies a cataloged data set with only the same name, no copy is made. For cataloged image copy data sets, CATLG should be specified for the normal termination disposition in the DD statement. For example, DISP=(,CATLG). The DSVOLSER field of the SYSIBM.SYSCOPY entry will be blank.

FULL

Makes either a full or an incremental image copy.

YES

Makes a full image copy. The default is **YES**.

Making a full image copy resets the “copy pending” status for the table space (or from a single data set, if you used DSNUM).

NO

Makes only an incremental image copy. Only changes since the last image copy are copied.

After a successful LOAD or REORG operation, you cannot use FULL NO until a full image copy has been made.

SHRLEVEL

Tells whether other programs that access the table space while COPY is running must use read-only access, or may change the table space.

REFERENCE

Allows read-only access by other programs if LOCKRULE in the SYSIBM.SYSTABLESPACE catalog table is either ANY(A) or PAGE(P). The default is **REFERENCE**.

CHANGE

Allows other programs to change the table space, *but only if* LOCKRULE in the SYSIBM.SYSTABLESPACE catalog table is either ANY (A) or PAGE (P).

With SHRLEVEL CHANGE, the utility will take longer to execute, and the resulting copy will take longer to process by RECOVER.

Output

Output from the COPY utility consists of:

- A sequential non-VSAM data set (described by the DD statement named in the COPYDDN option) containing the image copy.

To find the approximate size of the image copy data set, in bytes:

1. Find the *high allocated page number*, either from the NACTIVE column of SYSIBM.SYSTABLESPACE after running the RUNSTATS utility, or from information in the VSAM catalog data set.
 2. Multiply the high allocated page number by the page size.
- Rows in the SYSIBM.SYSCOPY catalog table that describe the image copy data sets available to the RECOVER utility. It is your installation’s responsibility to ensure that these data sets are available if the RECOVER utility requests them.
 - Resetting the copy pending status, if the copy was a full image copy.

Usage Notes

Pending Restrictions: You cannot copy a table space that is in the check or recovery pending status. See “CHECK (Utility)” on page 178 for information about resetting the check pending status; see “RECOVER (Utility)” on page 257 for information about resetting the recovery pending status.

Parallel Image Copy: If a table space has multiple underlying data sets, you can copy several or all of the data sets independently in separate parallel jobs. This can reduce the time it takes to create an image copy of the total table space.

If the table space is not partitioned, specify SHRLEVEL CHANGE in each parallel job.

BLKSIZE parameter for output: You may specify a block size for the output by using the BLKSIZE parameter on the DD statement for the output data set. Valid block sizes are 4096, 8192, and 16384 bytes; the default is 16384 bytes. The most efficient block size for the 3380 is 8192 bytes. You can increase the buffering with the BUFNO parameter; for example, you could specify BUFNO=20.

Copying Segmented Table Spaces: COPY distinguishes between segmented and unsegmented table spaces. If the table space you specify is segmented, COPY locates empty and unformatted data pages in the table space, and does not copy them.

Image Copy and Reorganization: RECOVER does not use image copies taken before a reorganization to recover data changed after that reorganization. Because of this, REORG makes an entry in the SYSIBM.SYSCOPY catalog table to show that all image copies taken before the reorganization are invalid for recovery to currency. However, the image copies are still valid for recovery to a prior point in time.

Multiple image copies are recommended after a LOAD or REORG operation, with the LOG option set to NO. This is recommended so that fallback recovery will be possible.

Incremental Copy and LOAD or REORG: If a LOAD or REORG operation is successful, and if the LOG option was set to NO, the copy pending status is set to prevent you from making an incremental image copy instead of a full image copy.

Terminating a Copy: If -TERM UTILITY is used to terminate a full or incremental copy, the table space may be placed in the copy pending status. This will require that you make a full image copy before any further updates to the data in the table space.

Restarting an Incremental Image Copy: If -TERM UTILITY is used, the output data set is deleted and the copy pending status is set. In that case, the job cannot be restarted. Make a full image copy before making any more incremental image copies, so the object can be completely recovered.

If -TERM UTILITY is *not* used, the job can be restarted at the last commit point. It continues where it left off by positioning to the end of the output data set. Do not restart the job at the beginning of the phase. For instructions on restarting a utility job, see “Chapter 3. Running DB2 Utilities” on page 151.

Phases of execution: The COPY utility operates in these phases:

1. UTILINIT: initialization and setup
2. COPY: copying
3. UTILTERM: cleanup

Generation Data Sets: COPY allows you to use generation data sets.

When the generation data group is defined, make the limit number of generation data sets equal to the number of copies to keep. Use NOEMPTY to avoid deleting all the data sets from the catalog when the limit is reached.

If the high-level qualifier of the data set name is not the catalog name or its alias, then include a JOBCAT or STEPCAT DD statement in the COPY job, and the eventual RECOVER job, to tell what catalog the data set is in.

Examples

Example 1: Make a full image copy of the DSN8S21E table space in database DSN8D21A.

```
COPY TABLESPACE DSN8D21A.DSN8S21E
  DEVT SYSDA
```

Example 2: Make an incremental image copy of the DSN8S21D table space in database DSN8D21A, allowing update activity to occur during the copy process.

```
COPY TABLESPACE DSN8D21A.DSN8S21D
  FULL NO
  SHRLEVEL CHANGE
  DEVT SYSDA
```

DIAGNOSE (Utility)

The DIAGNOSE utility generates information useful in diagnosing problems. It is intended for use after direction by your IBM Support Center.

Environment

See "Chapter 3. Running DB2 Utilities" on page 151, for a description of ways to invoke DB2 utilities.

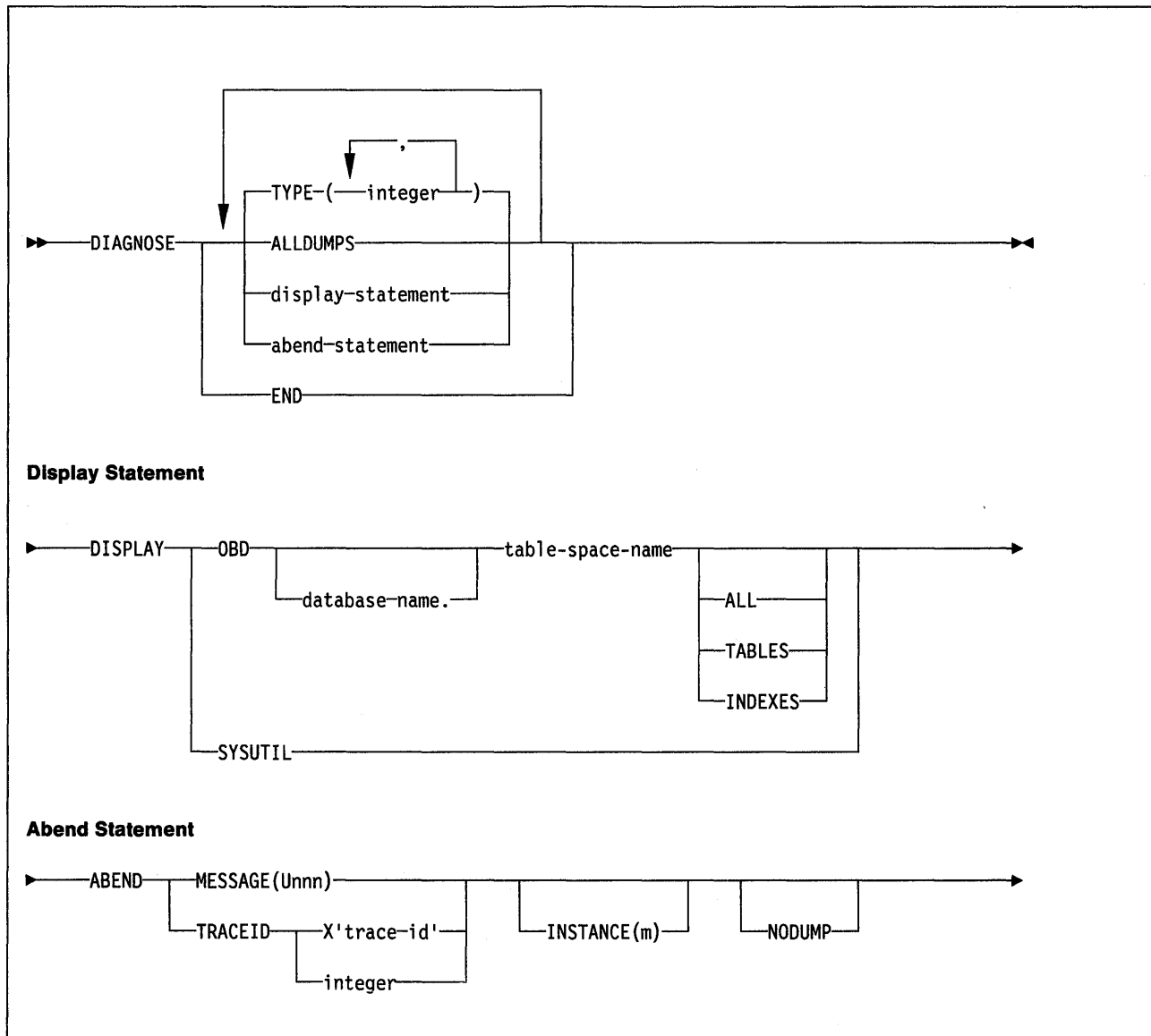
Authorization

To execute this utility, the privilege set defined below must include the REPAIR privilege for the database containing the named table space. The REPAIR privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following privileges:

- SYSADM authority
- DBADM or DBCTRL authorities for the database.

If the utility is being executed on a table space in database DSNDB01 or DSNDB06, the privilege set must include the SYSADM authority defined when DB2 was installed.

The authorization required to invoke a utility job step is discussed under "Authorization" in the description of each utility.



Keyword and Parameter Descriptions

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see "Utility Statement Parsing" on page 15.

TYPE

Specifies one or more types of diagnose you wish to perform. The maximum number of types is 32.

END

Ends DIAGNOSE processing.

ALLDUMPS

Indicates that a dump is taken for any utility abend. This parameter overrides any elsewhere specified dump suppression.

DISPLAY

OBD

Specifies that the OBD of the specified table space is to be formatted to SYSPRINT.

ALL

Specifies that all OBDs of the specified table space are to be formatted to SYSPRINT. The OBD of any relationship associated with the table space is also formatted.

TABLES

Specifies that the OBDs of all tables in the specified table spaces are to be formatted to SYSPRINT.

INDEXES

Specifies that the OBDs of all indexes in the specified table spaces are to be formatted to SYSPRINT.

SYSUTIL

Specifies that every SYSUTIL record is to be formatted to SYSPRINT.

ABEND

Forces an abend during utility execution.

MESSAGE(*Unnn*)

Specifies that an abend is to occur when the DSN*Unnn* message is issued.

TRACEID *trace-id*

Specifies that an abend is to occur when the specified *trace-id* is encountered.

trace-id must be a trace-id associated with the utility trace (RMID 21), and can be specified in decimal or hexadecimal.

INSTANCE(*m*)

Specifies that an abend is to occur when the message specified on the MESSAGE option or the trace-id specified on the TRACE option has been encountered *m* times.

The **default** is one time.

NODUMP

Suppresses the dump generated by an abend of DIAGNOSE.

Examples

Example 1: Force a dump for any utility abend that occurs during the execution of the COPY utility.

```
DIAGNOSE
ALLDUMPS
COPY TABLESPACE DSNDB06.SYSDBASE
DEVT SYSDA
DIAGNOSE END
```

Example 2: Abend the LOAD utility the fifth time message DSNU311 is issued. Do not generate a dump.

```
DIAGNOSE
ABEND MESSAGE (U311)
  INSTANCE (5)
  NODUMP
LOAD DATA INDDN INPUT
  RESUME NO
  INTO TABLE TABLE1
  (NAME POSITION(1)
  CHAR(20))
```

Example 3: Display all rows in the SYSUTIL table.

```
DIAGNOSE
DISPLAY SYSUTIL
```

DSN1CHKR (Service Aid)

The DSN1CHKR service aid verifies the integrity of DB2 directory and catalog table spaces. DSN1CHKR scans the specified table space for broken links, hash chains, and orphans (records that are not part of any link or chain).

Environment

Run the DSN1CHKR program as an MVS job.

You can run DSN1CHKR on a table space while DB2 is operational; however, data manipulation operations on the table space should be prevented while the service aid is running.

It is recommended that you copy the table space to a temporary data set and run DSN1CHKR on the copy. This frees the actual table space for DB2 use. You can use DSN1COPY to do this. DSN1CHKR requires a VSAM cluster as input; it cannot check a physical sequential data set. "Example 1" on page 197 gives a sample job to create a temporary cluster; run DSN1COPY to copy a table space into it.

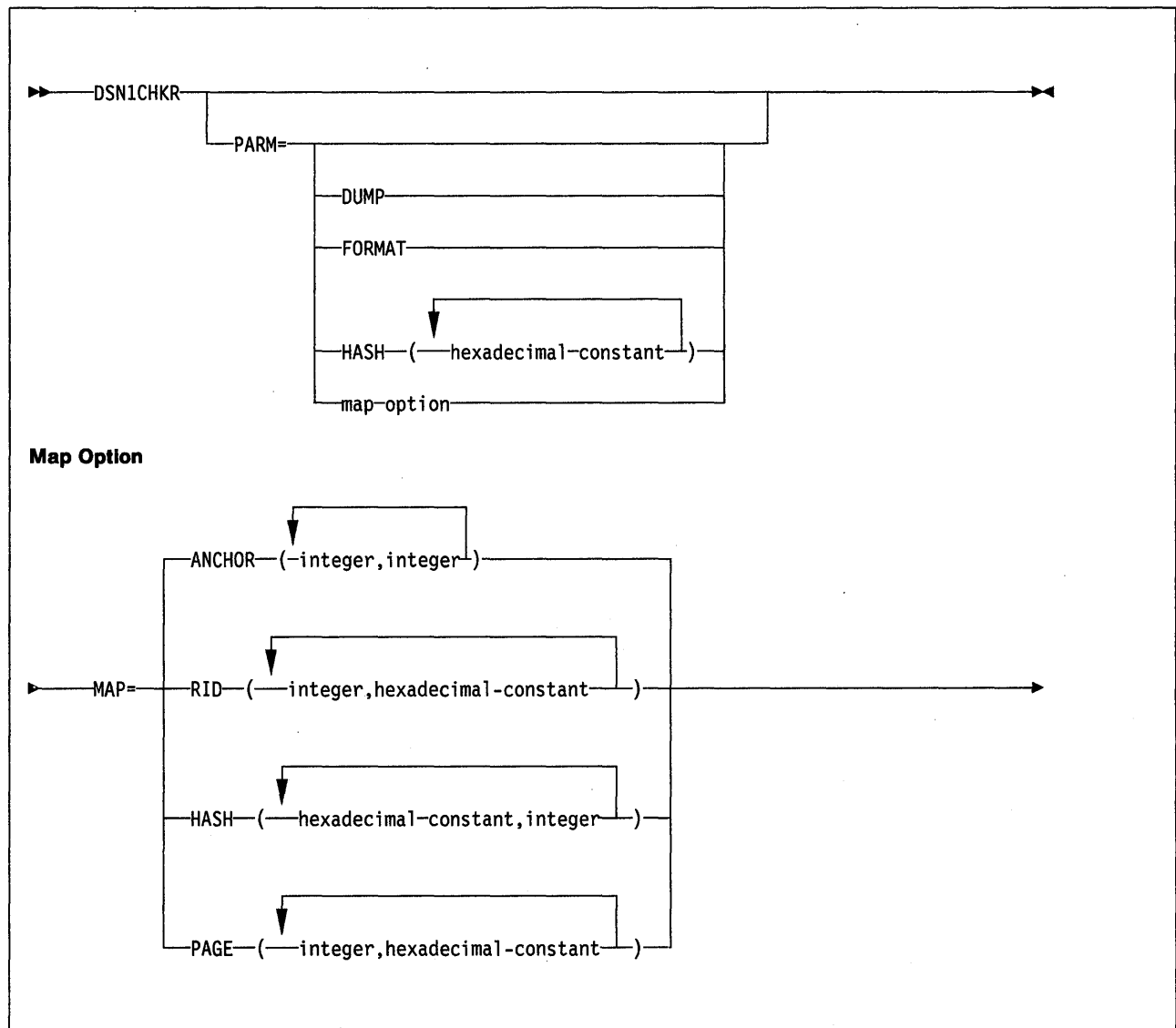
If you choose not to copy the table space to a temporary data set, you must ensure that no database operations are performed while DSN1CHKR runs by issuing the -STOP DATABASE command for the database and table space you want to check.

Note: When you run DSN1COPY, use the CHECK option to examine the table space for page integrity errors. Although DSN1CHKR does check for these errors, running DSN1COPY with CHECK will prevent an unnecessary invocation of DSN1CHKR.

DSN1COPY cannot be run on table spaces DSNDB06.SYSCOPY, DSNDB06.SYSBPAUT, and DSNDB06.SYSUSER.

Authorization

None is required. However, if any of the data sets are RACF protected, the primary authorization ID designated by the process must have the necessary RACF authority. Also, if the data sets are password protected, the primary authorization ID designated by the process must have the appropriate VSAM password to execute this service aid.



Keyword and Parameter Descriptions

The following parameters are optional. If you do not specify any parameters, DSN1CHKR scans all table space pages for broken links and orphans, and only prints the appropriate diagnostic messages.

DUMP

Specifies that printed table space pages, if any, are in dump format. If you specify DUMP, you cannot specify the FORMAT parameter.

FORMAT

Specifies that printed table space pages, if any, are formatted on output. If you specify FORMAT, you cannot specify the DUMP parameter.

HASH = (hexadecimal-constant)

Specifies a hash value for a hexadecimal database identifier (*dbid*) in table space DBD01. DSN1CHKR returns hash values for each *dbid* in page and anchor point offset form.

The maximum number of *dbids* is 10.

MAP =

Identifies a record whose forward, backward or child pointer is followed. DSN1CHKR prints each record as it follows the pointer. Use this parameter only after you have determined which chain is broken (you can determine this by running DSN1CHKR without any parameters, or with FORMAT or DUMP only).

The options for this parameter help DSN1CHKR locate the record whose pointer it follows. Each option must point to the beginning of the 6-byte prefix area of a valid record, or to the beginning of the hash anchor. If the value you specify does not point to one of these, DSN1CHKR issues an error message and continues with the next pair of values.

ANCHOR(*integer,integer*)

Names the anchor point that DSN1CHKR maps.

id identifies the starting page and anchor point in the form PPPPPAA, where PPPPP is the page number and AA is the anchor point number. *integer* determines which pointer to follow while mapping. 0 specifies the forward pointer; 4 specifies the backward pointer.

The maximum number of pairs is 5.

RID(*integer,hexadecimal-constant*)

Identifies the record or hash anchor from which DSN1CHKR will start mapping.

integer. is the page and record, in the form PPPPPRR, where PPPPP is the page number and RR is the record number.

hexadecimal-constant specifies the hexadecimal displacement from the beginning of the record to the pointer in the record from which mapping starts.

The maximum number of pairs is 5.

HASH(*hexadecimal constant,integer*)

Names the *dbid* that DSN1CHKR hashes and maps for table space DBD01.

integer determines which pointer to follow while mapping. 0 specifies the forward pointer; 4 specifies the backward pointer.

The maximum number of pairs is 5.

PAGE(*integer,hexadecimal-constant*)

integer specifies the page number on which the record or hash anchor is located. *hexadecimal-constant* specifies the offset to the pointer from the beginning of the page.

When you use the PAGE option, DSN1CHKR follows the forward pointer while mapping. If a forward pointer does not exist, DSN1CHKR will stop mapping after the first record.

The maximum number of pairs is 5.

Examples

Example 1: In the following example, DSN1CHKR is run on a temporary data set.

STEP1 allocates a temporary data set. STEP2 copies the target table space into the temporary data set with DSN1COPY. The CHECK option is used to check the table space for page integrity errors. Once DSN1COPY with the CHECK option has ensured that no such errors exist, STEP3 runs DSN1CHKR on the temporary data set.

DSN1CHKR prints the chains beginning at the pointers specified on the RID option of the MAP parameter. The first pointer is located on page 2, at an offset of 6 bytes from record 1. The second pointer is located on page B, at an offset of 6 bytes from record 1.

DSN1CHKR (Service Aid)

```

//YOUR JOBCARD
//*
//JOB CAT DD DSN=DSN1CAT1.USER.CATALOG,DISP=SHR
//STEP1 EXEC PGM=IDCAMS
//*****
//* ALLOCATE A TEMPORARY DATA SET FOR SYSDBASE *
//*****
//SYS PRINT DD SYSOUT=A
//SYS DUMP DD SYSOUT=A
//SYS IN DD *
DELETE -
 (TESTCAT.DSNDBC.TEMPDB.TMPDBASE.I0001.A001) -
 CATALOG(DSN1CAT)
DEFINE CLUSTER -
 ( NAME(TESTCAT.DSNDBC.TEMPDB.TMPDBASE.I0001.A001) -
 NONINDEXED -
 REUSE -
 CONTROLINTERVALSIZE(4096) -
 VOLUMES(XTRA02) -
 RECORDS(783 783) -
 RECORDSIZE(4089 4089) -
 SHAREOPTIONS(3 3) )
DATA -
 ( NAME(TESTCAT.DSNDBC.TEMPDB.TMPDBASE.I0001.A001) -
 CATALOG(DSN1CAT)
/*
//STEP2 EXEC PGM=DSN1COPY,PARM=(CHECK)
//*****
//* CHECK SYSDBASE AND RUN DSN1COPY *
//*****
//STEPLIB DD DSN=DSN210.DSNLOAD,DISP=SHR
//SYS PRINT DD SYSOUT=A
//SYSUT1 DD DSN=DSN1CAT.DSNDBC.DSNDB06.SYSDBASE.I0001.A001,DISP=SHR
//SYSUT2 DD DSN=TESTCAT.DSNDBC.TEMPDB.TMPDBASE.I0001.A001,DISP=SHR
/*
//STEP3 EXEC PGM=DSN1CHKR,PARM='MAP=RID(00000201,06,00000B01,06)',
// COND=(4,LT)
//*****
//* CHECK LINKS OF SYSDBASE *
//*****
//STEPLIB DD DSN=DSN210.DSNLOAD,DISP=SHR
//SYS PRINT DD SYSOUT=A
//SYSUT1 DD DSN=TESTCAT.DSNDBC.TEMPDB.TMPDBASE.I0001.A001,DISP=SHR
/*

```

Figure 20. Sample JCLs for running DSN1CHKR on a temporary data set.

Example 2: In the following example, DSN1CHKR is run on the actual table space.

STEP1 stops database DSNDB06 with the -STOP DATABASE command. STEP2 runs DSN1CHKR on the target table space; its output is identical to the output in Example 1. STEP3 restarts the database with the -START DATABASE command.

```

//YOUR JOBCARD
//*
//STEP1 EXEC PGM=IKJEFT01,DYNAMNBR=20
//*****
//*                               EXAMPLE 2                               *
//*                               *                                       *
//*                               STOP DSND06.SYSDBASE                       *
//*****
//STEPLIB DD DSN=DSN210.DSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSTSIN DD *
DSN SYSTEM(SSTR)
  -STOP DB(DSND06) SPACENAM(SYSDBASE)
END
/*
//STEP2 EXEC PGM=DSN1CHKR,PARM='MAP=RID(00000201,06,00000B01,06)',
//      COND=(4,LT)
//*****
//*                               CHECK LINKS OF SYSDBASE                       *
//*****
//STEPLIB DD DSN=DSN210.DSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=DSNCAT.DSNDBD.DSND06.SYSDBASE.I0001.A001,DISP=SHR
/*
//STEP3 EXEC PGM=IKJEFT01,DYNAMNBR=20
//*****
//*                               RESTART DSND06.SYSDBASE                       *
//*****
//STEPLIB DD DSN=DSN210.DSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSTSIN DD *
DSN SYSTEM(SSTR)
  -START DB(DSND06) SPACENAM(SYSDBASE)
END
/*

```

Figure 21. Sample JCLs for running DSN1CHKR on a stopped table space.

DSN1COPY (Service Aid)

The DSN1COPY service aid allows you to copy DB2 VSAM data sets that contain table spaces or index spaces. You can also copy:

- VSAM data sets to sequential data sets
- Sequential data sets to VSAM data sets
- VSAM data sets to other VSAM data sets
- Sequential data sets to other sequential data sets.

Using DSN1COPY, you can also print hexadecimal dumps of DB2 data sets and databases; perform validity checking on data or index pages; and perform OBID translation to enable moving data sets between different systems.

Notes:

1. You should not use DSN1COPY on DB2 recovery log data sets, because the CHECK, RESET, and 32K options can cause incorrect results.
2. If you require only a printed hexadecimal dump of a data set, DSN1PRNT may better suit your purpose than DSN1COPY.

Environment

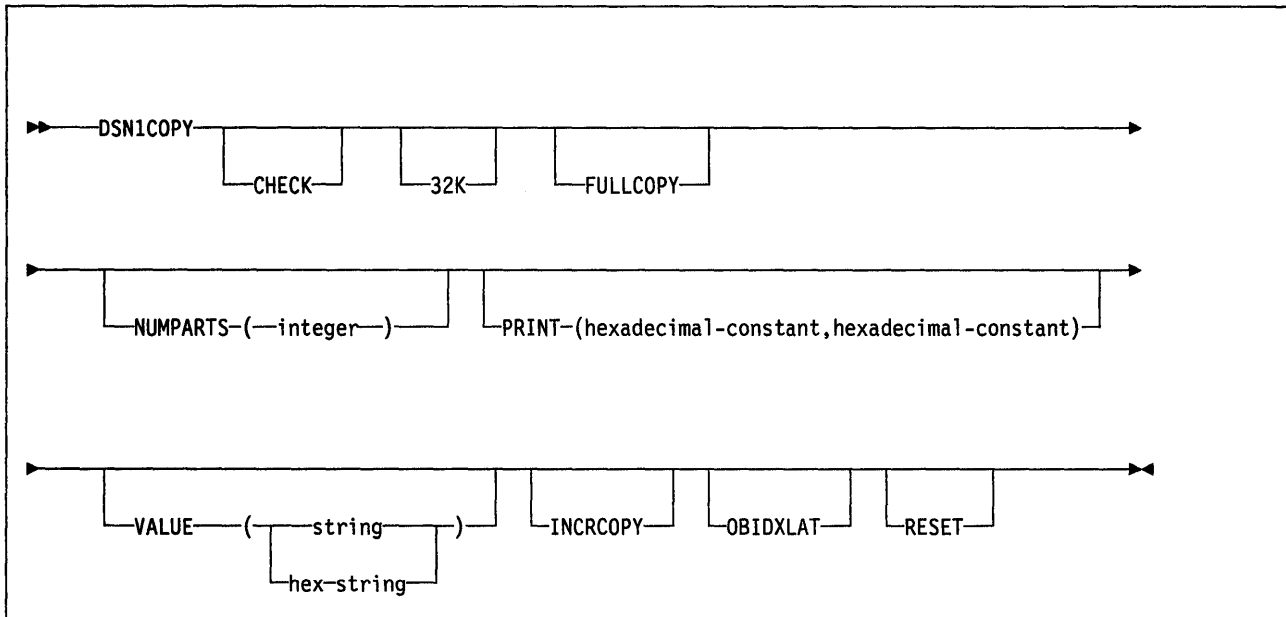
Run DSN1COPY as an MVS job.

You can use DSN1COPY even when the DB2 subsystem is not operational. If you choose to use DSN1COPY when the DB2 subsystem is operational, you should be sure that the DB2 data sets that are to be copied are not currently allocated to DB2.

To make sure that a data set is not currently allocated to DB2, issue the DB2 -STOP DATABASE command, specifying the database that contains the table space(s) and index(es) you want to copy.

Authorization

None is required. However, if any of the data sets are RACF protected, the primary authorization ID designated by the process must have the necessary RACF authority. Also, if the data sets are password protected, the primary authorization ID designated by the process must have the appropriate VSAM passwords to execute this service aid.



Parameter Descriptions

Specify one or all of the parameters listed below on the EXEC card to run DSN1COPY. If you specify more than one parameter:

- Separate them by commas (and no blanks).
- Specify them in any order.

CHECK

Causes each page from the SYSUT1 data set to undergo validity checking. The validity checking operates on one individual page at a time and does not include any cross-page checking. If an error is found during validity checking, a message is issued that describes the type of error, and the hexadecimal dump of the page is sent to the SYSPRINT data set. (If you don't receive any messages, no errors were found.) If more than one error exists in a given page, the validity checking identifies only the first of the errors. The entire page is dumped, however; see *Diagnosis Guide and Reference* for an explanation of how to analyze a page yourself.

Note: DSN1COPY does not perform validity checking on page 0's (header pages) or on space map pages.

32K

Specifies that the SYSUT1 data set has a 32K-byte page size. This option must be specified if the SYSUT1 data set has a 32K-byte page size. If you don't specify this parameter, the default page size of 4K bytes is assumed.

FULLCOPY

Specifies that you want to use a full image copy of your data as input to DSN1COPY. If this data is partitioned, you will also need to specify the NUMPARTS parameter in order to identify the number and length of the partitions. If you specify FULLCOPY without including a NUMPARTS specification, DSN1COPY will assume that your input file is not partitioned.

NUMPARTS(integer)

Specifies the number of partitions associated with the data set whose full image copy you are using as input to DSN1COPY, or whose page range you are printing. Valid specifications range from 1 to 64. DSN1COPY uses this

value to calculate the size of its output data sets and to help locate the first page in a range to be printed. If you omit NUMPARTS, or specify it as 0, DSN1COPY will assume that your input file is not partitioned.

Note: DSN1COPY cannot always validate the NUMPARTS parameter. If you specify it incorrectly, DSN1COPY may copy the data to the wrong data sets, return an error message indicating that an unexpected page number was encountered, or fail to allocate the data sets correctly (in which case a VSAM PUT error may be detected, probably resulting in a request parameter list (RPL) error code of 24).

PRINT(*hexadecimal-constant,hexadecimal-constant*)

Causes the SYSUT1 data set to be printed in hexadecimal format on the SYSPRINT data set. You may enter the PRINT parameter with or without page range specifications (*hexadecimal-constant,hexadecimal-constant*). If you do not specify a range, all pages of the SYSUT1 will be printed. If you want to limit the range of pages printed, you may do so by indicating the beginning and ending page numbers with the PRINT parameter or, if you want to print a single page, by indicating only the beginning page. In either case, your range specifications must be from one to six hexadecimal digits in length.

The following example shows how you would code the PRINT parameter if you wanted to begin printing at page X'2F0' and to stop at page X'35C':

```
PRINT(2F0,35C)
```

Note: Because the CHECK and RESET options, as well as the COPY function itself, run independently of the PRINT range, they will apply to the entire input file whether or not a range of pages is being printed. See usage note 9 on page 206 for a warning about this parameter.

VALUE

Causes each page of the input data set SYSUT1 to be scanned for the character string you specify in parentheses following the VALUE parameter. Each page that contains that character string will then be printed in SYSPRINT. The VALUE parameter may be specified in conjunction with any of the other DSN1COPY parameters.

string

May consist of 1 to 20 alphanumeric characters.

hexadecimal-constant

May consist of 2 to 40 hexadecimal characters.

If hexadecimal characters are specified, they must be enclosed in single quotation marks.

If, for example, you want to search your input file for the string **12345**, your JCL might look like the following example:

```
//STEP1 EXEC PGM=DSN1COPY,PARM='VALUE(12345)'
```

On the other hand, you might want to search for the equivalent hexadecimal character string, in which case your JCL might look like this:

```
//STEP1 EXEC PGM=DSN1COPY,PARM='VALUE(''F1F2F3F4F5'')'
```

Note: See the usage notes on pages 203 through 206 for a warning about this parameter.

INRCOPY

Specifies that you want to use an incremental image copy of your data sets as input to DSN1COPY. Before you apply an incremental image copy to your data set, you must first apply a full image copy to the data set, using the FULLCOPY

parameter, in a separate execution step. You must apply each incremental image copy in a separate step, starting with the oldest incremental image copy. It is an error to specify both the FULLCOPY and the INRCOPY parameters in the same execution step. Specifying neither FULLCOPY nor INRCOPY implies that the input is not image copy data sets; therefore, only a single output data set will be used. DSN1COPY with the INRCOPY parameter does updates to existing data sets; therefore, redefinition of the data sets must not be done.

OBIDXLAT

Specifies that OBID translation must be done before the DB2 data set is copied. With this parameter specified, DSN1COPY requires additional input from SYSXLAT file by using the DD cards (see Usage Notes, below).

Note: DSN1COPY can only translate up to 100 record OBIDs at a time.

RESET

Causes the log RBAs in each index or data page to be reset to 0. If you specify this option, CHECK processing will be performed regardless of whether you specified the CHECK option.

CAUTION:

You should only use this option when the output file will be used to build a DB2 table space that will be processed on a DB2 subsystem that has a different recovery log than the source subsystem. Failure to specify RESET in such a case can eventually result in an abend during subsequent update activity. The abend reason code in this case is 00C200C1, indicating that the specified RBA value is outside the valid range of the recovery log.

Usage Notes

1. You should use the CHECK option when critical data is involved to prevent the undetected copying of inconsistent data to the output data set.
2. Under either of the following circumstances you should run a CHECK utility job on the table space involved to ensure that there are no inconsistencies between data and indexes on the data:
 - Before using DSN1COPY to save critical data that is indexed
 - After using DSN1COPY to restore critical data that is indexed.
3. If you use DSN1COPY to load data into a table space, and you do **not** specify the OBIDXLAT parameter, you must be careful not to invalidate DB2 internal identifiers (object descriptors, or OBIDs) that are embedded within the data. These OBIDs (described in Section 4 of *Diagnosis Guide and Reference*) can become invalid in the following ways:
 - When tables are dropped and re-created after the version of the data DSN1COPY saved was created and before it was used.
 - When the target subsystem differs from the source subsystem in any of these ways:
 - Table space attributes of BUFFERPOOL or NUMPARTS
 - Table attributes other than table name, table space name, and database name
 - The order in which all table spaces, indexes, and tables are defined or dropped in the source and target databases in the two subsystems.

To protect against invalidating the OBIDs, it is recommended that you specify the OBIDLAT parameter of DSN1COPY in order to perform OBID, DBID or PSID translation before the data is copied.

4. DSN1COPY uses several DD cards, each of which is listed and described below:

SYSPRINT Defines the data set that contains output messages from the DSN1COPY program and all hexadecimal dump output.

SYSUT1 Defines the input data set. This data set can be a sequential data set or a VSAM data set. DSN1COPY assumes that the block size is 4096 bytes (as is standard for DB2 data sets).
Disposition for this data set should be specified as OLD (DISP=OLD) to ensure that it is not in use by DB2. Disposition for this data set should be specified as SHR (DISP=SHR) only in circumstances where the DB2 -STOP DATABASE command does not work.

Note: The requested operation will take place only for the data set specified. If the input data set is a partitioned table space or index or if it is part of a linear table space that is larger than 2 gigabytes, care must be taken to ensure the correct data set is specified. For example, to print a page range in the second partition, the data set name must specify the second data set in the group of VSAM data sets comprising the table space (in other words, DSN=...A002).

SYSUT2 Defines the output data set. This data set can be a sequential data set, a VSAM data set, or a DUMMY data set.

Notes:

- a. Except when INRCOPY is specified, DSN1COPY assumes that the output data sets are empty (that is, the program adds the blocks). You must use access method services to redefine all the VSAM output data sets you are restoring before you run DSN1COPY. Be sure that any output VSAM data sets are empty (newly defined) before running this program.
- b. You might want to specify a DUMMY SYSUT2 DD card if you are doing only page checking or page dumping (as opposed to doing copying of data sets without doing checking or dumping).
- c. Do not code unit and volume serial parameters when using VSAM data sets so that necessary information may be obtained from the catalog.

SYSXLAT Defines the DBIDs, PSIDs, and OBIDs (ISOBIDs for indexes) to be translated.

If you have dropped a table without a subsequent REORG of the table space, then you will need to REORG the table space before running DSN1COPY with the OBIDLAT option. This removes any records that have been previously dropped from the table space.

Each record in the SYSXLAT file must contain a pair of decimal integers, less than 10000 and separated by a nonnu-

meric character. The first integer of each record pertains to the source and the second integer pertains to the target. The first record in the SYSXLAT file contains the source and target DBIDs. The second record contains the source and target PSIDs (page set IDs) or ISOBIDs for indexes. All subsequent records in the SYSXLAT data set are for table OBIDs. Only the first two records are required for an index space. Sample data in a SYSXLAT file may look as follows:

```
0114,0314
0001,0007
0003,0009
0004,0010
0005,0011
```

The user can create the SYSXLAT file by running the DSNTEP2 sample application on both the source and target systems to obtain the NAMES, DBIDs, PSIDs, ISOBIDs, and OBIDs of the tables and indexes. The following examples of DSNTEP2 JCL and input yield the above information:

-- For table spaces:

```
//EXECUTE EXEC PGM=IKJEFT01
//DSNTRACE DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN S(DSN)
RUN PROGRAM(DSNTEP2) PLAN(DSNTEP22)
/*
//SYSIN DD *
SELECT DBID, PSID FROM SYSIBM.SYSTABLESPACE
  WHERE NAME='tablespace_name'
  AND CREATOR='creator_name';
SELECT NAME, OBID FROM SYSIBM.SYSTABLES
  WHERE TSNAME='tablespace_name'
  AND CREATOR='creator_name';
/*
```

-- For index spaces:

```
//EXECUTE EXEC PGM=IKJEFT01
//DSNTRACE DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSIN DD *
DSN S(DSN)
RUN PROGRAM(DSNTEP2) PLAN(DSNTEP22)
/*
//SYSIN DD *
SELECT DBID, ISOBID FROM SYSIBM.SYSINDEXES
  WHERE NAME='index_name'
  AND CREATOR='creator_name';
/*
```

5. If you intend to include the FULLCOPY parameter in order to use image copies as input to DSN1COPY, be sure that those image copies are produced using the COPY utility with the SHRLEVEL(REFERENCE) parameter included. This will ensure that the data contained in your image copies is consistent.
6. Before invoking DSN1COPY with the FULLCOPY parameter, make sure that the table space or index has been defined with REUSE. If the table space or index has not been defined with REUSE, you must use access method services to redefine all the VSAM data sets that constitute the table space or index space you are restoring. The data set name you specify in the SYSUT2 DD statement, then, is the name of the first data set for that space, unless you are using the FULLCOPY parameter to copy a single partition of a table or index space. In this instance, the SYSUT2 DD statement should specify the name of the partition being copied.
7. After using the FULLCOPY parameter to restore a table space, you will need to recover any indexes associated with that table space. You may do this by using the RECOVER utility and specifying the INDEX parameter.

For more information on the RECOVER utility, refer to 257.

8. If you want to print one or more pages without having the copy function performed, it is advisable to use DSN1PRNT to avoid possible unnecessary reading of the input file.
9. When you use DSN1COPY for printing purposes, you should specify the PRINT parameter. The requested operation will take place only for the data set specified. If the input data set is a partitioned table space or index or if the input data set is part of a linear table space which is larger than 2 gigabytes, care must be taken to ensure the correct data set is specified. For example, to print a page range in the second partition, the data set name must specify the second data set in the group of VSAM data sets comprising the table space (in other words, DSN = ...A002).

To print a full image copy data set (rather than recover a table space), specify a DUMMY SYSUT2 DD card and do not specify the FULLCOPY parameter.

Examples of Using DSN1COPY

Several examples of using DSN1COPY are identified below:

- Create a backup copy of a DB2 data set
 - SYSUT1: DB2/VSAM
 - SYSUT2: sequential data set
- Restore a backup copy of a DB2 data set
 - SYSUT1: sequential data set
 - SYSUT2: DB2/VSAM
- Move a DB2 data set to another DB2 data set compact.
 - SYSUT1: DB2/VSAM
 - SYSUT2: DB2/VSAM
- Perform validity checking on a DB2 data set
 - SYSUT1: DB2/VSAM
 - SYSUT2: DUMMY
 - Parameter: CHECK

- Perform validity checking on and print a DB2 data set
 - SYSUT1: DB2/VSAM
 - SYSUT2: DUMMY
 - Parameters: CHECK,PRINT
- Restore a table space from a nonpartitioned image copy data or page set
 - SYSUT1: DB2 full image copy
 - SYSUT2: DB2/VSAM
 - Parameter: FULLCOPY
- Restore a table space from a partitioned image copy data or page set
 - SYSUT1: DB2 full image copy
 - SYSUT2: DB2/VSAM
 - Parameters: FULLCOPY,NUMPARTS(*nn*)
- Perform RBA RESET on a DB2 data set
 - SYSUT1: VSAM or sequential data set
 - SYSUT2: DB2/VSAM
 - Parameter: RESET

Notes:

In order for your output data set to be useful, you must make sure that it has the same name as the data set you are resetting. You may do this in one of two ways:

1. Use DSN1COPY to copy your existing data set to a sequential data set. Specify this data set as SYSUT1. Next, delete and redefine the data set you copied, specifying this as SYSUT2 if the data set was defined without the REUSE parameter.
2. Use your existing DB2 data set as the SYSUT1 specification, creating a new VSAM data set for SYSUT2. After the reset operation has been completed, delete the data set you specified as SYSUT1 and rename the SYSUT2 data set, giving it the name of the data set you just deleted.

Sample JCL for Running DSN1COPY

Figure 22, Figure 24 on page 208 and Figure 23 on page 208 are samples of JCL used for running DSN1COPY.

```
//RUNCOPY EXEC PGM=DSN1COPY,PARM='CHECK'
//* COPY VSAM TO SEQUENTIAL AND CHECK PAGES
//STEPLIB DD DSN=PDS CONTAINING DSN1COPY
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=DSNCAT.DSNDBC.DSNDB01.SYSUTIL.I0001.A001,DISP=OLD
//SYSUT2 DD DSN=TAPE.DS,UNIT=TAPE,DISP=(NEW,KEEP),VOL=SER=UTLBAK
```

Figure 22. Sample JCL for Running DSN1COPY

```
//EXECUTE EXEC PGM=DSN1COPY,PARM='OBIDXLAT'  
//STEPLIB DD DSN=PDS CONTAINING DSN1COPY  
//SYSPRINT DD SYSOUT=A  
//SYSUT1 DD DSN=DSNCAT.DSNDBC.SYSUTIL.I0001,A001.DISP=SHR  
//SYSUT2 DD DSN=TAPE.DS.UNIT=TAPE,DISP=(NEW,KEEP)  
//SYSXLAT DD *  
0114,0314  
0001,0007  
0003,0009  
0004,0010  
0005,0011  
/*
```

Figure 23. Sample JCL for Running DSN1COPY Using the OBIDXLAT Parameter with the SYSXLAT DD Card

```
//PRINT EXEC PGM=DSN1COPY,PARM='PRINT(2002A1),NUMPARTS(8)'  
/* PRINT A PAGE IN THE THIRD PARTITION OF A TABLESPACE CONSISTING  
/* OF 8 PARTITIONS.  
//SYSUDUMP DD SYSOUT=A  
//SYSPRINT DD SYSOUT=A  
//SYSUT2 DD DUMMY  
//SYSUT1 DD DSN=DSNCAT.DSNDBD.MMRDB.PARTEMP1.I0001.A003,DISP=OLD
```

Figure 24. Sample JCL for Printing a Page with DSN1COPY

Return Codes Issued During DSN1COPY Processing

- RC=0** Processing completed successfully.
- RC=4** All pages have been processed, but some errors (for instance, pages contain inconsistent data, warning from ACB open) have been detected.
- RC=8** Processing did not complete successfully (for example, a VSAM GET or PUT error might have occurred).

DSN1LOGP (Service Aid)

DB2 provides a service aid, called DSN1LOGP, that reads the contents of the recovery log and formats the contents for display. Two report formats are available to you:

- A *detail report* formats and displays individual log records. This information is designed to help IBM support center personnel resolve problems that require extensive analysis of the recovery log contents. This section does not include an extensive description of the detail report.
- A *summary report* summarizes the information contained in the log in order to help you to perform a conditional restart.

When you use DSN1LOGP, you can specify the range of the recovery log that you want processed and, optionally, selection criteria within the range to limit the records displayed to those you want to see.

To specify optional range criteria, you can:

1. Name one or more units of recovery by specifying a set of unit of recovery IDs (URIDs).
2. Name a single database.

You can use DSN1LOGP to display the contents of the entire recovery log by not specifying a range or these additional criteria. By specifying a single URID, you can display recovery log records that correspond only to the unit of recovery you are interested in. By specifying a single database, you can display recovery log records that correspond only to the database you are interested in. Lastly, by specifying a URID and a database, you can display recovery log records that correspond to the use of one database by a single unit of recovery.

Environment

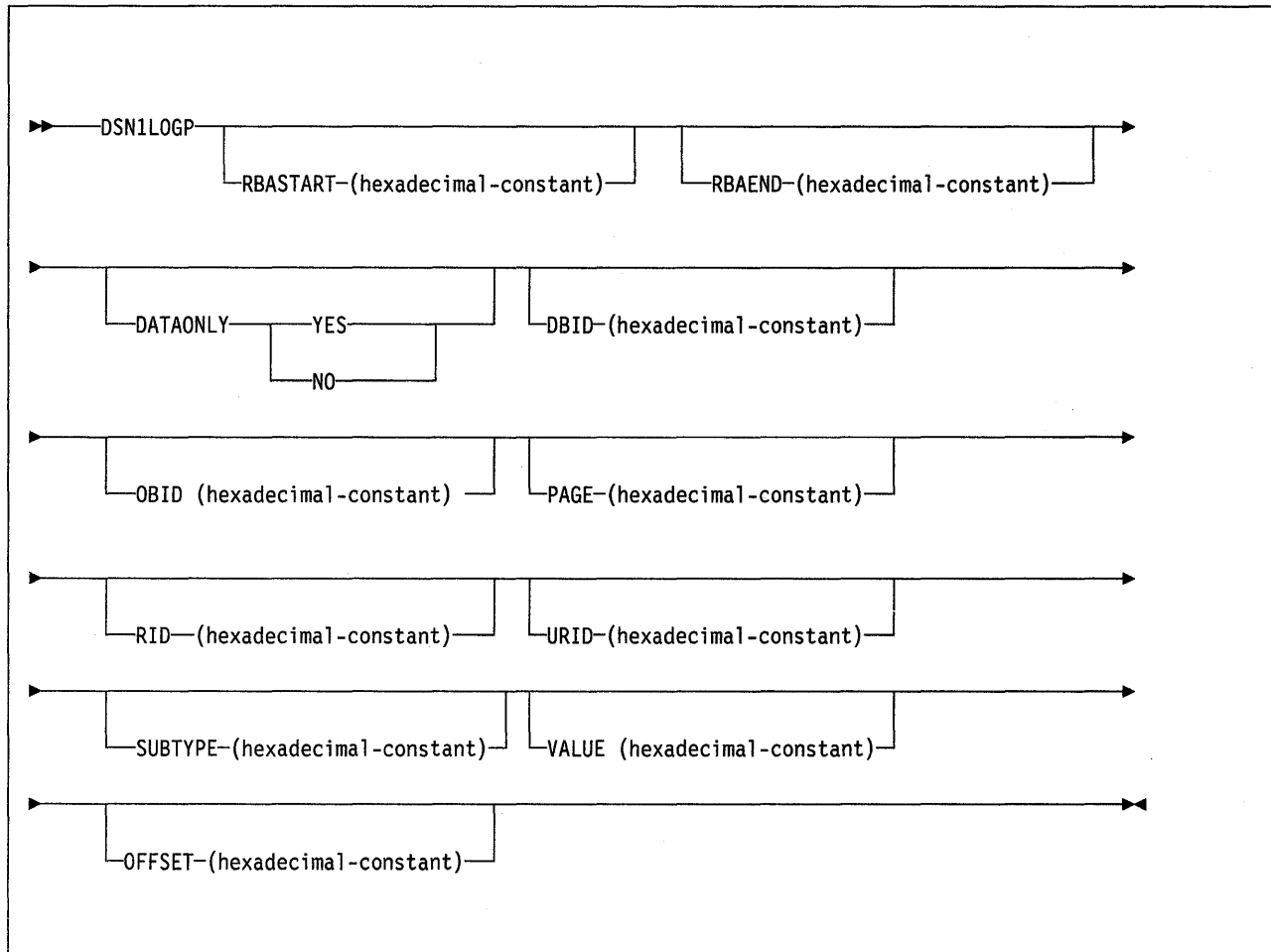
To display the contents of the DB2 recovery log, you should execute DSN1LOGP as a batch MVS job.

DSN1LOGP does not communicate with the DB2 online subsystem. The program cannot access a DB2 recovery log data set at the same time the DB2 online subsystem is accessing that data set. Therefore, DSN1LOGP cannot access any of the active or archive recovery log data sets that the DB2 online system is actively using. You may not run DSN1LOGP while DB2 is active.

Authorization

DSN1LOGP requires no special authorization. However, if any of the data sets involved are RACF protected, the primary authorization ID designated by the process must have RACF authority. Similarly, if the necessary data sets are password protected, the primary authorization ID designated by the process must have the appropriate VSAM passwords.

DSN1LOGP (Service Aid)



Parameter Descriptions

To execute DSN1LOGP, construct a batch job. The service aid name, "DSN1LOGP", should appear on the EXEC statement, as shown in Figure 25 on page 211.

Figure 25 is an example of the JCL you should use to extract the contents of the BSDS for display.

When you invoke DSN1LOGP, you should provide the following DD statements:

SYSPRINT All error messages, exception conditions, and the detail report are written to the SYSPRINT file. The logical record length (LRECL) is 131.

SYSIN Input selection criteria can be specified on this statement. The control statement keywords are described under "Control Statement and Keywords" on page 214. The LRECL must be 80. DSN1LOGP will inspect the first 72 characters for keywords and values, but will not examine the last 8 characters. This permits you to place sequence information in columns 73 through 80.

SYSSUMRY If you plan to generate a summary report, the formatted output is written to the SYSSUMRY file. The logical record length (LRECL) is 131. For an example of the appropriate JCL, see Figure 28 on page 213.

The recovery log is identified by DD statements described by the stand-alone log services. For a description of these services, see Section 1 of *Diagnosis Guide and Reference*.

The following sections provide a summary of information about those specifications and describe different circumstances in which you might want to use DSN1LOGP.

- “Using DSN1LOGP With an Available BSDS”
- “Using DSN1LOGP on Active Log Data (No BSDS Available)”
- “Using DSN1LOGP on Archive Log Data (No BSDS Available)” on page 212
- “Using DSN1LOGP with the SUMMARY Option” on page 213.

Using DSN1LOGP With an Available BSDS: The BSDS identifies and provides information about all active log data sets and archive log data sets that exist in your DB2 subsystem. When you specify the BSDS to DSN1LOGP, you must also provide the beginning and ending relative byte addresses (RBAs) for the range of the recovery log you want displayed. DSN1LOGP will then associate the beginning and ending RBA specifications you provide with the appropriate data set names.

To specify the beginning RBA, use the RBASTART keyword on your statement, and, to specify the ending RBA, use the RBAEND keyword. (Both are described under “Control Statement and Keywords” on page 214.)

```
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB DD DSN=PDS containing DSN1LOGP
//SYSPRINT DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//BSDS DD DSN=DSNCAT.BSDS01,DISP=SHR
//SYSIN DD *
    RBASTART (AF000) RBAEND (B3000)
    DBID (10A) OBID(1F)
/*
```

Figure 25. Example of the JCL to Invoke DSN1LOGP to Extract Log Data Using the BSDS

You can think of the DB2 recovery log as a large sequential file. Whenever recovery log records are written, they are written to the end of the log. A log RBA is the address of a byte on the log. Because it is larger than a single data set, the recovery log is physically stored on many data sets. DB2 records the RBA ranges and their corresponding data sets in the BSDS. To determine which data set contains a specific RBA, read the information about the Print Log Map utility on page 254 and in Section 5 of *System and Database Administration Guide*. During normal DB2 operation, multiple messages are issued that include information about log RBAs.

Using DSN1LOGP on Active Log Data (No BSDS Available): If the BSDS is not available and if the active log data sets involved have been copied and sent to you, you can indicate the set of active log data sets to be processed by DSN1LOGP by specifying one or more ACTIVE_n DD statements as shown in Figure 26 on page 212. If the REPRO command of access method services was used to copy the active log to tape, you must specify this data set in an archive DD statement. Each DD statement you include specifies another active log data set. If you specify more than one active log data set, you must list the ACTIVE_n DD statements in ascending log RBA sequence.

For example, ACTIVE1 must specify a portion of the log that is less than ACTIVE2; ACTIVE2 must specify a portion of the log that is less than ACTIVE3. If you do not specify this correctly, errors that DSN1LOGP doesn't detect can occur.

When you specify active log data sets, you don't need to use the RBASTART and RBAEND keywords (as you do when you specify BSDS). DSN1LOGP scans all active log data sets the job indicates, as long as they are in the correct log RBA sequence.

```
//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB DD DSN=PDS containing DSN1LOGP
//SYSPRINT DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//ACTIVE1 DD DSN=DSNCAT.LOGCOPY1.DS02,DISP=SHR      RBA X'A000' - X'BFFF'
//ACTIVE2 DD DSN=DSNCAT.LOGCOPY1.DS03,DISP=SHR      RBA X'C000' - X'FFFF'
//ACTIVE3 DD DSN=DSNCAT.LOGCOPY1.DS01,DISP=SHR      RBA X'F000' - X'12FFF'
//SYSIN DD *
          DBID (10A) OBID(1F) PAGE(3B) PAGE(8C)
/*
```

Figure 26. Example of the JCL to invoke DSN1LOGP to Extract Contents of Active Logs

Using DSN1LOGP on Archive Log Data (No BSDS Available): If the BSDS is not available (as described under "Using DSN1LOGP on Active Log Data (No BSDS Available)," above), you can indicate which archive log data sets are to be processed by DSN1LOGP by specifying one ARCHIVE DD statement, concatenated with one or more DD statements as shown in Figure 27 on page 213. Each DD statement you include specifies another archive log data set. If you specify more than one archive log data set, you must list the DD statements corresponding to the multiple archive log data sets in ascending log RBA sequence. If you do not specify this correctly, errors that DSN1LOGP doesn't detect can occur.

When you specify archive log data sets, you don't need to use the RBASTART and RBAEND keywords. DSN1LOGP scans all archive log data sets the job indicates, as long as they are in the correct log RBA sequence.

If your archive logs are stored on tape, be aware that, during the archiving process, two files are constructed on tape. The first file is the BSDS, and the second is a dump of the active log currently being archived. If a failure occurs during the time the BSDS is being archived, DB2 might omit the BSDS. In this case, the first file contains the active log, because DB2 overlooks the BSDS.

If archiving is performed on tape, also be aware that the first letter of the lowest-level qualifier of the archived information varies for the first and second data sets on the tape: The first letter of the first data set is "B" (for BSDS); the first letter of the second data set is "A" (for archive). Hence, in Figure 27, the data set names all end in "Axxxxxx", and the DD statement identifies each of them as the second data set on the corresponding tape ((2,SL)).

```

//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB DD DSN=PDS containing DSN1LOGP
//SYSPRINT DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//ARCHIVE DD DSN=DSNCAT.ARCHLOG1.A0000037,UNIT=TAPE,VOL=SER=T10067,
//          DISP=(OLD,KEEP),LABEL=(2,SL)
//          DD DSN=DSNCAT.ARCHLOG1.A0000039,UNIT=TAPE,VOL=SER=T30897,
//          DISP=(OLD,KEEP),LABEL=(2,SL)
//          DD DSN=DSNCAT.ARCHLOG1.A0000041,UNIT=TAPE,VOL=SER=T06573,
//          DISP=(OLD,KEEP),LABEL=(2,SL)
//SYSIN DD *
          RBASTART (61F321)
          URID (61F321) DBID(4)
/*

```

Figure 27. Example of the JCL to invoke DSN1LOGP to Extract Contents of Archive Logs

Using DSN1LOGP with the SUMMARY Option: The DSN1LOGP SUMMARY option allows you to scan the recovery log in order to determine what work is incomplete at restart time.

You may specify this option either by itself or when you use DSN1LOGP to produce a detail report of log data. Summary log results appear in SYSSUMRY, so you must include a SYSSUMRY DD statement as part of the JCL with which you invoke DSN1LOGP.

Figure 28 is an example of the JCL necessary to invoke DSN1LOGP. In this case, the program will produce both a detail and a summary report using the BSDS to identify the log data sets. DBID, OBID, URID, and PAGE specifications do not affect the summary option, so the summary report will summarize *all* information on the log within the RBASTART and RBAEND specifications.

```

//STEP1 EXEC PGM=DSN1LOGP
//STEPLIB DD DSN=PDS containing DSN1LOGP
//SYSPRINT DD SYSOUT=A
//SYSSUMRY DD SYSOUT=A
//SYSABEND DD SYSOUT=A
//BSDS DD DSN=DSNCAT.BSDS01,DISP=SHR
//SYSIN DD *
          RBASTART (AF000) RBAEND (B3000)
          DBID (10A) OBID(1F) SUMMARY(YES)
/*

```

Figure 28. Example of the JCL to Invoke DSN1LOGP to Produce a Summary Report. The facility can produce both a detail report and a summary report, as it will when invoked with this JCL.

Valid specifications for the SUMMARY control statement include:

SUMMARY(NO | YES | ONLY)

- When you want to generate both a detail and a summary report, specify YES, as in Figure 28.
- If you want to generate only a summary report, specify ONLY.

- If you want to generate only a detail report, specify NO.

NO is the default.

Control Statement and Keywords

Specify the control statement with the appropriate keywords in the SYSIN file. If you choose not to specify any keywords in the control statement, you can either use a SYSIN file with no keywords following it, or you can omit the SYSIN file from the job JCL.

For information about the rules for using the control statement keywords, see “Usage Notes” on page 217.

The keywords you can use in your control statement are described below. Each can have alternative spellings or abbreviations, as noted.

RBASTART (*hexadecimal-constant*)

Specifies the hexadecimal log RBA (*hexadecimal-constant*) from which to begin extraction. You can specify RBASTART, as shown above, STARTRBA, or ST. For any given job, specify this keyword only once.

RBAEND (*hexadecimal-constant*)

Specifies the last valid hexadecimal log RBA (*hexadecimal-constant*) that is to be extracted. You can specify RBAEND, as shown above, ENDRBA, or EN. To indicate that the search should end at the last valid RBA in the log, specify RBAEND(FFFFFFFFFFFF); DSN1LOGP scans the log, starting with the RBASTART value and ending at the end of the log. For any given job, specify this keyword only once.

DATAONLY

Specifies a YES or NO for DATAONLY. The DATAONLY option limits the log records extracted in DSN1LOGP’s detail report to those that represent data modifications (insert, page repair, update space map, and so forth). For example, specifying DATAONLY in conjunction with the DBID and OBID will result in only the log records that modified data for that DBID and OBID.

NO

Yields a detail report with all record types; this is synonymous with not specifying the DATAONLY option.

YES

Extract log records that represent data modifications.

The DATAONLY option can only be specified once.

DBID (*hexadecimal-constant*)

Specifies a hexadecimal database identifier (DBID). If you specify a DBID, DSN1LOGP extracts only records associated with the DBID you specify. For any given job, specify this keyword only once.

The DBID is displayed in many DB2 messages. You can also find the DBID in the DB2 catalog for a specific object (for example, in the column named “DBID” of the SYSIBM.SYSTABLESPACE catalog table).

OBID (*hexadecimal-constant*)

Specifies a hexadecimal database object identifier (OBID). “OBID” is a general term used for all DB2 object identifiers within a database. In this context, you should be most interested in OBIDs for two of the objects: data page set OBIDs (PSIDs), and index page set OBIDs (ISOBIDs).

For information about OBIDs, see Section 4 of *Diagnosis Guide and Reference*.

Each time a change is made to data or indexes in a database, DB2 records information about the change in the recovery log. This information is in the form of “undo” and “redo” recovery log records.

Whenever DB2 makes a change to data, a recovery log record describing the change is recorded in the log and identifies the database (by DBID) and the table space (by page set ID, or PSID). You can find the PSID for a table space in the column named “PSID” in the SYSIBM.SYSTABLESPACE catalog table.

Note: Also in the SYSIBM.SYSTABLESPACE catalog table, you will find a column named “OBID”; this column actually contains the “OBID” of a file descriptor, and *should not be confused with* the PSID, which is the information you must include when you invoke DSN1LOGP.

Whenever DB2 makes a change to an index, a recovery log record describing the change is recorded in the log and identifies the database (by DBID) and the index space (by index space OBID, or ISOBID). You can find the ISOBID for an index space in the column named “ISOBID” in the SYSIBM.SYSINDEXES catalog table.

Note: Also in the SYSIBM.SYSINDEXES catalog table, you will find a column named “OBID”; this column actually contains the “OBID” of a fan set descriptor, and *should not be confused with* the ISOBID, which is the information you must include when you invoke DSN1LOGP.

When you SELECT the DBID, PSID, or ISOBID from a catalog table, the value is displayed in decimal format. Because DSN1LOGP requires that these values be specified in hexadecimal format, you must convert the IDs from decimal to hexadecimal.

For any given DSN1LOGP job, use this keyword only once. If you specify OBID, you must also specify DBID.

PAGE (*hexadecimal-constant*)

Specifies a hexadecimal page number (*hexadecimal-constant*). When data or an index is changed, a recovery log record is written to the log, identifying the DBID, OBID, and the page number of the data or index page being modified. By specifying a PAGE number in your DSN1LOGP job, you can limit the search to a single page; otherwise, all pages for a given combination of DBID and OBID are extracted.

You can specify a maximum of 100 PAGE keywords in any given DSN1LOGP job. You can specify the PAGE keyword(s) only if you also specify the DBID and OBID keywords that correspond to the page(s) you specify.

RID (*hexadecimal-constant*)

Specifies a 4-byte hexadecimal number, with the first 3 bytes representing the page number and the last byte representing the record ID (*hexadecimal-constant*). You can use the RID option to limit the log records extracted in DSN1LOGP to those associated with a particular record. By specifying the RID keyword in a DSN1LOGP invocation, only those log records associated with the record ID will be extracted. The log records extracted will include not only those records directly associated with the RID (insert, delete,...), but also control records associated with the DBID and OBID specifications (page set open, page set close, set write, write,...).

You can specify a maximum of 38 RID keywords in any given DSN1LOGP job. You can only specify the RID keyword if the DBID and OBID keywords correspond to the record(s) specified.

Note: The PAGE and RID keywords cannot both be specified.

URID (*hexadecimal-constant*)

Specifies a hexadecimal unit of recovery identifier (*hexadecimal-constant*). Changes to data and indexes occur in the context of a DB2 unit of recovery. A unit of recovery is identified on the recovery log via a BEGIN UR record. The log RBA of that BEGIN UR record is the URID value you should use. If you know the URID for a given UR that you are interested in, you can limit the extraction of information from the DB2 log to that URID.

You can specify a maximum of 10 URID keywords in any given DSN1LOGP job. In order to narrow the search, you can specify URID keywords in a job that contains other keywords

SUBTYPE (*hexadecimal-constant*)

Restricts log records to those belonging to a particular data manager operation.

hexadecimal-constant is a hexadecimal data manager log operation code; these codes are listed below.

Constant Value	Description
1	Update File Data Page
2	Format Page or Update Space Map
3	Update Space Map Bits
4	Update Index Space Map
5	Update Index Page
6	RBA Table Log Record
7	Checkpoint RBA Table Log Record
8	Soft Log a CUB Position
9	DBD Virtual Memory Copy
10	Exclusive Lock on Pageset Partition or DBD
11	Format File Pageset
12	Format Index Pageset
13	Invalidate Object - CTDB, PSCB, or CUB
14	Temporary File Created
15	Update by Repair (first half if 32K)
16	Update by Repair (second half if 32K)
17	Allocating or Deallocating a Segment Entry

The VALUE and OFFSET options must be used together; you may specify a maximum of 10 VALUE/OFFSET pairs.

VALUE (*hexadecimal-constant*)

Specifies a value to be located in the log record before the record is chosen for output.

Value has a maximum of 64 hexadecimal characters.

OFFSET (*hexadecimal-constant*)

Specifies an offset from the log record header at which DSN1LOGP is to begin scanning for the value specified in the VALUE option.

The hexadecimal offset has a maximum of 8 characters.

Usage Notes

1. You can specify as many as 50 records for a given job. All records will be concatenated into a single contiguous string. Remember that DSN1LOGP will inspect only the first 72 characters of your control statement for values, and that you may therefore use the last 8 characters for sequence information.
2. Specifications are of this form:

keyword(hexadecimal-constant)

For example, to specify a database identifier of X'10A', you should specify the following:

DBID(10A)

You can include blanks between keywords, and also between the keywords and the corresponding values. For example:

DBID (10A)

3. **Specify all values in hexadecimal**, not in decimal.
4. A maximum of 40 keywords is permitted.
5. You can specify keywords in any order.

Examples

Three examples are discussed under "Environment" on page 209, and the JCL for each is included in figures. The section below summarizes the scenarios covered by each of these examples.

Example 1 This example (shown in Figure 25 on page 211) shows how to extract the information from the recovery log when you have the BSDS available. The extraction starts at the log RBA of X'AF000' and ends at the log RBA of X'B3000', for the table space or index space identified by the DBID of X'10A' (266 decimal) and the OBID of X'1F' (31 decimal).

Example 2 This example (shown in Figure 26 on page 212) shows how to extract the information from the active log when the BSDS is not available. The extraction includes log records that apply to the table space or index space identified by the DBID of X'10A' and the OBID of X'1F'. The only information that is extracted is information relating to page numbers X'3B' and X'8C'. Note that no beginning and ending RBA values are specified. You can omit beginning and ending RBA values for ACTIVEn or ARCHIVE DD statements, because the DSN1LOGP search includes all specified ACTIVEn DD statements. Note also that the DD statements ACTIVE1, ACTIVE2, and ACTIVE3 specify the log data sets in ascending log RBA range. Use the Print Log Map utility to determine what the log RBA range is for each active log data set. If the BSDS is not available, and you cannot determine the ascending log RBA order of the data sets, each log data set will have to be run individually.

Example 3 This example (shown in Figure 27 on page 213) shows how to extract the information from archive logs when the BSDS is not available. The extraction includes log records that apply to a single unit of recovery (whose URID is X'61F321'). Because the BEGIN UR is the first record for the unit of recovery and it is at X'61F321', the beginning RBA is specified to indicate that it is the first RBA in the range from which to extract recovery log records. Also, because no ending RBA value is specified, all specified archive logs are scanned for qualifying log records. The specification of DBID(4) limits the scan to changes that have been made to all table spaces and index spaces in the database whose DBID is X'4' by the specified unit of recovery.

Example 4 This example (shown in Figure 28 on page 213) is like Example 1, in that it shows how to extract the information from the recovery log when you have the BSDS available. However, in addition to this information, Example 4 also shows you how to specify a summary report of all

DSN1LOGP (Service Aid)

logged information between the log RBA of X'AF000' and the log RBA of X'B3000'. This summary will be generated with a detail report, but will be printed to SYSSUMRY separately.

Sample Output

Figure 29 on page 219 shows a sample of the summary output that DSN1LOGP generates.

```
=====
DSN1150I SUMMARY OF COMPLETED EVENTS
```

```
DSN1153I CHECKPOINT START=000007425468 END=000007426C6C DATE=84.284 TIME=14:49:25
```

```
DSN1151I UR CONNID=UTILITY CORRID=C10ATLD AUTHID=SYSADM PLAN=DSNUTIL
START DATE=84.284 TIME=14:49:40 DISP=COMMITTED INFO=COMPLETE
START=0000074277B5 END=0000074278DC
```

```
DATA MODIFIED:
DATABASE=0001=DSNDB01 PAGESET=004F=SYSUTIL
```

```
DSN1151I UR CONNID=UTILITY CORRID=C10ATLD AUTHID=SYSADM PLAN=DSNUTIL
START DATE=84.284 TIME=14:49:40 DISP=COMMITTED INFO=COMPLETE
START=00000742792B END=000007427A60
```

```
DATA MODIFIED:
DATABASE=0001=DSNDB01 PAGESET=004F=SYSUTIL
```

```
DSN1213I LAST LOG RBA ENCOUNTERED 000007428A9E
```

```
DSN1214I NUMBER OF LOG RECORDS READ 0000000000000184
```

```
=====
DSN1157I RESTART SUMMARY
```

```
DSN1153I CHECKPOINT START=000007425468 END=000007426C6C DATE=84.284 TIME=14:49:25
```

```
DSN1162I UR CONNID=BATCH CORRID=PROGRAM2 AUTHID=ADMFO01 PLAN=TCEU02
START DATE=84.284 TIME=11:12:01 DISP=INFLIGHT INFO=COMPLETE
START=0000063DA17B
```

```
DATA MODIFIED:
DATABASE=0101=STVDB02 PAGESET=0002=STVTS02
```

```
DSN1162I UR CONNID=BATCH CORRID=PROGRAM5 AUTHID=ADMFO01 PLAN=TCEU05
START DATE=84.284 TIME=12:21:02 DISP=INFLIGHT INFO=COMPLETE
START=000006A57C57
```

```
DATA MODIFIED:
DATABASE=0104=STVDB05 PAGESET=0002=STVTS05
```

```
DSN1162I UR CONNID=TEST001 CORRID=CTHDCORID001 AUTHID=MULT002 PLAN=DONSQL1
START DATE=84.278 TIME=06:49:33 DISP=INDOUBT INFO=PARTIAL
START=000005FBCC4F
```

```
NO DATA MODIFIED (BASED ON INCOMPLETE LOG INFORMATION)
```

Figure 29. Sample DSN1LOGP Output

The SUMMARY report is divided into two distinct sections, the first headed by the message:

```
DSN1150I SUMMARY OF COMPLETED EVENTS
```

This section lists all completed units of recovery (URs) and checkpoints within the range of the log scanned. Events are listed chronologically: URs by the order of their completion, and checkpoints when the end of a checkpoint is processed. The Log Extractor will list the page sets modified by each completed UR. If a log record associated with a UR is unavailable (as it would be if, for example, the range of the log scanned were not large enough to contain all records for a given UR), the attribute INFO=PARTIAL is displayed for the UR.

The second section is headed by the message:

```
DSN1157I RESTART SUMMARY
```

Here the Log Extractor lists the work required of DB2 at restart *as it is recorded in the log you specified*. If it is available, the checkpoint to be used is identified, as is each outstanding UR together with the page sets it modified. Because database writes may be pending at restart, each page set with pending writes is also identified, as is the earliest log record required to complete those writes. If a log record associated with a UR is unavailable, the attribute INFO=PARTIAL is displayed, and the identification of page sets modified is incomplete for that UR.

DSN1PRNT (Service Aid)

DSN1PRNT allows you to print DB2 VSAM data sets that contain table spaces or index spaces. You can also print:

- Image copy data sets
- Sequential data sets.

Using DSN1PRNT, you can print hexadecimal dumps of DB2 data sets and databases. If you specify the FORMAT option, DSN1PRNT formats the data and indexes for any page that does not contain an error that would prevent formatting. If DSN1PRNT detects such an error, it prints an error message just before the page and dumps the page without formatting. Formatting resumes with the next page.

DSN1PRNT is especially useful when you want to identify the contents of a table space or index.

Environment

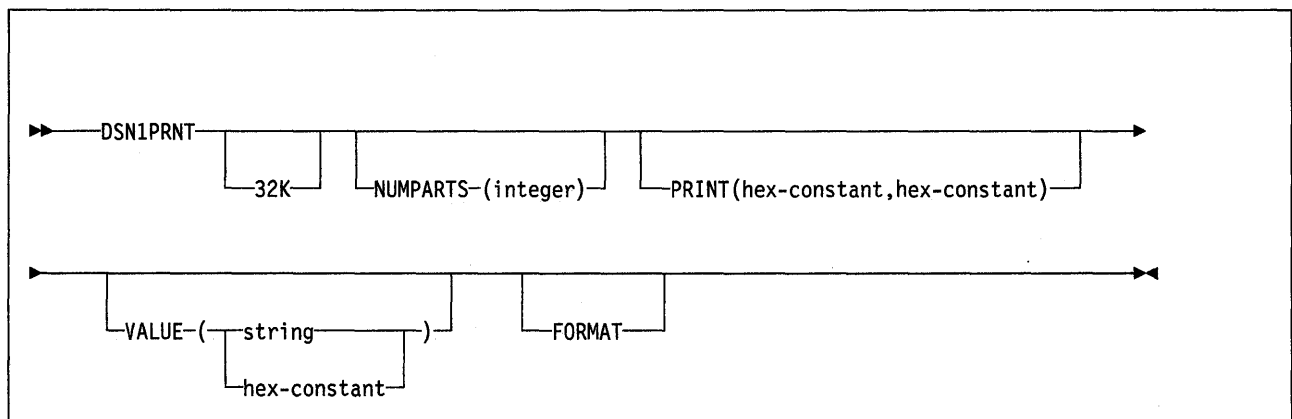
Run DSN1PRNT as an MVS job.

You can run DSN1PRNT even when the DB2 subsystem is not operational. If you choose to use DSN1LOGP when the DB2 subsystem is operational, you should be sure that the DB2 data sets that are to be printed are not currently allocated to DB2.

To make sure that a data set is not currently allocated to DB2, issue the DB2 -STOP DATABASE command, specifying the table space(s) and index(es) you want to print.

Authorization

None is required. However, if any of the data sets are RACF protected, the primary authorization ID designated by the process must have RACF authority. Also, if the data sets are password protected, the primary authorization ID designated by the process must have the appropriate VSAM passwords to execute this service aid.



Parameter Descriptions

Specify one or all of the parameters listed below on the EXEC card to run DSN1PRNT. If you specify more than one parameter:

- Separate them by commas (and no blanks).
- Specify them in any order.

32K

Specifies that the SYSUT1 data set has a 32K-byte page size. This option must be specified if the SYSUT1 data set has a 32K-byte page size. If you don't specify this parameter, the default page size of 4K bytes is assumed.

NUMPARTS(*integer*)

Specifies the number of partitions associated with the data set whose full image copy you are using as input to DSN1PRNT. Valid specifications range from 1 to 64. DSN1PRNT uses this value to help locate the first page in a range to be printed. If you omit NUMPARTS, or specify it as 0, DSN1PRNT will assume that your input file is not partitioned.

Note: DSN1PRNT cannot always validate the NUMPARTS parameter. If you specify it incorrectly, DSN1PRNT may print the wrong data sets, or return an error message indicating that an unexpected page number was encountered.

PRINT(*hexadecimal-constant,hexadecimal-constant*)

Causes the SYSUT1 data set to be printed in hexadecimal format on the SYSPRINT data set. You may enter the PRINT parameter with or without page range specifications. If you do not specify a range, all pages of the SYSUT1 will be printed. If you want to limit the range of pages printed, you may do so by indicating the beginning and ending page numbers with the PRINT parameter or, if you want to print a single page, by indicating only the beginning page. In either case, your range specifications must be from one to six hexadecimal digits in length.

The following example shows how you would code the PRINT parameter if you wanted to begin printing at page X'2F0' and to stop at page X'35C':

```
PRINT(2F0,35C)
```

VALUE

Causes each page of the input data set SYSUT1 to be scanned for the character string you specify in parentheses following the VALUE parameter. Each page that contains that character string will then be printed in SYSPRINT. The VALUE parameter may be specified in conjunction with any of the other DSN1PRNT parameters.

string

The string you specify may consist of from 1 to 20 alphameric characters.

hexadecimal constant

May consist of 2 to 40 hexadecimal characters. If hexadecimal characters are specified, they must be enclosed in single quotation marks.

If, for example, you want to search your input file for the string **12345**, your JCL might look like the following example:

```
//STEP1 EXEC PGM=DSN1PRNT,PARM='VALUE(12345)'
```

On the other hand, you might want to search for the equivalent hexadecimal character string, in which case your JCL might look like this:

```
//STEP1 EXEC PGM=DSN1PRNT,PARM='VALUE(''F1F2F3F4F5'')'
```

FORMAT

Causes the printed output to be formatted. Page control fields are identified and individual records are printed.

Usage Notes

DSN1PRNT uses two DD cards, each of which is listed and described below:

SYSPRINT Defines the data set that contains output messages from DSN1PRNT and all hexadecimal dump output.

SYSUT1 Defines the input data set. This data set can be a sequential data set or a VSAM data set. DSN1PRNT assumes that block size is a multiple of 4096 bytes (as is standard for DB2 data sets).

Disposition for this data set should be specified as OLD (DISP=OLD) to ensure that it is not in use by DB2. Disposition for this data set should be specified as SHR (DISP=SHR) only in circumstances where the DB2 -STOP DATABASE command does not work.

Note: The requested operation will take place only for the data set specified. If the input data set is a partitioned table space or index or if it is part of a linear table space that is larger than 2 gigabytes, care must be taken to ensure the correct data set is specified. For example, to print a page range in the second partition, the data set name must specify the second data set in the group of VSAM data sets comprising the table space (In other words, DSN=...A002).

Sample JCL for Using DSN1PRNT

Figure 30 is a sample of JCL used for running DSN1PRNT.

```
//jobname JOB acct info
//RUNPRNT EXEC PGM=DSN1PRNT,PARM='PRINT,FORMAT'
//STEPLIB DD DSN=PDS containing DSN1PRNT
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=DSNCAT.DSNDBC.DSNDB01.SYSUTIL.I0001.A001,DISP=SHR
```

Figure 30. Sample JCL for Running DSN1PRNT

Return Codes Issued During DSN1PRNT Processing

RC=0 Processing completed successfully.

RC=4 All pages have been processed, but some errors (for instance, warning from ACB open) have been detected.

RC=8 Processing did not complete successfully (for example, a VSAM GET error might have occurred).

LOAD (Utility)

The LOAD utility loads data into one or more tables in a table space or partition; LOAD can also replace the contents of a single partition, or of an entire table space. The LOAD DATA statement describes the data to be loaded and provides information needed for allocating resources. The loaded data is processed by any edit or validation routine associated with the table, and any field procedure associated with any column of the table.

Environment

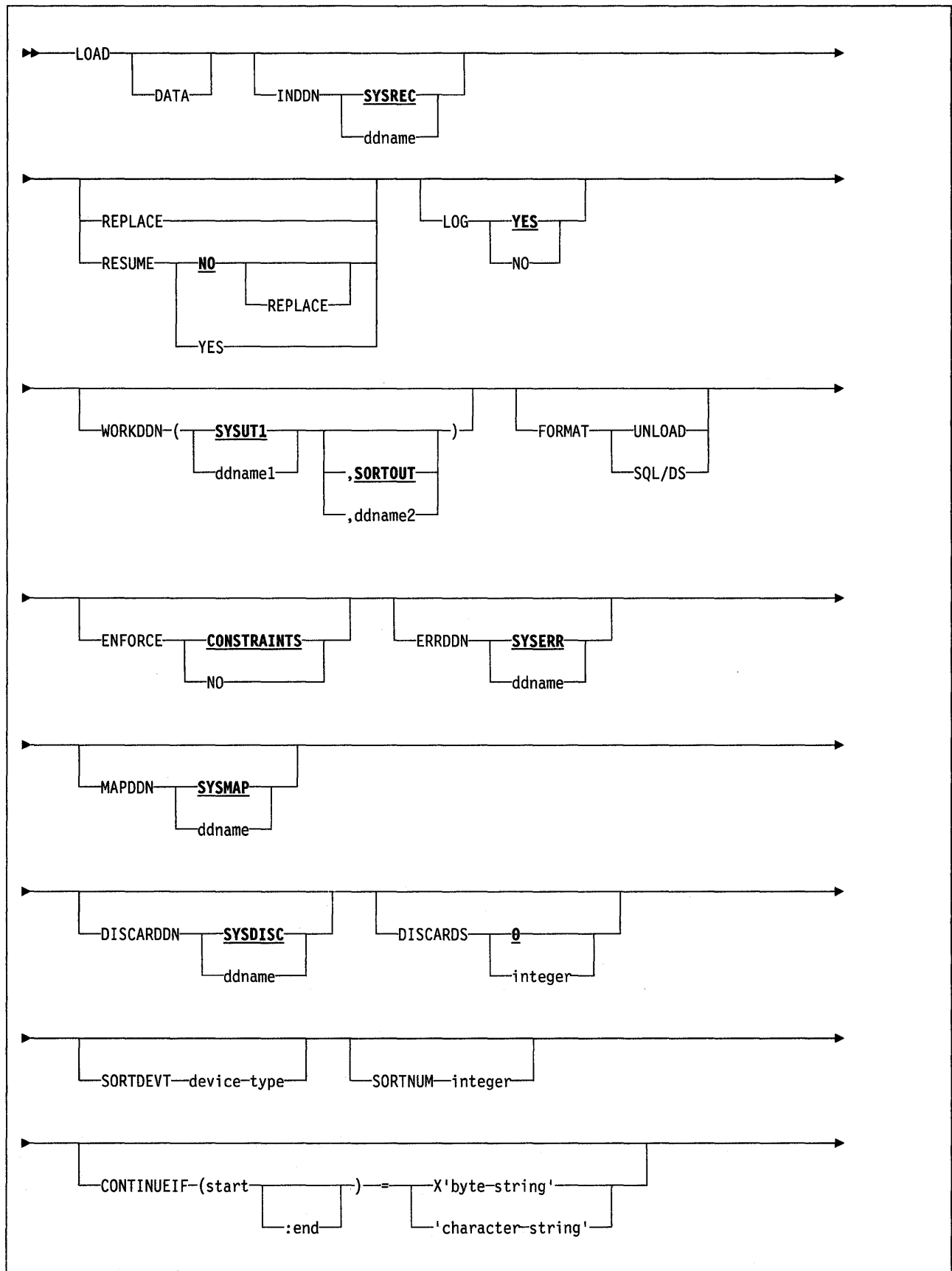
See “Chapter 3. Running DB2 Utilities” on page 151, for a description of ways to invoke DB2 utilities.

Authorization

To execute this utility, the privilege set defined below must include the LOAD privilege for the database containing the named table space. The LOAD privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following privileges:

- SYSADM authority
- DBADM or DBCTRL authority for the database.

The authorization required to invoke a utility job step is discussed under “Authorization” in the description of each utility.



Syntax (continued)



Keyword and Parameter Descriptions

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see “Utility Statement Parsing” on page 15.

DATA

Is used for clarity only. You identify the data selected for loading with *table-name* on the INTO TABLE option. See “INTO TABLE Option” on page 230 for a description of the statement.

INDDN *ddname*

Names the DD statement for the input data set. Record format for the input data set must be fixed or variable. The data set must be readable by the MVS BSAM access method.

ddname is the DD name. The **default** is **SYSREC**.

RESUME

Tells whether records are to be loaded into an empty or nonempty table space.

NO

Loads records into an empty table space. If the table space is not empty, and you have not used REPLACE, a message is issued and the utility job step terminates with a job step condition code of 8. The **default** is **NO**.

YES

Loads records into a nonempty table space. If the table space is empty, a warning message is issued, but the table space is loaded.

REPLACE

Tells whether the table space and all its indexes should be reset to empty before records are loaded. With this option, the newly loaded rows effectively replace all existing rows of all tables in the table space.

You cannot use REPLACE with the PART *integer* REPLACE option of INTO TABLE; you must either replace an entire table space, using the REPLACE option, or a single partition, using the PART *integer* REPLACE option of INTO TABLE. Also, you cannot use REPLACE with RESUME YES: Use RESUME NO, or omit the RESUME keyword.

If you specify a table space that is in the “recovery pending” status, REPLACE resets the restriction. If you specify a parent table space, its dependent table spaces are placed in the “check pending” status. Using REPLACE with the ENFORCE CONSTRAINTS option resets the “check pending” status.

LOG

Tells whether logging is to occur during the load process (RELOAD phase).

YES

Specifies normal logging during the load process. All records loaded are logged. The **default** is **YES**.

NO

Specifies no logging of data during the load process. This option sets the “copy-pending” restriction against the loaded table: The table cannot be updated until the restriction is removed. For ways to remove the restriction, see “Usage Notes” on page 239.

WORKDDN (ddname1,ddname2)

Names the DD statements for the temporary work file for sort input and the temporary work file for sort output. Temporary work files for sort input and output are required if the LOAD involves tables with indexes.

ddname1 is the DD name for the temporary work file for sort input. The **default** is **SYSUT1**.

ddname2 is the DD name for the temporary work file for sort output. The **default** is **SORTOUT**.

FORMAT

Identifies the format of the input record. If you use FORMAT, it uniquely determines the format of the input, and no field specifications are allowed in an INTO TABLE option. Follow FORMAT by either UNLOAD or SQL/DS.

If you omit FORMAT, the format of the input data is determined by the rules for field specifications described for the WHEN option of “Keyword and Parameter Descriptions for INTO TABLE” on page 232.

UNLOAD

Specifies that the input record format is compatible with the DB2 unload format. (The DB2 unload format is the result of a REORG UNLOAD(ONLY) option.) Input records that were unloaded by the REORG utility are loaded:

- Only into the tables from which they were unloaded; and
- Only if there is an INTO TABLE option to name each table. Any WHEN clause on that statement is ignored.

Input records that cannot be loaded are discarded.

If the DCB RECFM parameter is specified on the DD statement for the input data set, and the data set format has not been modified since the REORG UNLOAD (ONLY), the record format should be variable (RECFM=V).

SQL/DS

Specifies that the input record format is compatible with the SQL/DS unload format. The data type of a column in the table to be loaded must be the same as the data type of the corresponding column in the SQL/DS table.

If the SQL/DS input contains rows for more than one table, the WHEN clause of the INTO TABLE option tells which input records load into which DB2 table.

For information on the correct DCB parameters to specify on the DD statement for the input data set, refer to *SQL/Data System Data Base Services Utility for VM/System Product*.

LOAD cannot load SQL/DS strings that are longer than the DB2 limit. For information about DB2 limits, see Appendix, “DB2 Limits” on page 297.

SQL/DS data that has been unloaded to disk under VM/370 resides in a simulated OS/VS type data set with a record format of VBS. This must be taken into consideration when transferring the data to another system to be loaded into a DB2 table (for example, the VM/370 FILEDEF must define it as an OS/VS type data set). Processing it as a standard CMS file will

cause the block descriptor and record descriptor fields to be included as data in the records of the intermediate data set. The result will be that the SQL/DS record type field will be at the wrong offset within the records and LOAD will be unable to recognize them as valid SQL/DS input.

ENFORCE

Specifies whether or not LOAD is to enforce referential constraints.

CONSTRAINTS

Indicates that referential constraints are to be enforced. If LOAD detects a violation of referential constraints, it deletes, and issues a message identifying, the errant row.

NO

Indicates that referential constraints are not to be enforced. This option places the target table space in the "check pending" status. For ways to reset this status, see "Usage Notes" on page 239.

The **default** is **CONSTRAINTS**.

ERRDDN *ddname*

Names the DD statement for a work data set for error processing. Information about errors encountered during processing is stored in this data set. A SYSERR data set is required if you request discard processing.

ddname is the DD name. The **default** is **SYSERR**.

MAPDDN *ddname*

Names the DD statement for a work data set for error processing. Information about where input data sets are loaded is stored in this data set. A SYSMAP data set is *required* if you specify ENFORCE CONSTRAINTS and the tables have a referential relationship, or if you request discard processing when loading one or more tables that contain unique indexes.

ddname is the DD name. The **default** is **SYSMAP**.

DISCARDN *ddname*

Names the DD statement for a "discard data set," to hold copies of records that are not loaded (for example, if they contain conversion errors); the discard data set also holds copies of records loaded, then removed (due to unique index or referential integrity errors). The discard data set must be a sequential data set that can be written to by BSAM, with the same record format, record length, and block size as the input data set.

ddname is the DD name. The **default** is **SYSDISC**.

If you omit the DISCARDN option, the utility program saves discarded records only if there is a SYSDISC DD statement in the JCL input.

DISCARDS *integer*

Specifies the maximum number of source records to be written on the discard data set. *integer* may range from 0 to 2147483647. If the discard maximum is exceeded, LOAD abends. This allows you to either restart the job with a larger limit, or to -TERM the utility.

Note: Only records containing *primary* referential integrity errors are applied toward the discard limit. There is no limit on the number of records containing secondary errors.

DISCARDS 0 specifies that there is no maximum: The entire input data set can be discarded.

The **default** is **DISCARDS 0**.

SORTDEVT *device-type*

Names the device type for temporary data sets to be dynamically allocated by DFSORT. It can be any device type acceptable to the DYNALLOC parameter of the SORT or OPTION option for DFSORT as described in *DFSORT Application Programming: Guide*.

If you omit SORTDEVT and a sort is required, you must provide the DD statements that the sort program needs for the temporary data sets.

SORTNUM *integer*

Tells the number of temporary data sets to be dynamically allocated by the sort program.

If you omit SORTDEVT, SORTNUM is ignored. If you use SORTDEVT and omit SORTNUM, no value is passed to DFSORT; It is allowed to take its own default.

CONTINUEIF

Allows you to treat each input record as a portion of a larger record. After CONTINUEIF, write a condition in one of these forms:

```
(start:end) = X'byte-string'
(start:end) = 'character-string'
```

If the condition is true in any record, the next record is concatenated with it before loading takes place. You can concatenate any number of records into a larger record.

start:end

Are column numbers in the input record; the first column of the record is column 1. The two numbers tell the starting and ending columns of a continuation field in the input record.

Note: Other field position specifications (such as those for WHEN, POSITION, or NULLIF) refer to the field position within the final assembled load record, not the input record.

The continuation field is removed from the input record and is not part of the final load record.

If you omit *:end*, the length of the continuation field is taken as the length of the byte string or character string. If you use *:end*, and the length of the resulting continuation field is not the same as the length of the byte string or character string, the shorter one is padded. Character strings are padded with blanks. Hexadecimal strings are padded with zeros.

byte-string

is a string of hexadecimal digits. That value in the continuation field indicates that the next input record is a continuation of the current load record. Records with that value are concatenated until the value in the continuation field changes. For example, a specification could be

```
CONTINUEIF (72) = X'FF'
```

character-string

is a string of characters that has the same effect as X'*byte-string*'. For example, a specification could be

```
CONTINUEIF (99:100) = 'CC'
```

LOAD (Utility)

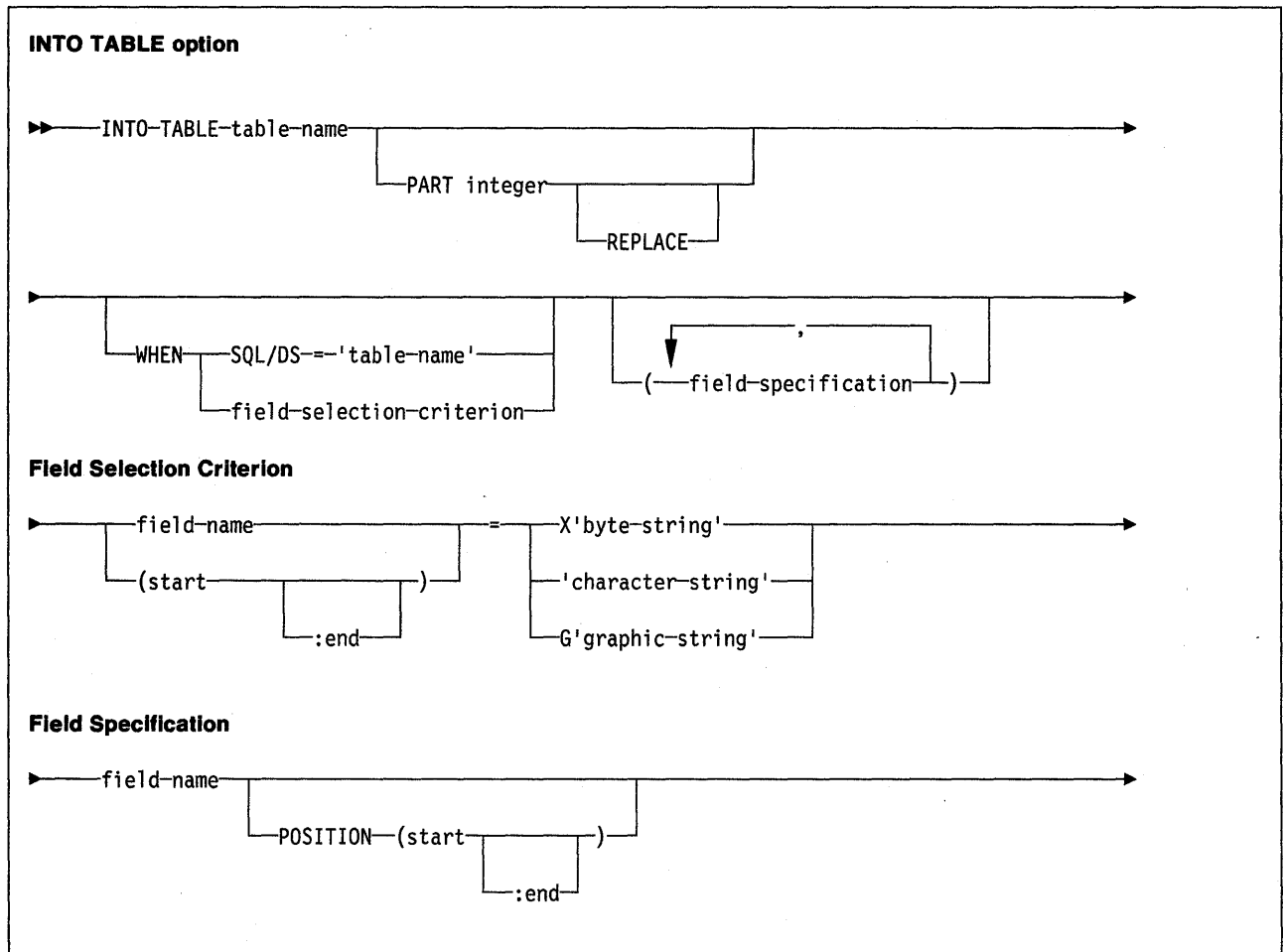
INTO TABLE Option

More than one table for each table space can be loaded with a single invocation of the LOAD utility. At least one INTO TABLE statement is required for each table to be loaded, to:

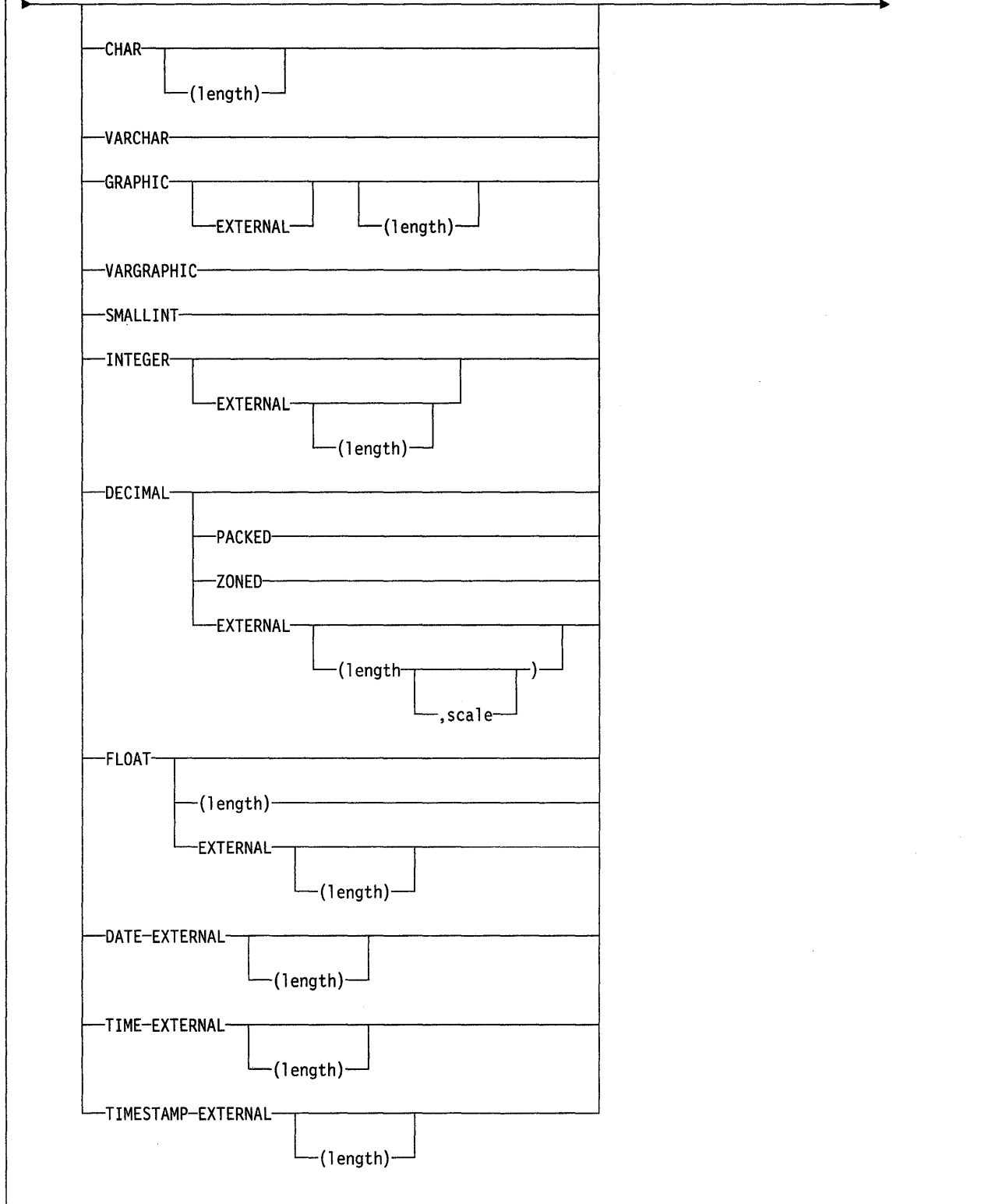
- Identify the table that is to be loaded.
- Describe fields within the input record.
- Define the format of the input data set.

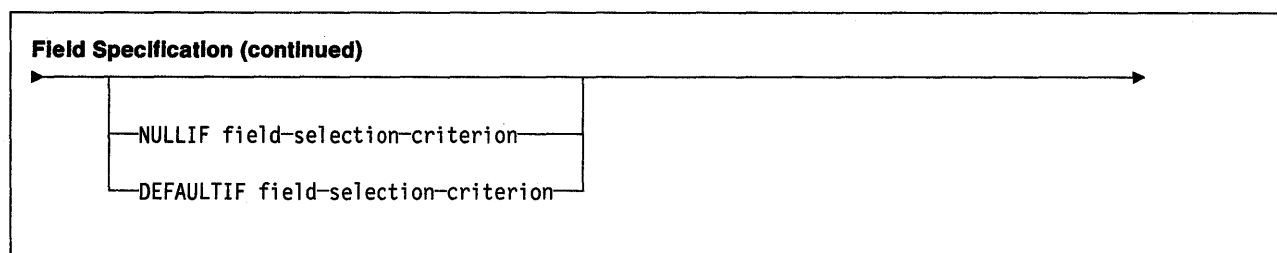
All tables named by INTO TABLE statements must belong to the same table space.

If the data is already in UNLOAD or SQL/DS format, and FORMAT UNLOAD or FORMAT SQL/DS is used on the LOAD statement, no field specifications are allowed.



Field Specification (continued)





Keyword and Parameter Descriptions for INTO TABLE

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see “Utility Statement Parsing” on page 15.

table-name

Is the name of a table to be loaded. The table must be described in the catalog and must not be a catalog table.

If the table name is not qualified by an *authid*, the *userid* of the invoker of the utility job step is used as the *authid-qualifier* of the table name.

Data from every load record in the data set is loaded into the table named, UNLESS:

- A WHEN clause is used and the data does not match the field selection criterion,
- the FORMAT UNLOAD option is used on the LOAD statement and the data comes from a table not named in an INTO TABLE statement,
- a certain partition is specified, and the data does not belong to that partition,
- any errors not generated by data conversion occur, or
- data conversion errors occur.

The following are optional.

PART *integer*

Is valid only for partitioned table spaces. *integer* is the number of the partition for which records are accepted for loading.

REPLACE

If specified, this option indicates that you want to replace only the contents of the partition cited by the PART option, rather than the entire table space.

You cannot use LOAD REPLACE with the PART *integer* REPLACE option of INTO TABLE. If you specify the REPLACE option, you must either replace an entire table space, using LOAD REPLACE, or a single partition, using the PART *integer* REPLACE option of INTO TABLE. You can, however, use PART *integer* REPLACE with RESUME YES.

WHEN

The WHEN clause tells which records in the input data set are to be loaded. If there is no WHEN clause (and if FORMAT UNLOAD was not used in the LOAD statement), *all* records in the input data set are loaded into the specified tables or partitions.

The option following WHEN describes a condition; input records that satisfy the condition are loaded. Input records that do not satisfy any WHEN clause of any INTO TABLE statement are written to the discard data set, if one is being used.

SQL/DS = 'table-name'

Is valid only when the FORMAT SQL/DS option is used on the LOAD statement. Here 'table-name', enclosed between apostrophes, is the SQL/DS name of a table that has been unloaded onto the SQL/DS unload data set. The table name after INTO TABLE tells which DB2 table the SQL/DS table is loaded into.

If there is no WHEN clause, input records from every SQL/DS table are loaded into the table named after INTO TABLE.

field-selection-criterion

Describes a field and a character constant. Only those records in which the field contains the specified constant are loaded into the table named after INTO TABLE.

A field in a selection criterion must:

- Contain a character or graphic string. No data type conversions are performed when the contents of the field in the input record are compared to a string constant.
- Start at the same byte offset in each assembled input record. If any record contains varying-length strings—stored with length fields—that *precede* the selection field, then they must be padded so the start of the selection field is always at the same offset.

The field and the constant need not be the same length. If they are not, the shorter of the two is padded before a comparison is made. Character and graphic strings are padded with blanks. Hexadecimal strings are padded with zeros.

field-name

Is the name of a field defined by a *field-specification*. If *field-name* is used, the start and end positions of the field are given by the POSITION option of the field specification.

(start:end)

start and *:end* are column numbers in the assembled load record; the first column of the record is column 1. The two numbers tell the starting and ending columns of a selection field in the load record.

If *:end* is not used, the field is assumed to have the same length as the constant.

X'byte-string'

Gives the constant as a string of hexadecimal digits. For example, write

```
WHEN (33:34) = X'FFFF'
```

'character-string'

Gives the constant as a string of characters. For example, write

```
WHEN DEPTNO = 'D11'
```

G'graphic-string'

Gives the constant as a string of double-byte characters. For example, write

```
WHEN (33:36) = G'<*>'
```

where "<" is the shift-out character, "*" is a double-byte character, and ">" is the shift-in character.

When the constant is specified in this form, and if the first byte of the input data is a shift-out character, it is ignored in making the comparison; if the last byte is a shift-out character, it is ignored.

field-specification

Describes the location, format, and null value identifier of the data to be loaded.

If NO field specifications are used:

- The fields in the input records are assumed to be in the same order as in the DB2 table.
- The formats are set by the FORMAT option on the LOAD statement, if that is used.
- Fixed strings in the input are assumed to be of fixed maximum length. VARCHAR and VARGRAPHIC fields must contain a valid 2-byte binary length field preceding the data; there cannot be intervening gaps between them and the fields that follow.
- Numeric data is assumed to be in the appropriate internal DB2 number representation.
- The NULLIF or DEFAULTIF options may not be used.

If any field specification is used for an input table, there must be a field specification for each field of the table that does not have a default value. Any field in the table with no corresponding field specification is loaded with its default value.

If any column in the output table does not have a field specification and is defined as NOT NULL, with no default, the utility job step is terminated.

field-name

Is the name of a field to be loaded. The name must be the name of a column in the table named after INTO TABLE.

The starting location of the field is given by the POSITION option. If POSITION is not used, the starting location is one column after the end of the previous field.

The length of the field is determined in one of the following ways, in the order listed:

1. If the field has data type VARCHAR or VARGRAPHIC, the length is assumed to be contained in a 2-byte binary field preceding the data. For VARCHAR fields, the length is in bytes; for VARGRAPHIC fields, it is in (double-byte) characters.
2. If *:end* is used in the POSITION option, the length is calculated from *start* and *end*. In that case, any length attribute after the CHAR, GRAPHIC, INTEGER, DECIMAL, or FLOAT specifications is ignored.
3. The length attribute on the CHAR, GRAPHIC, INTEGER, DECIMAL, or FLOAT specifications is used as the length.
4. The length is taken from the DB2 field description.

If a data type is not given for a field, its data type is taken to be the same as that of the column it is loaded into, as given in the DB2 table definition.

POSITION (*start:end*)

Tells where a field is in the assembled load record.

start and *end* are the locations of the first and last columns of the field; the first column of the record is column 1. The option may be omitted.

Column locations may be given as:

- An integer n , meaning an actual column number
- $*$, meaning one column after the end of the previous field
- $*+n$, where n is an integer, meaning n columns after the location specified by $*$

The POSITION option specification *cannot* be enclosed in parentheses; but the *start:end* description *must* be enclosed in parentheses, as the following example shows.

Valid	Invalid
POSITION (10:20)	POSITION ((10:20))

Data Types in a Field Specification: The data type of the field can be specified by any of the keywords that follow. Except for graphic fields, *length* is the length in bytes of the input field.

All numbers designated “EXTERNAL” are in the same format in the input records.

CHAR

CHAR (*length*)

For a fixed-length character string. The length of the string is determined from the POSITION specification or from *length*. You may also specify CHARACTER and CHARACTER(*length*).

DATE EXTERNAL

DATE EXTERNAL (*length*)

For a character string representation of a date. Length, if unspecified, is the length given by the LOCAL DATA LENGTH install option, or, if none was provided, defaults to ten bytes. If you specify a length, it must be within the range of 8 to 254 bytes.

Dates may be in any of the following formats. You may omit leading zeros for month and day.

- dd.mm.yyyy
- mm/dd/yyyy
- yyyy-mm-dd
- any local format defined by your site at the time DB2 was installed

DECIMAL

DECIMAL *internal*

DECIMAL PACKED

For a number of the form $ddd...ds$, where d is a decimal digit represented by four bits, and s is a four-bit sign value. (The plus sign (+) is represented by A, C, E, or F and the minus sign (-) is represented by B or D.) The maximum number of ds is the same as the maximum number of digits allowed in the SQL definition.

DECIMAL ZONED

For a number of the form $znznzn...z/sn$, where n is a decimal digit represented by the right four bits of a byte (called the *numeric bits*); z is that digit's zone, represented by the left four bits; and s is the rightmost byte of the decimal operand, and can be treated as a zone or as the sign value for that digit. (The plus sign (+) is represented by A, C, E, or F and the minus sign (-) is represented by B or D.) The maximum number of zns is the same as the maximum number of digits allowed in the SQL definition.

You may also specify DEC.

DECIMAL EXTERNAL(*length,scale*)*length*

Overall length of the input field in bytes.

scale

Specifies the number of digits to the right of the decimal point. Must be an integer greater than or equal to 0, and may be greater than *length*.

If *scale* value is greater than *length*, or the number of digits provided is less than the *scale* specified, the input number is padded on the left with zeros until the decimal point position is reached.

If *scale* is greater than the target scale, the source scale locates the implied decimal position. All fractional digits greater than target scale are truncated.

If *scale* value is specified and the target column is small integer or integer the decimal portion of the input number is ignored. If a decimal point is physically present, its position will override the field specification of *scale*.

If a *scale* specification is not specified the **default** is zero.

You may also specify DEC EXTERNAL and DEC EXTERNAL (*length*).

FLOAT**FLOAT EXTERNAL****FLOAT *internal***

For a 64-bit floating point number where,

- Bit 0 represents a sign (0 = +, and 1 = -).
- Bits 1-7 represent an exponent in excess-64 notation.
- Bits 8-64 represent a mantissa.

FLOAT(*length*) *internal***FLOAT(*length*)****FLOAT EXTERNAL (*length*)**

For either a 64-bit floating point number, or a 32-bit floating point number. If *length* is between 1 and 21 inclusive, the number is 32 bits in the following format:

- Bit 0 represents a sign (0 = +, and 1 = -).
- Bits 1-7 represent an exponent in excess-64 notation.
- Bits 8-31 represent a mantissa.

If *length* is between 22 and 53 inclusive, the number is 64 bits in the following format:

- Bit 0 represents a sign (0 = +, and 1 = -).
- Bits 1-7 represent an exponent in excess-64 notation.
- Bits 8-63 represent a mantissa.

You may also specify REAL for single precision floating point, and DOUBLE PRECISION for double precision floating point.

GRAPHIC**GRAPHIC EXTERNAL****GRAPHIC (*length*)****GRAPHIC EXTERNAL (*length*)**

For a graphic field.

You can specify both *start* and *end* for the field, whether you use GRAPHIC, or GRAPHIC EXTERNAL.

If you use GRAPHIC, the input data must not contain shift characters; *start* and *end* should indicate the starting and ending positions of the data itself.

If you use GRAPHIC EXTERNAL, the input data must contain a shift-out character in the starting position, and a shift-in character in the ending position. Aside from the shift characters, there must be an even number of bytes in the field. The first byte of any pair must not be a shift character.

In both cases, *length* is a number of double-byte characters. *length* for GRAPHIC EXTERNAL does not include the bytes of shift characters. The length of the field in bytes is taken as twice the value of *length*.

For example, let *** represent 3 double-byte characters, and let < and > represent shift-out and shift-in characters. Then:

To describe <***>, use either POS(1:8) GRAPHIC EXTERNAL or POS(1) GRAPHIC EXTERNAL(3).

To describe ***, use either POS(1:6) GRAPHIC or POS(1) GRAPHIC(3).

A GRAPHIC field described in this way may not be named in a field selection criterion.

INTEGER

INTEGER *internal*

INTEGER EXTERNAL

INTEGER EXTERNAL (*length*)

Specifies a fullword binary number. Negative numbers are in two's-complement notation. You may also specify INT. You may also specify INT EXTERNAL and INT EXTERNAL (*length*).

SMALLINT

For a halfword binary number. Negative numbers are in two's-complement notation.

TIME EXTERNAL

TIME EXTERNAL (*length*)

For a character string representation of a time. Length, if unspecified, is the length given by the LOCAL TIME LENGTH install option, or, if none was provided, defaults to eight bytes. If you specify a length, it must be within the range of 4 to 254 bytes.

Times may be in any of the following formats.

- hh.mm.ss
- hh:mm AM
- hh:mm PM
- hh:mm:ss
- any local format defined by your site at the time DB2 was installed

Note: You may omit the *mm* portion of the *hh:mm AM* and *hh:mm PM* formats if *mm* is equal to *00*. For example, *5 PM* is a valid time, and may be used instead of *5:00 PM*

TIMESTAMP EXTERNAL

TIMESTAMP EXTERNAL (*length*)

For a character string representation of a time. Length, if unspecified, defaults to 26 bytes. If you specify a length, it must be within the range of 16 to 86 bytes.

Timestamps may be in either of the following formats. Note that *nnnnnn* represents the number of microseconds, and can be from zero to six digits.

You may omit leading zeros from the month, day, or hour parts of the timestamp; you may omit trailing zeros from the microseconds part of the timestamp.

- yyyy-mm-dd-hh.mm.ss
- yyyy-mm-dd-hh.mm.ss.nnnnnn

See Chapter 1 of *SQL Reference* for more information about the timestamp data type.

VARCHAR

For a character field of varying-length. The length in bytes must be given in a 2-byte binary field preceding the data. (The length given there does NOT include the 2-byte field itself.) The length field must start in the column named as *start* in the POSITION option. *:end*, if used, is ignored.

VARGRAPHIC

For a graphic field of varying-length. The length, in double-byte characters, must be given in a 2-byte binary field preceding the data. (The length given there does NOT include the 2-byte field itself.) The length field must start in the column named as *start* in the POSITION option. *:end*, if used, is ignored.

VARGRAPHIC input data must not contain shift characters.

NULLIF *field-selection-criterion*

Describes a condition that causes the DB2 column to be loaded with NULL. The *field-selection-criterion* can be written with the same options as described on page 233. If the contents of the NULLIF field match the character constant given, the field named in *field-specification* is loaded with NULL.

If the NULLIF field is defined by the name of a VARCHAR or VARGRAPHIC field, the length of the field is taken from the 2-byte binary field that appears before the data portion of the VARCHAR or VARGRAPHIC field.

The fact that a field in the output table is loaded with NULL does not change the format or function of the corresponding field in the input record. The input field can still be used in a field selection criterion. For example, with the field specification:

```
(FIELD1 POSITION(*) CHAR(4)
ELD2 POSITION(*) CHAR(3) NULLIF(FIELD1='SKIP')
FIELD3 POSITION(*) CHAR(5))
```

and the source record:

```
SKIP  FLD03
```

the record would be loaded so that:

```
FIELD1 has the value 'NULL'
FIELD2 is NULL (not ' ' as in the source record)
FIELD3 has the value 'FLD03'
```

DEFAULTIF *field-selection-criterion*

Describes a condition that causes the DB2 column to be loaded with its default value. The *field-selection-criterion* can be written with the same options as described on page 233. If the contents of the DEFAULTIF field match the character constant given, the field named in *field-specification* is loaded with its default value.

If the DEFAULTIF field is defined by the name of a VARCHAR or VARGRAPHIC field, the length of the field is taken from the 2-byte binary field that appears before the data portion of the VARCHAR or VARGRAPHIC field.

Data Conversion

The LOAD utility converts data between compatible data types: CHAR, VARCHAR, and LONG VARCHAR for character data; SMALLINT, INTEGER, DECIMAL, and FLOAT for numeric data. Figure 31 on page 240 identifies the allowable data conversions by the letters X and D. D indicates the default used when the input data type is not specified in a field specification of the INTO TABLE statement.

Conversion errors cause LOAD:

- to abend, if there is no DISCARDS processing, or
- to discard the record and to continue, if there is DISCARDS processing.

Truncation of the decimal part of numeric data is not considered a conversion error.

Output

Output from LOAD DATA consists of one or more of the following:

- A loaded table space or partition
- A discard file of rejected records
- A summary report of errors encountered during processing, generated only if you specify ENFORCE CONSTRAINTS or if the LOAD involves unique indexes.

When loading into an unsegmented table space, LOAD leaves one free page after reaching the FREEPAGE limit, regardless of whether the records loaded belong to the same or different tables.

When loading into a segmented table space, LOAD leaves free pages, and free space on each page, in accordance with the current values of the FREEPAGE and PCTFREE parameters. (Those values may be set by the CREATE TABLESPACE, ALTER TABLESPACE, CREATE INDEX, or ALTER INDEX statements). LOAD leaves one free page after reaching the FREEPAGE limit for each table in the table space. When loading into a segmented table space, sort your data by table to ensure that the data is loaded in the best physical organization.

Usage Notes

Actions Required when LOAD Completes: After a load has completed:

- **If you have used LOG YES**, consider taking a full image copy of the loaded table space or partition to reduce the processing time of later recovery operations. If you also specified RESUME NO, indicating that this is the first load into the table space, we recommend that you take multiple full image copies so fallback processing will be possible during recovery.
- **If you have used LOG NO**, LOAD places the loaded table space or partition in the “copy pending” status. This status prevents any update by SQL or an application program. See “Partial Recovery” on page 264 for information on resetting the copy pending status.
- **If you have used ENFORCE NO**, LOAD places the target table space in the check pending status. This indicates that rows in the table space have not been checked for violations of referential constraints. See “CHECK (Utility)” on page 178 for information on resetting the check pending status.
- **If a parent table space is replaced**, all dependent table spaces of the table space being loaded are placed in the check pending status. See “CHECK (Utility)” on page 178 for information on resetting the check pending status.

LOAD (Utility)

Input	SMALLINT	INTEGER	DECIMAL	FLOAT	CHAR	VARCHAR	LONGCHAR	GRAPHIC	VARGRAPHIC	LVARCHAR	DATE	TIME	TIMESTAMP
SMALLINT	D	X	X	X									
INTEGER	X	D	X	X									
DECIMAL	X	X	D	X									
FLOAT	X	X	X	D									
CHAR					D	X	X						
VARCHAR					X	D	D						
GRAPHIC								D	X	X			
VARGRAPHIC								X	D	D			
DATE EXTERNAL											D		
TIME EXTERNAL												D	
TIMESTAMP EXTERNAL											X	X	D

Figure 31. Data Conversions

- Use the RUNSTATS utility so that the DB2 catalog statistics take into account the newly loaded data, and SQL paths can be selected with accurate information.
- Rebind any application plans that depend on the loaded tables, to update the path selection of any embedded SQL statements.

Data Sets: During execution of the LOAD utility, several non-DB2 sequential data sets may be needed.

Input Data Set: The input data set (identified by the DD statement named by the INDDN option) must be a sequential data set that is readable by BSAM.

Discard Data Set: The discard data set (identified by the DD statement named by the DISCARD option) must be a sequential data set that is readable by BSAM, with the same record format, record length, and block size as the input data set.

Work Data Set: Use the following table to calculate the size of work data sets for LOAD.

Work Data Set	Size	Overriding Default Name
SYSUT1	Simple table space: max (k,e) Partitioned or segmented table space: max (k,e,m)	WORKDDN (newut1, SORTOUT)
SORTOUT	max (k,e)	WORKDDN (SYSUT1, newout)
SYSERR	e	ERRDDN (newerr)
SYSMAP	Simple table space or discard processing: m Partitioned or segmented table space without discard processing: max (m,e)	MAPDDN (newerr)
SYSDISC	Same size as input data set	DISCARDN (newdisc)

Figure 32. Work Data Set Calculation

k = key calculation
e = error calculation
m = map calculation

- key calculation

(length of longest index or foreign key + 12) • (number of foreign keys extracted.
To calculated number of keys extracted:

- Count 1 for each index.
- Count 1 for each foreign key that is not exactly indexed.
- For each foreign key that is exactly indexed:
 - Count 0 for the first relationship in which the foreign key participates.
 - Count 1 for subsequent relationships in which the foreign key participates (if any).
- Multiply by the number of rows to be loaded.

- map calculation

The data set must be large enough to accommodate 1 map entry (length = 16 bytes) per table row produced by the LOAD job.

- error calculation

The data set must be large enough to accommodate 1 error entry (length = 80 bytes) per defect detected by LOAD (for example, conversion errors, unique index violations, violations of referential constraints).

Restarting the LOAD Utility: You can restart LOAD at its last commit point or at the beginning of the phase during which operation ceased. The phases that have completed are identified in LOAD output messages; the specific phase during which operation stopped can be identified with the -DISPLAY command. For instructions on restarting a utility job, see "Using the DSNU CLIST in TSO" on page 160.

The roles and restart characteristics of each LOAD phase are discussed below:

UTILINIT During this phase, the utility is set up and initialized.

Both RESTART(PHASE) and RESTART(CURRENT) restart the utility from its beginning. The same amount of processing time is used as if the utility had been -TERMed and resubmitted.

RELOAD During this phase, one pass through the sequential input data set is made. All record types are loaded and temporary file records are written for indexes and foreign keys. Internal commits are taken to provide commit points at which to restart should operation halt in this phase.

RESTART(CURRENT) restarts LOAD processing at an internal commit point. This saves the processing time it took to reach that commit point. RESTART(PHASE), however, removes all data records loaded before operation halted, and then reapplies the input data. This saves no processing time. You must empty all work data sets before requesting RESTART(PHASE).

It is not recommended that you restart during RELOAD phase if you specified SYSREC DD *. This prevents internal commits from being taken, and RESTART(CURRENT) will perform like RESTART(PHASE), except with no data backout. It is also not recommended that you restart if your SYSREC input consists of multiple, concatenated data sets. In this case, you may specify RESTART(CURRENT) only if no more than the first data set has been processed; else, restarting may cause duplicate data to be loaded.

SORT If there are indexes or foreign keys, this phase is executed to sort the temporary file records before creating the indexes or validating referential constraints.

Both RESTART(PHASE) and RESTART(CURRENT) restart from the beginning of the phase.

BUILD Indexes are created from temporary file records for all indexes defined on the loaded tables, and duplicate keys are detected.

Both RESTART(PHASE) and RESTART(CURRENT) restart from the beginning of the phase. You cannot restart during BUILD phase if you use RESUME YES.

INDEXVAL Unique index violations, if any, are corrected from the information in SYSERR.

RESTART(CURRENT) restarts processing at the last internal commit point.

ENFORCE Referential constraints are checked, and violations are corrected. Information about violations of referential constraints are stored in SYSERR.

RESTART(CURRENT) restarts processing at the last internal commit point.

DISCARD Records causing errors are copied from the input data set to the discard data set.

RESTART(CURRENT) restarts processing at the last internal commit point.

REPORT If you specified ENFORCE CONSTRAINT, or if load index validation is performed, a summary report is generated and sent to SYSPRINT.

RESTART(CURRENT) restarts processing at the last internal commit point.

UTILTERM Final cleanup is performed.

After Terminating a LOAD Job: If LOAD is terminated by the -TERM UTILITY command during the reload phase, the records are not erased. The table space and indexes are placed in recovery pending status; see “Recovery Pending Status” on page 264 for information about resetting this status. and if the job was run with the LOG option set to YES, then the records that have been loaded are erased. You can submit the LOAD job with the same options again, to rerun from the beginning. But the time required to back out loaded records can be significant, and, during that time, the terminal that issued the -TERM command will be occupied.

If LOAD is terminated by -TERM UTILITY during the sort or build phases, then the indexes not yet built are placed in the recovery pending status. See “Recovery Pending Status” on page 264 for information on resetting the recovery pending status for these indexes.

Phase	Effect on Pending Status
Unload	No effect.
Reload	Places table space in recovery pending status, then resets the status. Places indexes in recovery pending status. Places table space in copy pending status. Places table space in check pending status.
Build	Resets recovery pending status for indexes.
Indexval	Resets recovery pending status for indexes.
Enforce	Resets check pending status for table space.

Figure 33. LOAD Phases and Pending Status'

DFSORT Messages: The LOAD utility job step must contain a UTPRINT DD statement, to define a destination for messages issued by the DFSORT component during the SORT phase of LOAD. The default DD statement used by DB2I and the %DSNU CLIST is:

LOAD (Utility)

```
//UTPRINT DD SYSOUT=A
```

Data Set Definition: If the table space to be loaded is associated with a storage group, DB2 will automatically define the data sets for it. If the table space is NOT associated with a storage group, you must define the necessary data sets before invoking LOAD.

Field Procedures: Any field procedure associated with a column of a table being loaded is invoked to encode the data before it is loaded. The field procedures for all columns are invoked before any edit or validation procedure for the row.

But any field specification that describes the data is checked before a field procedure is invoked. That is, the field specification must describe the data as it appears in the input record.

Examples

Example 1: Load data from the data set named by the EMPLDS DD statement into the EMP table.

```
LOAD DATA INDDN EMPLDS  
  INTO TABLE DSN8210.EMP
```

Example 2: Load data from the data set named by the EMPLDS DD statement into the DSN8210.EMP and SMITH.EMPEMPL tables.

```
LOAD DATA INDDN EMPLDS  
  INTO TABLE DSN8210.EMP  
  INTO TABLE SMITH.EMPEMPL
```

Example 3: Load data from the data set named by the EMPLDS DD statement into the EMP table. Load only from source input records which begin with 'LKA'.

```
LOAD DATA INDDN EMPLDS  
  INTO TABLE DSN8210.EMP  
  WHEN (1:3)='LKA'
```

Example 4: The data from the sequential data set identified by the SYSREC DD statement is selectively loaded into the DSN8210.DEPT table whenever positions 1 through 3 contain the value: 'LKA'. The table space need not be empty for loading to proceed.

For each source record that has 'LKA' in its first three positions:

- The characters in positions 7 through 9 are loaded into the DEPTNO column.
- The characters in positions 10 through 35 are loaded into the DEPTNAME VARCHAR column. The binary length fields of the DEPTNAME column will contain 26.
- The characters in positions 36 through 41 are loaded into the MGRNO column.
- Characters in positions 42 through 44 are loaded into the ADMRDEPT column. Processing is terminated if there is a source input record with a length less than 44, because all the columns have the NOT NULL attribute.


```

LOAD DATA
RESUME YES
INTO TABLE DSN8210.DEPT WHEN (1:3)='LKA'
(DEPTNO POSITION (7:9) CHAR,
DEPTNAME POSITION (10:35) CHAR,
MGRNO POSITION (36:41) CHAR,
ADMRDEPT POSITION (42:44) CHAR)

```

Example 5: Data from the sequential data set identified by the SYSRECPJ DD statement is selectively loaded into the DSN8210.PROJ table. The table space containing the DSN8210.PROJ table is currently empty, because the RESUME YES option was not specified.

For each source input record, data is loaded into the named columns (that is, PROJNO, PROJNAME, DEPNO, ..., etc.) to form a table row. Any other columns in a DSN8210.PROJ row are set to NULL.

Starting positions of the fields in the sequential data set are defined by the field specification POSITION options. The ending position of the fields in the sequential data set are implicitly defined either by the length specification of the data type options (CHAR length) or by the length specification of the external numeric data type (LENGTH).

The numeric data represented in SQL constant format (EXTERNAL format); it is converted to the correct internal format by the LOAD process and placed in the indicated column names. The two dates are assumed to be represented by 8 digits and 2 separator characters, as in the USA format (for example, 11/15/1987). The length of the date fields is given as 10 explicitly, though in many cases it would default to the same value.

```

LOAD DATA INDDN(SYSRECPJ)
INTO TABLE DSN8210.PROJ
(PROJNO POSITION (1) CHAR(6),
PROJNAME POSITION (8) CHAR(22),
DEPTNO POSITION (31) CHAR(3),
RESPEMP POSITION (35) CHAR(6),
PRSTAFF POSITION (42) DECIMAL EXTERNAL(5),
PRSTDATE POSITION (48) DATE EXTERNAL(10),
PRENDATE POSITION (59) DATE EXTERNAL(10),
MAJPROJ POSITION (70) CHAR(6))

```

Example 6: Data from the sequential data set named by the SYSRECOV DD statement is assembled and selectively loaded into the DSN8210.TOPTVAL table. The table space that contains DSN8210.TOPTVAL is currently empty because the RESUME YES option is not specified.

Fields destined for columns in the same table row can span more than one source record. Source records having fields containing columns that belong to the same row as the next source record all have an X in column 72 (that is, CONTINUEIF(72:72)='X').

For each assembled source record, fields are loaded into the DSN8210.TOPTVAL table columns (that is, MAJSYS, ACTION, OBJECT..., DSPINDEX) to form a table row. Any columns not mentioned are set to NULL.

The starting positions of the fields in the assembled source record input are given in the POSITION option. Starting positions are numbered from the first column of

LOAD (Utility)

the internally assembled input record, not from the start of the source records in the sequential data set. The ending positions are defined by the character string lengths given with the input data type.

No conversions are required to load the source character strings into their designated columns, which are also defined to be fixed character strings. However, because columns INFOTXT, HELPTXT, and PFKTXT are defined as 79 characters in length and the strings being loaded are 71 characters in length, these strings will be padded with blanks as they are loaded.

```
LOAD DATA INDDN(SYSRECOV) CONTINUEIF(72:72)='X'  
  INTO TABLE DSN8210.TOPTVAL  
  (MAJSYS POSITION (2) CHAR(1),  
   ACTION POSITION (4) CHAR(1),  
   OBJECT POSITION (6) CHAR(2),  
   SRCHCRIT POSITION (9) CHAR(2),  
   SCRTYPE POSITION (12) CHAR(1),  
   HEADTXT POSITION (80) CHAR(50),  
   SELTXT POSITION (159) CHAR(50),  
   INFOTXT POSITION (238) CHAR(71),  
   HELPTXT POSITION (317) CHAR(71),  
   PFKTXT POSITION (396) CHAR(71),  
   DSPINDEX POSITION (475) CHAR(2))
```

Example 7: Data from the sequential data set identified by the SYSREC DD statement is loaded into the DSN8210.PROJ. table. Referential constraints are enforced on data added. Output consists of a summary report of violations of referential constraints, and all records causing these violations are placed in the SYSDISC discard data set.

```
LOAD DATA INDDN(SYSREC) CONTINUEIF(72:72)='X'  
  RESUME YES  
  ENFORCE CONSTRAINTS  
  INTO TABLE DSN8210.PROJ  
  (PROJNO POSITION (1) CHAR (6),  
   PROJNAME POSITION (8) VARCHAR,  
   DEPTNO POSITION (33) CHAR (3),  
   RESPEMP POSITION (37) CHAR (6),  
   PRSTAFF POSITION (44) DECIMAL EXTERNAL (5),  
   PRSTDATE POSITION (50) DATE EXTERNAL,  
   PRENDATE POSITION (61) DATE EXTERNAL,  
   MAJPROJ POSITION (80) CHAR (6) NULLIF(MAJPROJ=' '))
```

MERGETCOPY (Utility)

The MERGETCOPY utility can merge several incremental copies of a table space to make one incremental copy, and can merge incremental copies with a full image copy to make a new full image copy.

Environment

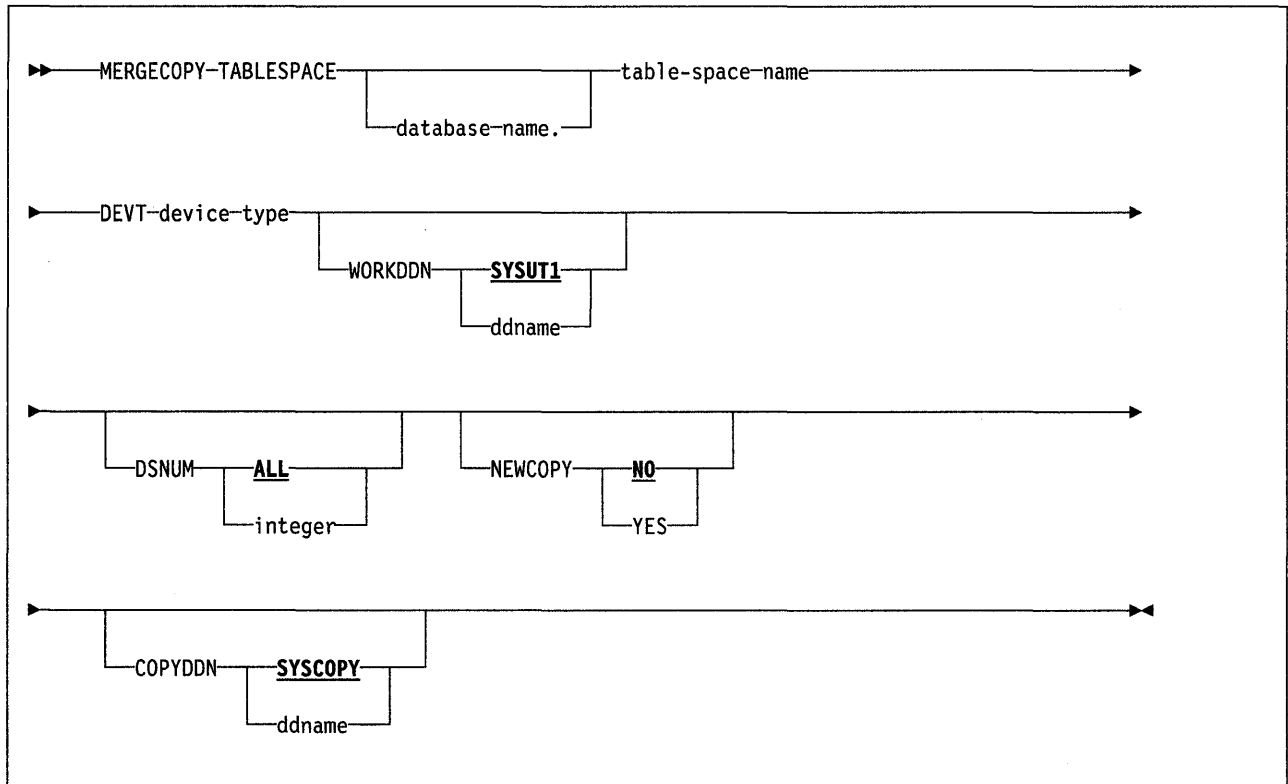
See "Chapter 3. Running DB2 Utilities" on page 151, for an explanation of ways to invoke DB2 utilities.

Authorization

To execute this utility, the privilege set defined below must include the IMAGCOPY privilege for the database containing the named table space. The IMAGCOPY privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following privileges:

- SYSADM authority
- DBADM, DBCTRL, DBMAINT authorities for a database.

If the utility is being executed on a table space in database DSNDB01 or DSNDB06, the privilege set must include either the SYSADM or SYSOPR authority defined when DB2 was installed.



Keyword and Parameter Descriptions

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see "Utility Statement Parsing" on page 15.

TABLESPACE

Names the table space (and, optionally, the data base it belongs to) that is to be copied.

database-name

Is the name of the database the table space belongs to. The **default** is **DSNDB04**.

tablespace-name

Is the name of the table space whose incremental image copies are to be merged.

DEVT *device-type*

Names the device type for the merged output. *device-type* must be the same as on the DEVT option of the COPY statement that created the data sets. The device type provides the value of the UNIT parameter in the DSNU CLIST.

The following are optional.

WORKDDN *ddname*

Names a DD statement for a temporary data set, to be used for intermediate merged output. *ddname* is the DD name. The **default** is **SYSUT1**.

Use the WORKDDN option if you are not able to allocate enough data sets to execute MERGECOPY; in that case, a temporary data set is used to hold intermediate output. (For instance, if you need ten data sets and can only allocate five, use WORKDDN.) If you omit the WORKDDN option, it is possible that only some of the image copy data sets will be merged. When MERGECOPY has ended, a message is issued that tells the number of data sets that exist and the number of data sets that have been merged. To continue the merge, repeat MERGECOPY with a new output data set.

DSNUM

Identifies a partition or data set, within the table space, that is to be merged; or it merges the entire table space.

ALL

Merges the entire table space. The **default** is **ALL**.

integer

Is the number of a partition or data set to be merged.

For a partitioned table space, the integer is its partition number.

For a nonpartitioned table space, find the integer at the end of the data set name as cataloged in the VSAM catalog. The data set name has this format:

catname.DSNDB*x*.*dbname*.*tsname*.I0001.*An*

where:

catname = VSAM catalog name or alias
x = C or D
dbname = database name
tsname = table space name
n = data set integer

Note: If image copies were taken by data set (rather than by table space), then MERGECOPY must use the copies by data set.

NEWCOPY

Tells whether incremental image copies are to be merged with the full image copy or not.

NO

Merges incremental image copies into a single incremental image copy, but does NOT merge them with the full image copy. The **default** is NO.

YES

Merges all incremental image copies with the full image copy to form a new full image copy.

COPYDDN *ddname*

Names a DD statement for the output image copy data set. *ddname* is the DD name. The **default** is SYSCOPY.

Output

Output from the MERGECOPY utility consists of one of the following:

- A new single incremental image copy
- A new full image copy.

If NEWCOPY is YES, the utility inserts an entry for the new full image copy into the SYSIBM.SYSCOPY catalog table.

If NEWCOPY is NO, the utility replaces the SYSIBM.SYSCOPY records of the incremental image copies that were merged with an entry for the new incremental image copy.

In either case, if all of the input data sets could not be allocated, and you did not specify a temporary work data set (WORKDDN), the utility performs a partial merge. The SYSIBM.SYSCOPY records for the merged image copies are then deleted.

Usage Notes

There is a trade-off between the speed of recovery and the frequency of execution of the MERGECOPY utility. If recovery becomes necessary, it will be much faster if MERGECOPY has been executed.

BLKSIZE parameter for output: You may specify a block size for the output by using the BLKSIZE parameter on the DD statement for the output data set. Valid block sizes are 4K, 8K, and 16K bytes; the default is 16K bytes.

Restart phases: You can restart a MERGECOPY utility job at the beginning of any of the phases listed below. For instructions on restarting a utility job, see “Chapter 3. Running DB2 Utilities” on page 151.

- UTILINIT: initialization and setup
- MERGECOPY: merging
- UTILTERM: cleanup

MERGECOPY (Utility)

Example

Example: Merge all existing incremental image copies of table space DSN8S21D with the last full image copy.

```
MERGECOPY TABLESPACE DSN8D21A.DSN8S21D  
NEWCOPY YES  
DEVT SYSDA
```

MODIFY (Utility)

The MODIFY utility deletes records of unwanted copies from the SYSIBM.SYSCOPY catalog table and records of related log records from the SYSIBM.SYSLGRNG directory table. You can remove records that were written before a specific date or you can remove records of a specific age. You can delete records for an entire table space, a data set, or pages within an error range.

Environment

See "Chapter 3. Running DB2 Utilities" on page 151, for an explanation of ways to invoke DB2 utilities.

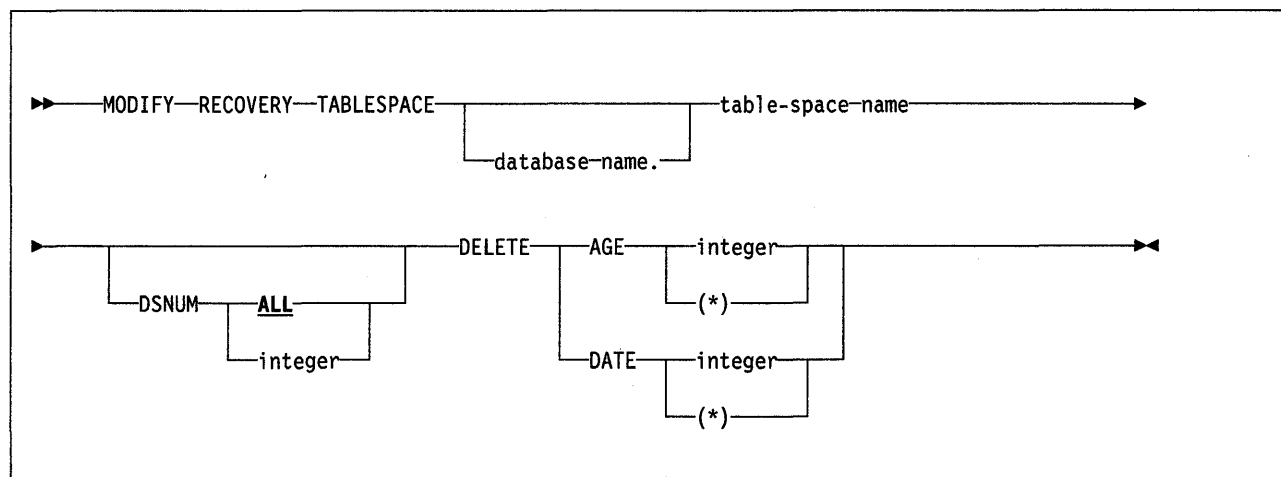
Authorization

To execute this utility, the privilege set defined below must include the IMAGCOPY privilege for the database containing the named table space. The IMAGCOPY privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following privileges:

- SYSADM authority
- DBADM, DBCTRL, DBMAINT authorities for a database.

If the utility is being executed on a table space in database DSNDB01 or DSNDB06, the privilege set must include either the SYSADM or SYSOPR authority defined when DB2 was installed.

Syntax



Keyword and Parameter Descriptions

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see "Utility Statement Parsing" on page 15.

RECOVERY

Must be used.

TABLESPACE

Names the table space (and, optionally, the data base it belongs to) for which records are to be deleted.

MODIFY (Utility)

database-name

Is the name of the database the table space belongs to. The **default** is **DSNDB04**.

tablespace-name

Is the name of the table space.

DSNUM

Identifies a single partition or data set of the table space for which records are to be deleted; or it deletes records for the entire table space.

ALL

Deletes records for the entire table space. The **default** is **DSNUM ALL**.

integer

Is the number of a partition or data set.

For a partitioned table space, *integer* is its partition number.

For a nonpartitioned table space, use the data set integer at the end of the data set name as cataloged in the VSAM catalog. The data set name has this format:

catname.DSNDBx.dbname.tsname.I0001.An

where:

catname = VSAM catalog name or alias

x = C or D

dbname = database name

tsname = table space name

n = data set integer

DELETE

Tells that records are to be deleted.

AGE *integer*

Deletes all records older than *n* days, where *n* is the value of *integer*. *n* may range from 0 to 32767. Records created today are of age 0, and cannot be deleted by this option. If you specify (*) for this option, all records will be deleted, regardless of their age.

DATE *integer*

Deletes all records written before the date given by *integer*. *integer* must specify a year (*yy*), month (*mm*), and day (*dd*) in the form *yyymmdd*. It must not be a date in the future. If you specify (*) for this option, all records will be deleted, regardless of the date on which they were written.

Output

The MODIFY utility deletes records of image copy data sets from the SYSIBM.SYSCOPY catalog table, and the related records from the SYSIBM.SYSLGRNG directory. For each record deleted from SYSCOPY, a message is written that gives the name of the copy data set.

Usage Notes

You cannot run MODIFY on a table space that is in recovery or check pending status.

Using MODIFY, it is possible (but not usually desirable) to delete all backup records for a table space. If you want to be able to recover the table space, you must then take a full image copy of it.

Phases of execution: The MODIFY utility operates in these phases:

- UTILINIT: initialization and setup
- MODIFY: record deletion
- UTILTERM: cleanup

Examples

Example 1: For the table space containing the employee table, delete all SYSCOPY records older than 90 days.

```
MODIFY RECOVERY TABLESPACE DSN8D21A.DSN8S21E  
DELETE AGE(90)
```

Example 2: For the table space containing the department table, delete all SYSCOPY records written before 4 July 1985.

```
MODIFY RECOVERY TABLESPACE DSN8D21A.DSN8S21D  
DELETE DATE(850704)
```

PRINT LOG MAP (DSNJU004) (Utility)

The PRINT LOG MAP utility lists the following information:

- Log data set name and log RBA association for both copy 1 and copy 2 of all active and archive log data sets
- Passwords for those data sets, if provided
- Active log data sets available for new log data
- Status of all conditional restart control records in the bootstrap data set, and
- Contents of the queue of checkpoint records in the bootstrap data set.

Sample output from this utility appears in Section 5 of *System and Database Administration Guide*.

Environment

The DSNJU004 program runs as a batch job.

This utility may be executed while the DB2 online subsystem is executing. However, to ensure consistent results from the utility job, the utility and the DB2 online subsystem must both be executing under control of the same MVS system.

Authorization

To use this utility, the primary authorization ID designated by the process must have requisite RACF authorization or, if the BSDS is password protected, the appropriate VSAM password for the data set.

Invoking the Utility

The following EXEC statement is used to invoke this utility:

```
// EXEC PGM=DSNJU004
```

Data Definition Statements

Print Log Map recognizes DD statements with the following ddnames:

JOBCAT

STEPCAT

Names the catalog in which the bootstrap data set (BSDS) is cataloged. The statement is optional. Typically, the high-level qualifier of the BSDS name will point to the ICF catalog that contains an entry for the BSDS.

SYSUT1

Is required to name and allocate the bootstrap data set. It allocates the BSDS. If the BSDS is to be shared with a concurrently executing DB2 online subsystem, use DISP=SHR on the DD statement.

SYSPRINT

Is required to name a data set for print output.

QUIESCE (Utility)

The QUIESCE utility establishes a quiesce point (the current log RBA) for a table space, or list of table spaces, and records it in the SYSIBM.SYSCOPY catalog table.

Environment

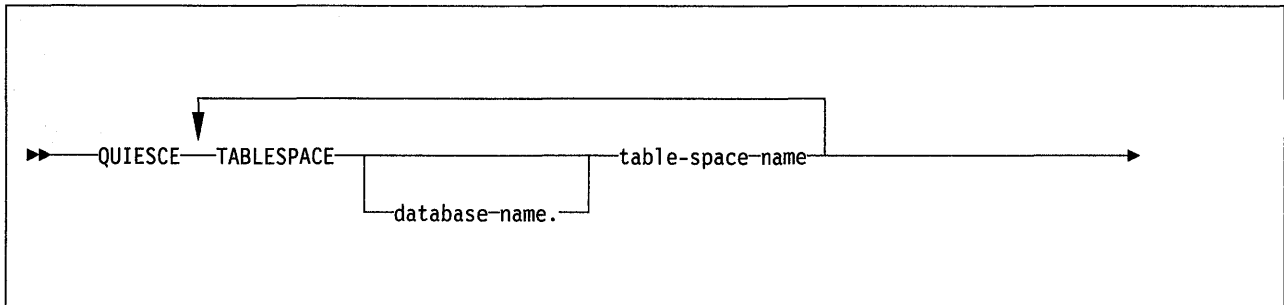
See “Chapter 3. Running DB2 Utilities” on page 151, for a description of ways to invoke DB2 utilities.

Authorization

To execute this utility, the privilege set defined below must include the IMAGCOPY privilege for the database containing the named table space. The IMAGCOPY privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following privileges:

- SYSADM authority
- DBADM, DBCTRL, DBMAINT authorities for a database.

If the utility is being executed on a table space in database DSNDB01 or DSNDB06, the privilege set must include either the SYSADM or SYSOPR authority defined when DB2 was installed.



Keyword and Parameter Descriptions

For a description of how utility statements are parsed, see “Utility Statement Parsing” on page 15.

TABLESPACE *database-name.tablespace-name*

Specifies the table space to be quiesced. *database-name.tablespace-name* identifies this table space.

To create a list of table spaces to be quiesced as a group, repeat the TABLESPACE keyword.

Usage Notes

Restrictions: You cannot QUIESCE a table space that is in “copy pending,” the “check pending,” or “recovery pending” status. See “Partial Recovery” on page 264, “CHECK (Utility)” on page 178, or “Recovery Pending Status” on page 264 for information about resetting these states.

Using QUIESCE for Recovery: You can recover a table space to its quiesce point with the RECOVER utility. See “RECOVER (Utility)” for information about the RECOVER utility; see “REPORT (Utility)” on page 285 for information about obtaining the quiesce point of a table space.

QUIESCE (Utility)

Terminating a Quiesce: If you use -TERM UTILITY to terminate QUIESCE, QUIESCE releases all locks on QUIESCEd table spaces and removes any entries added to the SYSCOPY catalog table.

Restarting a Quiesce: You cannot restart the QUIESCE utility.

Phases of execution: The QUIESCE utility operates in these phases:

1. UTILINIT: initialization and setup
2. QUIESCE: determining the quiesce point, and catalog update
3. UTILTERM: cleanup

Example

Example 1: Establish a quiesce point for the DSN8D21A.DSN8S21E and DSN8D21A.DSN8S21D table spaces.

```
QUIESCE TABLESPACE DSN8D21A.DSN8S21E TABLESPACE DSN8D21A.DSN8S21D
```

Example 2: Establish a quiesce point for the table space set of the sample application.

```
QUIESCE TABLESPACE DSN8D21A.DSN8S21D
        TABLESPACE DSN8D21A.DSN8S21E
        TABLESPACE DSN8D21A.PROJ
        TABLESPACE DSN8D21A.ACT
        TABLESPACE DSN8D21A.PROJACT
        TABLESPACE DSN8D21A.EMPPROJA
```

RECOVER (Utility)

The RECOVER utility recovers data to the current state or a previous state. The largest unit of data recovery is the table space; the smallest is the page. You can recover an entire table space, a data set, pages within an error range, or a single page. Data is recovered from image copies of a table space and database log change records. If the most recent full image copy data set is unusable, and there are previous image copy data sets existing in the system, then RECOVER uses the previous image copy data sets.

It also recovers indexes, by re-creating them from the table they reference. Indexes are recovered in their entirety.

Environment

See “Chapter 3. Running DB2 Utilities” on page 151, for a description of ways to invoke DB2 utilities.

Authorization

To execute this utility, the privilege set defined below must include the RECOVERDB privilege for the database containing the named table space. The RECOVERDB privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following privileges:

- SYSADM authority
- DBADM or DBCTRL authority for the database.

If the utility is being executed on a table space in database DSNDB01 or DSNDB06, the privilege set must include the SYSADM authority defined when DB2 was installed.

The authorization required to invoke a utility job step is discussed under “Authorization” in the description of each utility.

RECOVER (Utility)

Format 1

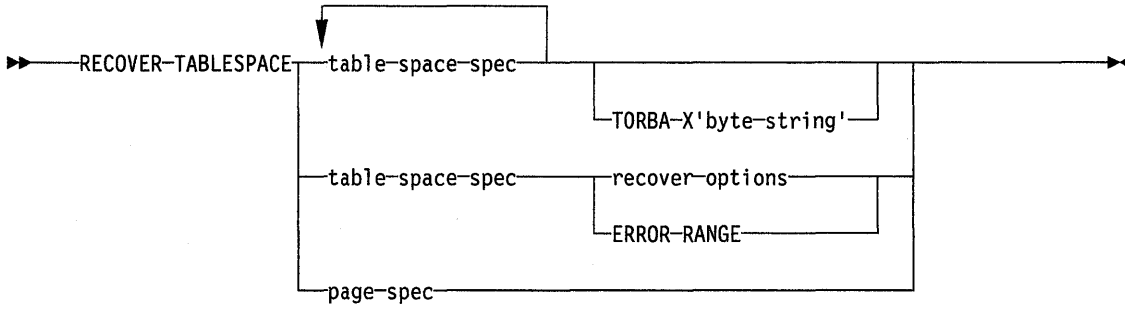
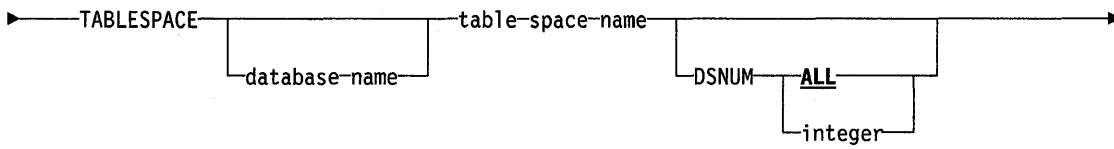
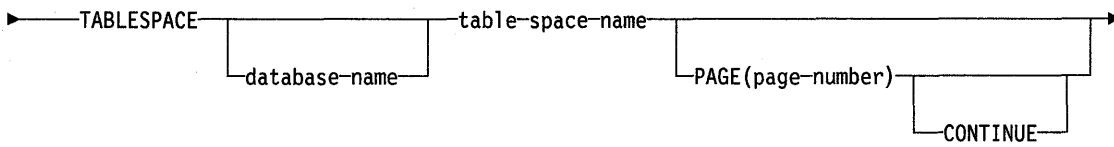


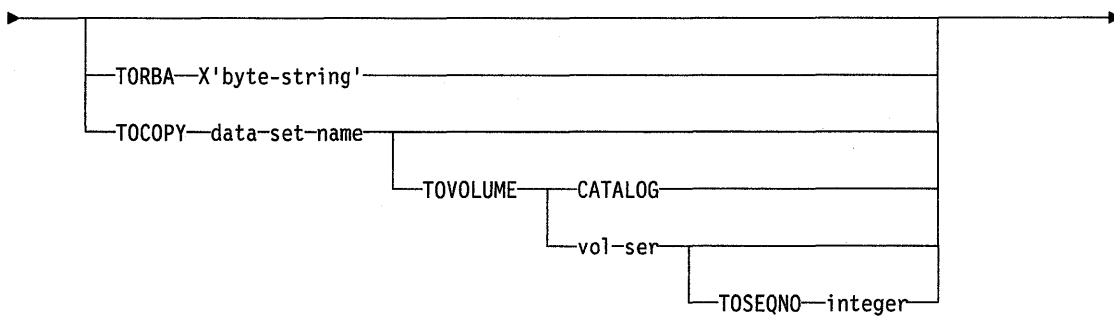
Table Space Spec



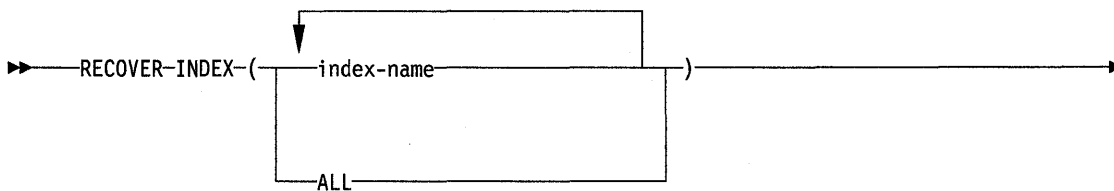
Page Spec

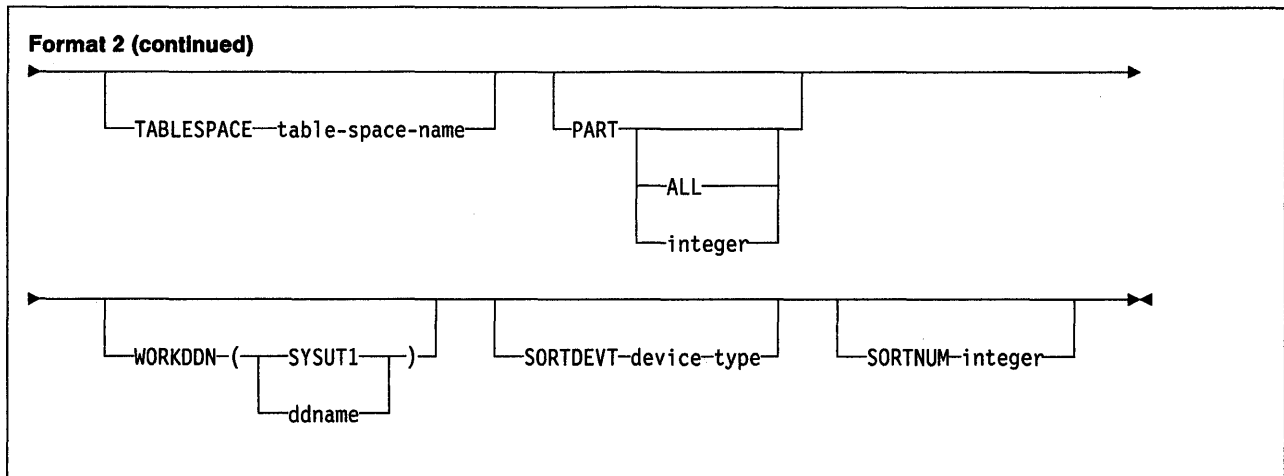


RECOVER options:



Format 2





Keyword and Parameter Descriptions

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see “Utility Statement Parsing” on page 15.

RECOVER TABLESPACE

TABLESPACE

Names the table space (and, optionally, the data base it belongs to) that is to be recovered.

You can specify a list of table spaces by repeating the TABLESPACE keyword. If you use a list of table spaces, all keywords, except DSNUM and TORBA, are invalid.

database-name

Is the name of the database the table space belongs to. The **default** is **DSNDB04**.

tablespace-name

Is the name of the table space to be recovered.

The following are optional.

DSNUM

Identifies a partition or data set, within the table space, that is to be recovered; or it recovers the entire table space.

ALL

Recovers the entire table space. The **default** is **ALL**.

integer

Is the number of a partition or data set to be recovered.

For a partitioned table space, the integer is its partition number.

For a nonpartitioned table space, find the integer at the end of the data set name as cataloged in the VSAM catalog. The data set name has this format:

catname.DSNDBx.dbname.tsname.I0001.An

where:

RECOVER (Utility)

catname = VSAM catalog name or alias
x = C or D
dbname = database name
tsname = table space name
n = data set integer

Note: If the image copies have been taken by data set rather than by table space (that is, with COPY DSNUM *integer*), then you *cannot* recover the entire table space with the single command RECOVER DSNUM ALL. However, you can recover a single data set, using RECOVER DSNUM *integer*, from a table space image copy (taken by COPY DSNUM ALL). Therefore, plan your recovery strategy before making the image copies.

PAGE *page-number*

Names a particular page to be recovered. *page-number* is the number of the page, in either decimal or hexadecimal notation. For example, both 999 and X'3E7' represent the same page.

CONTINUE

Specifies that the recovery process is to continue. Use this option only if RECOVER has terminated during reconstruction of a page, because of an error. In this case, the page is marked as "broken." To resolve the problem, refer to *Diagnosis Guide and Reference* for information about using the REPAIR utility. After you have repaired the page, you can use the CONTINUE option to recover the page, starting from the point of failure in the recovery log.

ERROR RANGE

Specifies that all pages within the range of reported I/O errors are to be recovered. Before data is recovered from the image copy data set and the log data set, each track that contains a page within the error range is examined. If an error is found on the track, an alternate track is used. If an alternate track cannot be found, the RECOVER utility terminates.

TORBA X'*byte-string*'

Terminates the recovery process with the last log record whose relative byte address (RBA) is less than *byte-string*, which is a string of up to 12 hexadecimal digits. If *byte-string* is the RBA of the first byte of a log record, that record is NOT used in recovery.

This option resets the "check pending" status when:

- All members of a table space set are recovered to the same quiesce point, and no referential constraints were defined on a dependent table after that quiesce point. The check pending status is reset for any table space in the table space set.

This option sets the "check pending" status when:

- One or more members of a table space set are recovered to a different quiesce point, so that all members of the table space set are not recovered to the same point in time. Dependent table spaces that are recovered (and their dependent table spaces) are placed in "check pending" status.
- All members of a table space set are recovered to the same quiesce point, but referential constraints were defined on a dependent table after that quiesce point; table spaces containing those dependent tables are placed in "check pending" status.

TOCOPY *dsname*

Terminates the recovery process after applying a particular image copy data set. *dsname* is the name of the data set.

If the data set is a full image copy, it will be the only data set used in recovery. If it is an incremental image copy, the recovery will also use the previous full image copy and any intervening incremental image copies. The effect of using TOCOPY is the same as using TORBA and naming the RBA associated with the image copy data set.

If you use TOCOPY with a particular partition or data set (identified with DSNUM), then the image copy must be for the same partition or data set, or for the whole table space. If you use TOCOPY with DSNUM ALL, the image copy must be for DSNUM ALL.

If there is more than one image copy data set with the name *dsname*, use one of the following options to identify the data set exactly.

TOVOLUME

Identifies the image copy data set.

CATALOG

Identifies the data set as cataloged. Use this option only for an image copy that was created as a cataloged data set. (Its volume serial is not recorded in SYSIBM.SYSCOPY.)

vol-ser

Identifies the data set by an alphameric volume serial identifier. Use this option only for an image copy that was created as a non-cataloged data set.

TOSEQNO *integer*

Identifies the image copy data set by its file sequence number. *integer* is the file sequence number.

RECOVER INDEX**INDEX** *index-name*

Is required only if indexes are to be recovered. *index-name* is the qualified name of an index, in the form *creator-id.index-name*. If you omit the qualifier *creator-id*, the user identifier for the utility job is used.

index-name

Identifies the index to be recovered. To recover multiple indexes, separate each index name with a comma. All indexes listed must reside in the same table space. If more than one index is listed and TABLESPACE keyword is not specified, DB2 will locate the first valid index name cited and determine the table space in which that index resides. That table space will be used as the target table space for all other valid index names listed.

ALL

Specifies that all indexes in the table space referenced by the TABLESPACE keyword are to be recovered.

TABLESPACE *tablespace-name*

Specifies the table space from which all listed indexes are to be recovered. *tablespace-name* identifies this table space.

This keyword is required if INDEX(ALL) is specified. It's optional if one or more index names are specified. If this keyword is omitted, DB2 will locate the first

RECOVER (Utility)

valid index name provided and determine the table space in which that index resides. All other specified indexes must belong to that same table space.

PART

Specifies the partition of a clustering index in a partitioned table space that is to be recovered. This option is only valid when specified for a clustering index in partitioned table space.

integer identifies the partition to be scanned.

ALL specifies that all partitions are to be scanned.

WORKDDN *ddname*

Names the DD statement for the temporary work file. *ddname* is the DD name. The default is **SYSUT1**.

SORTDEVT *device-type*

Names the device type for temporary data sets to be dynamically allocated by DFSORT. It can be any device type acceptable to the DYNALLOC parameter of

the SORT or OPTION option for DFSORT, as described in *DFSORT Application Programming Guide*.

device-type is the device type.

SORTNUM *integer*

Specifies the number of temporary data sets to be dynamically allocated by the sort program. If you omit SORTDEVT, SORTNUM is ignored.

integer is the number of temporary data sets.

Input Data Sets

When RECOVER TABLESPACE is used, RECOVER refers to the SYSIBM.SYSCOPY catalog table.

Output

Output from RECOVER consists of recovered data (either a table space, data set, or error range within a table space).

Usage Notes

These notes do *not* apply to RECOVER INDEX.

If the ERROR RANGE and DSNUM options are omitted, the entire table space is recovered.

Phases of Execution: The RECOVER utility includes the following phases:

UTILINIT The utility is set up and initialized.

RESTORE This phase locates and merges any appropriate image copies, and restores the table space to a backup level.

LOGAPPLY This phase applies all outstanding log changes to the restored table space, to complete the recovery.

UTILTERM Final cleanup is performed.

Terminating with -TERM: Terminating a RECOVER job with the -TERM UTILITY command leaves the object being recovered in an indeterminate state. The data will be unavailable until the object has been successfully recovered.

Input Data Sets: When RECOVER TABLESPACE is used, RECOVER refers to the SYSIBM.SYSCOPY catalog table.

Data Set Definition: If the table space to be recovered is associated with a storage group, DB2 will define the necessary data sets automatically. If the table space is NOT associated with a storage group, DB2 will reset the high used relative byte address (HURBA) for the data set.

Avoiding Damaged Media: When a media error is detected, DB2 prints a message giving the extent of the damage. If an entire volume is bad, and storage groups are being used, you should:

1. Use -ALTER STOGROUP to remove the bad volume and add another.
2. Invoke the RECOVER utility for all table spaces on that volume.

Remove the bad volume first; otherwise, the RECOVER utility could reaccess the damaged media.

Avoiding Damaged Media: If the image copy data sets from which you want to recover reside on the same tape, you do not need to dismount the tape. Specify the following parameters on DD cards (in this example, DD cards COPY1 and COPY2):

```
COPY1 DD UNIT=3480,DSN=COPYT1,DISP=(OLD,PASS),LABEL=1,
        VOL=(,RETAIN,SER=USRDMPI)
COPY2 DD UNIT=3480,DSN=COPYT2,DISP=(OLD,PASS),LABEL=2,
        VOL=(,REF=*.COPY1)
```

Fallback Recovery: If RECOVER cannot use the latest image copy as a starting point for recovery, it attempts to use previous copies, or, if a previous REORG was performed, it attempts to recover from the log. If no good full image copies are available, and no previous REORG was performed, the RECOVER utility will terminate.

However, RECOVER cannot restore from the log past the point at which the object was last reorganized successfully. Also, an object loaded or reorganized with the LOG NO option cannot be recovered until a full image copy is taken. Hence, to establish a fallback level of recovery, you must take two image copies after an operation using LOG NO; you need only one image copy if LOG YES was used.

Partial Recovery: A recovery operation done with the options TORBA or TOCOPY is termed "partial" because it presumably omits some log records.

When a partial recovery is done, updates cannot be allowed until a full image copy is made; otherwise, the log might contain records of updates made to two different sets of data.

Hence a partial recovery sets the "copy pending" status. While that condition is on, the data set may be accessed by SQL only for reading. The condition can be turned off by any of these operations:

- COPY with FULL YES and SHRLEVEL REFERENCE
- REPAIR SET with NOCOPYPEND
- -START DATABASE with ACCESS FORCE
- LOAD with REPLACE
- REORG with LOG YES.

Since a partial recovery leaves data and its indexes in an inconsistent state, plan to perform RECOVER INDEX after a partial recovery.

Recovery Pending Status: You can reset the recovery pending status with any of these operations:

- RECOVER TABLESPACE, for a table space
- RECOVER INDEX, for an index
- LOAD with REPLACE, for a table space
- REPAIR with SET NORCVRPEND, for a table space or index
- -START DATABASE with ACCESS FORCE, for a table space or index.

Note: The last two operations, REPAIR and -START DATABASE, do not fix the table space or index.

Examples

Example 1: Recover from reported media failure in partition 2 of table space DSN8D21A.DSN8S21D.

```
RECOVER TABLESPACE DSN8D21A.DSN8S21D DSNUM 2 ERROR RANGE
```

Example 2: Recover table space DSN8S21D in database DSN8D21A.

```
RECOVER TABLESPACE DSN8D21A.DSN8S21D
```

Example 3: Recover the second partition of table space DSN8S21D.

```
RECOVER TABLESPACE DSN8D21A.DSN8S21D
      DSNUM 2
```

Example 4: Recover the DSN8210.XDEPT1 index, which indexes the DSN8210.TDEPT table in the DSN8D21A database.

```
RECOVER INDEX(DSN8210.XDEPT1)
```

Example 5: Recover table spaces DSN8D21A.DSN8S21E and DSN8D21A.DSN8S21D to their quiesce point (RBA X'000007425468').

```
RECOVER TABLESPACE DSN8D21A.DSN8S21E DSNUM 2
      TABLESPACE DSN8D21A.DSN8S21D
      TORBA (X'000007425468')
```

Example 6: Recover partitions 2 and 3 of the DSN8210.XEMP1 index.

```
RECOVER INDEX (DSN8210.XEMP1) PART 2 PART 3
```

REORG (Utility)

The REORG utility reorganizes a table space to improve access performance and reorganizes indexes so that they are more efficiently clustered. If you specify REORG UNLOAD ONLY or REORG UNLOAD PAUSE, the REORG utility unloads data in a format that is acceptable to the LOAD utility of the same DB2 subsystem.

Environment

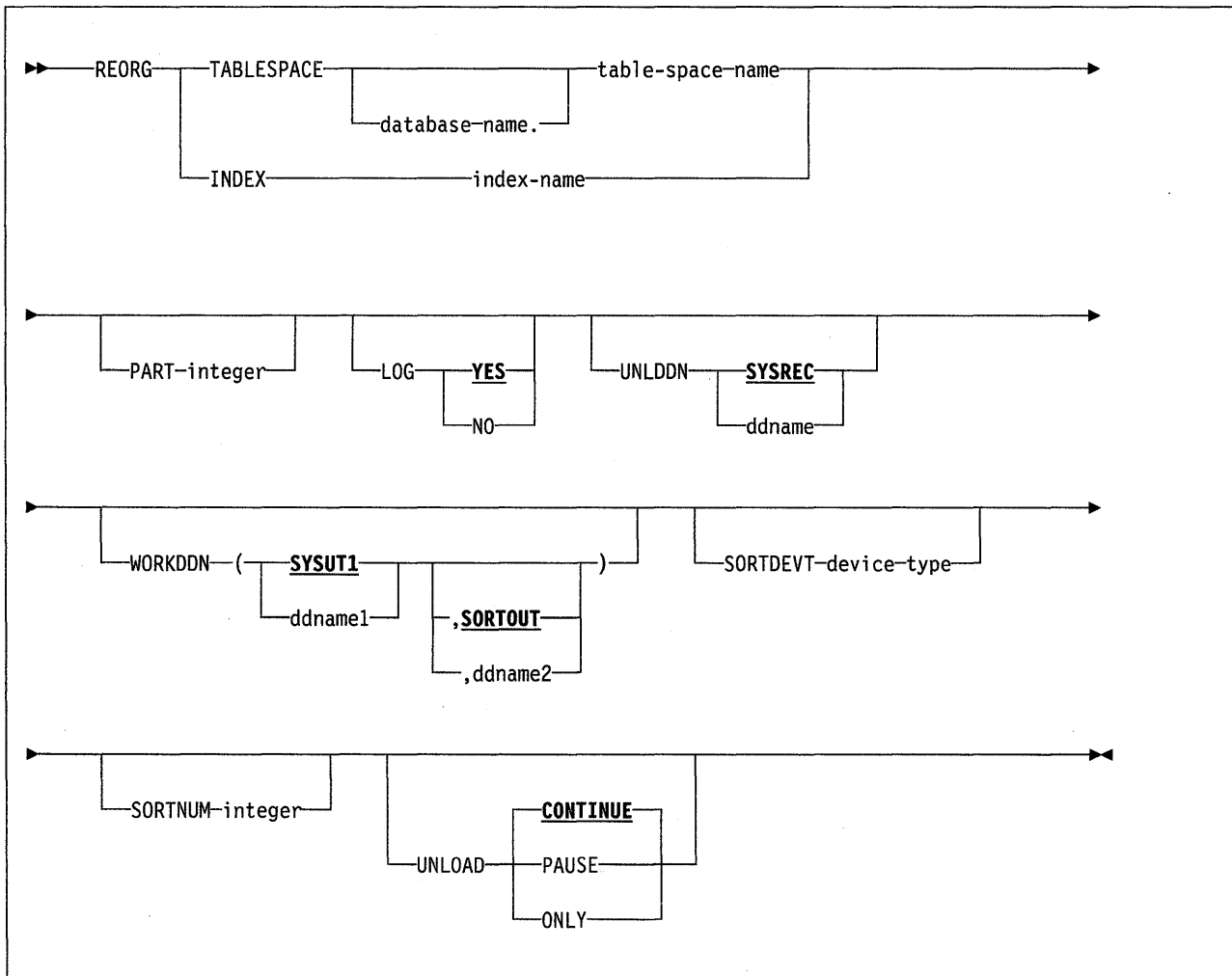
See “Chapter 3. Running DB2 Utilities” on page 151, for a description of ways to invoke DB2 utilities.

Authorization

To execute this utility, the privilege set defined below must include the REORG privilege for the database containing the named table space. The REORG privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following privileges:

- SYSADM authority
- DBADM or DBCTRL authority for the database.

The authorization required to invoke a utility job step is discussed under “Authorization” in the description of each utility.



Keyword and Parameter Descriptions

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see “Utility Statement Parsing” on page 15.

TABLESPACE

Names the table space (and, optionally, the data base it belongs to) that is to be reorganized.

database-name

Is the name of the database the table space belongs to. The name may not be DSNDB06 or DSNDB07. The **default** is **DSNDB04**.

tablespace-name

Is the name of the table space to be reorganized.

The following are optional.

INDEX *index-name*

Names an index to be reorganized. *index-name* is the qualified name of the index, in the form *creator-id.index-name*. If you omit the qualifier *creator-id*, the user identifier for the utility job is used.

PART *integer*

Identifies a partition to be reorganized. You can reorganize a single partition of a partitioned index. *integer* is the number of the partition.

Default: If PART is omitted, the entire table space or index is reorganized.

LOG

Tells whether records are logged during the reload phase of REORG. If the records are not logged, the table space is recoverable only after an image copy has been taken.

You cannot use this option with INDEX.

YES

Logs records during the reload phase. The **default** is **LOG YES**.

NO

Does not log records, and turns on the “copy pending” restriction.

UNLDDN *ddname*

Names the DD statement for the unload data set. *ddname* is the DD name. The **default** is **SYSREC**.

WORKDDN (*ddname1,ddname2*)

Names the DD statements for the temporary work file for sort input and the temporary work file for sort output. A temporary work file for sort output is *required*.

ddname1 is the DD name of the temporary work file for sort input. The **default** is **SYSUT1**.

ddname2 is the DD name of the temporary work file for sort output. The **default** is **SORTOUT**.

SORTDEVT *device-type*

Names the device type for temporary data sets to be dynamically allocated by DFSORT. It can be any device type acceptable to the DYNALLOC parameter of the SORT or OPTION control statement for DFSORT. as described in *DFSORT Application Programming: Guide*.

If you omit SORTDEVT and a sort is required, you must provide the DD statements that the sort program needs for the temporary data sets.

SORTNUM *integer*

Tells the number of temporary data sets to be dynamically allocated by the sort program.

If you omit SORTDEVT, SORTNUM is ignored. If you use SORTDEVT and omit SORTNUM, no value is passed to DFSORT. It is allowed to take its own default.

UNLOAD

Tells whether, after the data has been unloaded, the utility job should continue processing or end.

CONTINUE

Specifies that, after the data has been unloaded, the utility continues processing. Any edit routine or field procedure is bypassed on both the UNLOAD and RELOAD phases. Any validation routine is bypassed on the RELOAD phase.

The **default** is **CONTINUE**.

PAUSE

Specifies that, after the data has been unloaded, processing ends. However, the utility job step is retained, as stopped in the RELOAD phase, so that processing can be restarted in the RELOAD RESTART(PHASE).

This option is useful if you want to redefine data sets during reorganization. For example, with a user defined data set, you can:

1. Run REORG with the UNLOAD PAUSE option.
2. Redefine the data set using access method services.
3. Restart REORG by resubmitting the previous job and specifying RESTART(PHASE).

Any edit routine or field procedure is invoked during both the UNLOAD and RELOAD phases. Any validation procedure is not invoked during either phase.

ONLY

Specifies that, after the data has been unloaded, the utility job ends and the status of the utility is removed. The data on the target data set is in a format compatible with the FORMAT UNLOAD option of LOAD. But with LOAD you can load the data only into the same object from which it was unloaded.

If UNLOAD ONLY is used, any edit routine or field procedure will be invoked during record retrieval in the unload phase.

Input Data Sets

During table space reorganization, these non-DB2 sequential data sets may be needed:

The unload data set: The data will be unloaded to a sequential data set named in the UNLDDN option, then reloaded into the table space.

Segmented table spaces are unloaded segment by segment. Segments that contain a table that has no explicit cluster object are unloaded in physical sequential order into the sequential data set. Segments that contain a table that has an explicit cluster index are unloaded using the cluster index; when the table is unloaded, all data records are in cluster key order.

In a non-segmented table space containing more than one table, REORG does not relocate records to accord with clustering indexes.

The size of the sequential data set, in bytes, may be roughly calculated as *hi-used RBA*, where *hi-used RBA* may be obtained from the associated VSAM catalog.

The unload data set is needed only during the execution of REORG. It can be released upon successful termination.

The work data set: If there are indexes on the table space, a single sequential data set is used to update the index data pointers after the data has been moved. This data set is identified by the DD statement named by the WORKDDN option. It is needed only during the execution of REORG.

To find the approximate size of the WORKDDN data set, in bytes:

1. For each table, multiply the number of records in the table by the number of nonclustering indexes defined on the table.
2. Add the products obtained in step 1.
3. Multiply that sum by the largest key length plus 7.

The sum of the products found in step 2 (S) and the largest key length plus 7 (L, the record length of the WORKDDN data set) are also used to find the size of the temporary data set for sort. See *DFSORT Application Programming: Guide*.

Output

The output from REORG TABLESPACE consists of a reorganized table space or partition; from REORG INDEX, of a reorganized index or index partition.

When reorganizing a segmented table space, REORG leaves free pages, and free space on each page, in accordance with the current values of the FREEPAGE and PCTFREE parameters. (Those values may be set by the CREATE TABLESPACE, ALTER TABLESPACE, CREATE INDEX, or ALTER INDEX statements). REORG leaves one free page after reaching the FREEPAGE limit for each table in the table space. When reorganizing an unsegmented table space, REORG leaves one free page after reaching the FREEPAGE limit, regardless of whether the records loaded belong to the same or different tables.

Segments that contain a table that has an explicit cluster index are unloaded using the cluster index; when the table is loaded, all data records are in cluster key order. See the discussion of unload data sets under "Input Data Sets" for more information.

Usage Notes

Actions Required when the Utility Completes: After a reorganization has completed:

- **If you have used LOG YES**, consider taking an image copy of the reorganized table space or partition to:
 - Permit making incremental image copies later
 - Provide fallback recovery, in addition to that provided by the log records written during reorganization

You may not need to take an image copy of a table space for which **all** the following are true:

- It is relatively small.
- It is used only in read-only applications.
- It can be easily loaded again in the event of failure.

See “COPY (Utility)” on page 185 for information on making image copies.

- **If you have used LOG NO**, REORG places the reorganized table space or partition in copy pending status. See “Partial Recovery” on page 264 for information on resetting the copy pending status.
- Use the RUNSTATS utility on the table space and its indexes, so that the DB2 catalog statistics take into account the newly reorganized data, and SQL paths can be selected with accurate information.

Restarting the REORG Utility: You can restart a REORG utility job at the beginning of any of the phases listed below. For instructions on restarting a utility job, see “Chapter 3. Running DB2 Utilities” on page 151.

The phases are:

- UTILINIT** The utility is set up and initialized.
- UNLOAD** Reads the table in clustering or file scan order and writes it to a sequential data set.
- RELOAD** Reads the records from the sequential data set, loads them to the table space, and extracts index keys.
- SORT** Sorts the key entries before updating indexes, if those exist.
- BUILD** Updates any indexes to reflect the new location of records.
- UTILTERM** Final cleanup is performed.

If the PART option is used, REORG cannot be restarted at the beginning of the BUILD phase.

After an Error in the RELOAD Phase: Failure during the RELOAD phase (after the data has been unloaded and data sets have been deleted, but before the data has been reloaded) results in an unusable table space.

If the error IS NOT on the unloaded data:

- You may allocate new data sets, if necessary (either explicitly or by using storage groups).

- If you do allocate new data sets, restart the REORG job at the beginning of the phase. If you do not allocate new data sets, you can restart either at the last commit point or at the beginning of the phase, as you prefer.

If the error IS on the unloaded data, terminate REORG using -TERM UTILITY. Then recover the table space, using RECOVER, and run the REORG job again.

After Terminating REORG: If REORG is terminated by the -TERM UTILITY command during the unload phase, the object has not yet been changed and the job can be rerun.

Phase	Effect on Pending Status
Unload	No effect.
Reload	Places table space in recovery pending status, then resets the status. Places indexes in recovery pending status. Places table space in copy pending status.
Sort	No effect.
Build	Resets recovery pending status for indexes.

Figure 34. REORG Phases and Pending Status'

DFSORT Messages: The REORG utility job step must contain a UTPRINT DD statement, to define a destination for messages issued by DFSORT during the SORT phase of REORG. The default DD statement used by DB2I and the %DSNU CLIST is:

```
//UTPRINT DD SYSOUT=A
```

Data Set Definition: If the table space is defined by storage groups, space allocation is handled by DB2, and data set definitions cannot be altered during the reorganization process. However, if the table space is supported by data sets maintained by the installation, the physical space parameters can be changed as part of the reorganization.

The procedure is:

1. Specify UNLOAD PAUSE on the REORG statement.
2. When the utility has completed the unload and has stopped, delete and redefine the data sets.

If the table space was created with the CLOSE NO parameter, then the table space must be stopped with the -STOP DATABASE command (with the SPACENAM option) before you delete and redefine the data sets.

3. Resubmit the utility job with the RESTART (PHASE) parameter specified on the EXEC statement. The job will now use the new data sets to do the reload.

Use of the REORG utility to extend data sets causes the newly acquired free space to be distributed throughout the table space rather than to be clustered at the end.

Check Pending Status: You cannot reorganize a table space that is in the "check pending" status. Check the table space for violations of referential constraints, then run the REORG job again. See "CHECK (Utility)" on page 178 for more information about resetting the "check pending" status.

REORG (Utility)

Dropped Tables: Reorganization omits tables that have been previously dropped, reclaiming the space they acquired.

Fallback Recovery: If RECOVER cannot use the latest image copy or copies as a starting point for the recovery, it attempts to use previous copies; if that attempt fails, it restores from the log.

However, RECOVER cannot restore from the log past the point at which the object was last reorganized successfully. Hence, you must take an image copy after a reorganization, to establish a level of fallback recovery.

Examples

Example 1: Reorganize table space DSN8S21D in database DSN8D21A.

```
REORG TABLESPACE DSN8D21A.DSN8S21D
```

Example 2: Reorganize partition 3 of table space DSN8S21E, in database DSN8D21A.

```
REORG TABLESPACE DSN8D21A.DSN8S21E
PART 3
SORTDEVT 3330
```

Example 3: Reorganize index XMSGTXT1. Stop the utility after data has been unloaded, but allow for subsequent restart.

```
REORG INDEX DSN8210.XMSGTXT1
UNLOAD PAUSE
```

REPAIR (Utility)

The REPAIR utility repairs data. The data may be your own data, or data you would not normally access, such as space map pages and index entries.

REPAIR is intended as a means of replacing invalid data with valid data. **Be extremely careful in using REPAIR:** Improper use can damage the data even further.

For information about using REPAIR, refer to “Usage Notes” on page 282 and the section on using REPAIR in “Chapter 3. Running DB2 Utilities” on page 151.

Environment

See “Chapter 3. Running DB2 Utilities” on page 151 for a description of ways to invoke DB2 utilities. Also, see *Diagnosis Guide and Reference* for information on how to use this utility for diagnosing problems.

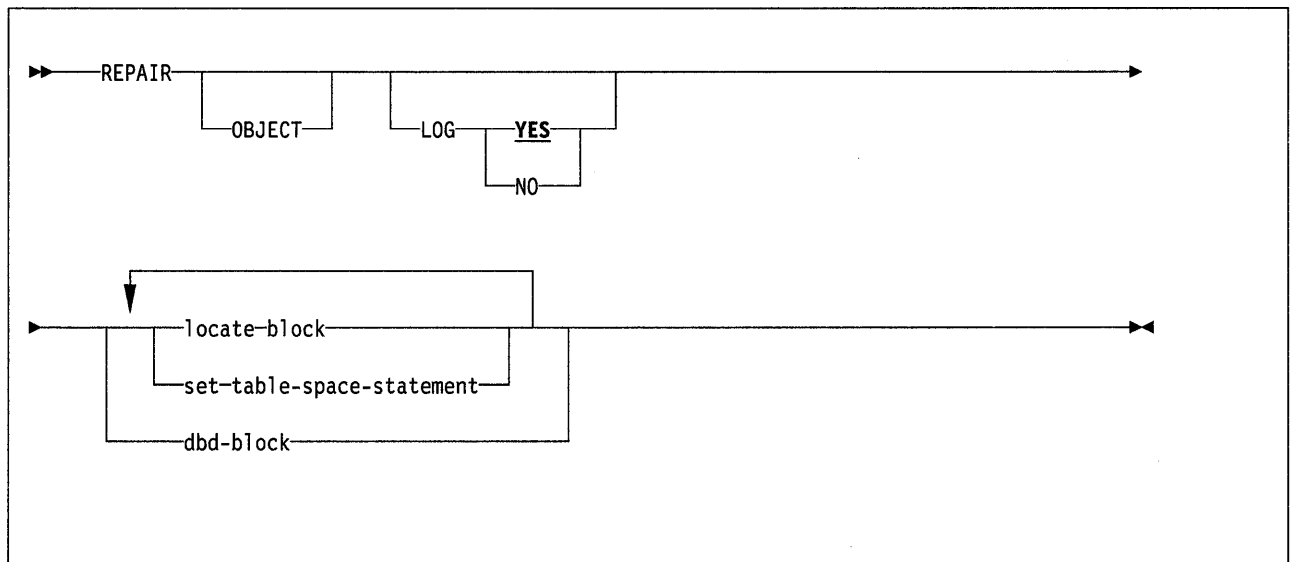
Authorization

To execute this utility, the privilege set defined below must include the REPAIR privilege for the database containing the named table space. The REPAIR privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following privileges:

- SYSADM authority
- DBADM or DBCTRL authorities for the database.

If the utility is being executed on a table space in database DSNDB01 or DSNDB06, or if REPAIR DBD is run, the privilege set must include the SYSADM authority defined when DB2 was installed.

The authorization required to invoke a utility job step is discussed under “Authorization” in the description of each utility.



REPAIR (Utility)

Keyword and Parameter Descriptions

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see “Utility Statement Parsing” on page 15.

OBJECT

Is optional, and used for clarity only.

LOG

Tells whether to log the changes made by REPAIR. If the changes are logged, they will be applied again if the data is recovered.

YES

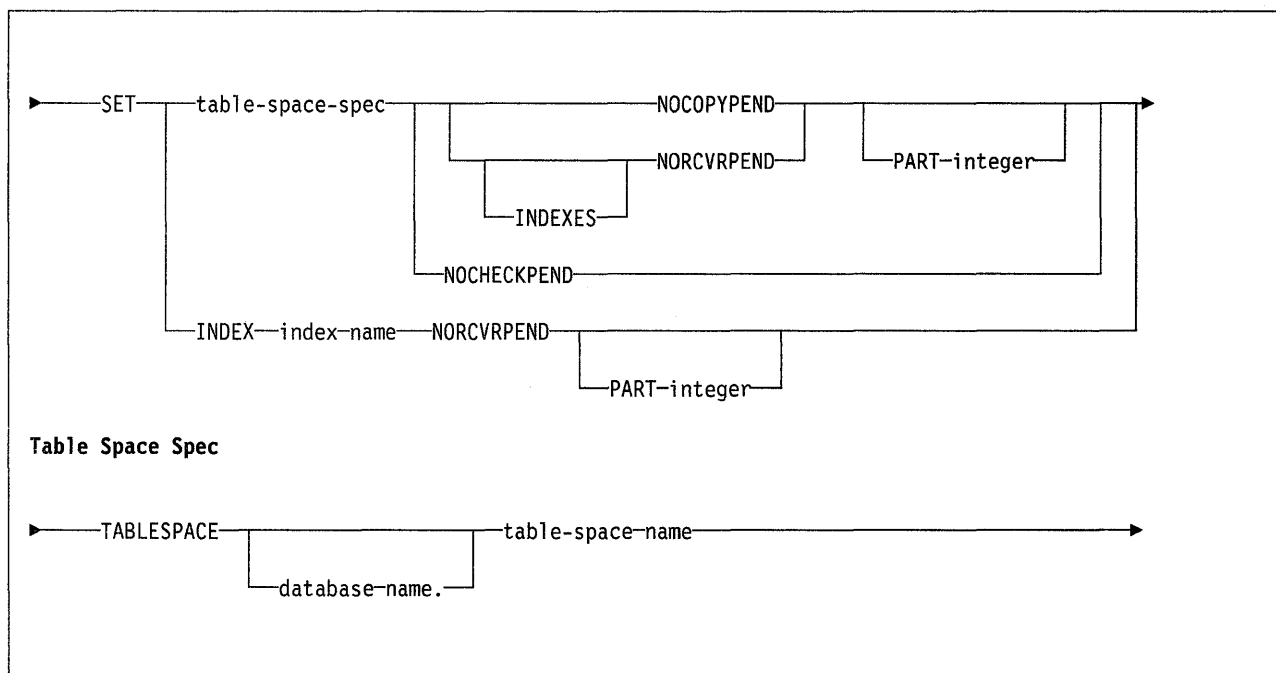
Logs the changes. The **default** is **LOG YES**.

NO

Does not log the changes. You may not use this option with a DELETE statement.

SET TABLESPACE and SET INDEX Statements

The SET TABLESPACE statement resets the copy pending recovery pending, and check pending status for a table space or data set. The SET INDEX statement resets the “recovery pending” status for an index.



Keyword and Parameter Specifications

TABLESPACE

Names the table space (and, optionally, the data base it belongs to) whose “pending” status is to be reset.

database-name

Is the name of the database the table space belongs to. The **default** is **DSNDB04**.

tablespace-name

Is the name of the table space.

INDEX

Names the index whose “recovery pending” status is to be reset.

index-name

Is the name of the index.

NOCOPYPEND

Resets the copy pending status of the specified table space.

NORCVRPEND

Resets the recovery pending status of the specified table space or index.

NOCHECKPEND

Resets the check pending status of the specified table space.

INDEXES

Resets the recovery pending state of all indexes in the specified table space.

If you specify INDEXES, you cannot specify the PART keyword.

PART *integer*

Names a particular partition whose “copy pending” or “recovery pending” status is to be reset. If you do not specify PART, REPAIR will reset the “pending” state of the entire table space or index.

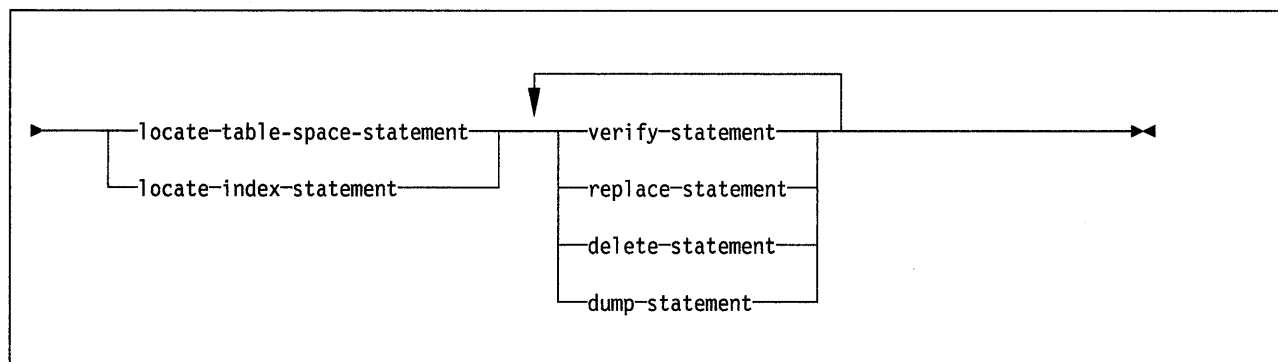
integer is the partition number.

If you specify PART, you cannot specify the INDEXES keyword.

Locate Block

A LOCATE block is a set of statements, each with its own options, that begins with a LOCATE statement and ends with the next LOCATE or SET statement, or with the end of the job. There may be more than one LOCATE block in a REPAIR utility statement.

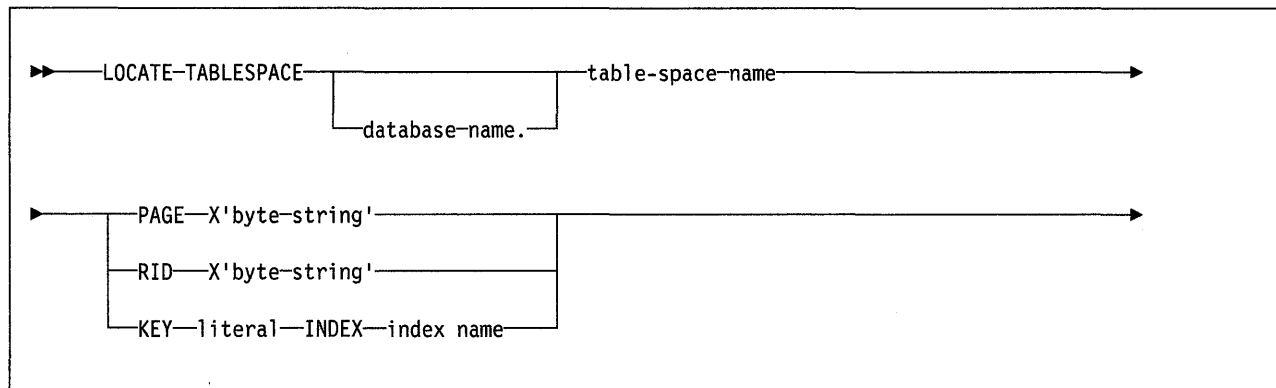
In any LOCATE block, you may use VERIFY, REPLACE, or DUMP as often as you like; you may use DELETE only once.



LOCATE TABLESPACE Statement

The LOCATE TABLESPACE statement locates data to be repaired within a table space.

One LOCATE statement is required for each unit of data to be repaired. Several LOCATE statements may appear after each REPAIR statement.

**Keyword and Parameter Specifications**

database-name.tablespace-name

tablespace-name is the name of the table space containing the data you want to repair.

database-name is the name of the data base to which the table space belongs and is optional. The default is **DSNDB04**.

PAGE X'byte-string'

Specifies that the data of interest is an entire page. The offsets given in *byte-string* and in later statements are relative to the beginning of the page. The first byte of the page is 0.

byte-string may be 1 to 6 hexadecimal digits. You do not need to enter leading zeros. For the method of writing page numbers in partitioned table spaces, see "Usage Notes" on page 282. Enclose the byte string between apostrophes and precede it with X.

RID X'byte-string'

Specifies that the data of interest is a single row. The offsets given in *byte-string* and in later statements are relative to the beginning of the row. The first byte of the stored row prefix is 0.

byte-string may be 1 to 8 hexadecimal digits. You do not need to enter leading zeros. Enclose the byte string between apostrophes and precede it with X.

KEY literal

Specifies that the data of interest is a single row, identified by *literal*. The offsets given in later statements are relative to the beginning of the row. The first byte of the stored row prefix is at offset 0.

literal is any SQL constant that can be compared with the key values of the named index.

If more than one row has the value *literal* in the key column, REPAIR returns a list of record identifiers (RIDs) for records with that key value, but does NOT perform any other operations (verify, replace, delete, or dump) until the next

LOCATE TABLESPACE statement is encountered. To repair the proper data, write a LOCATE TABLESPACE statement that selects the desired row, using the RID option, the PAGE option, or a different KEY and INDEX option. Then invoke REPAIR again.

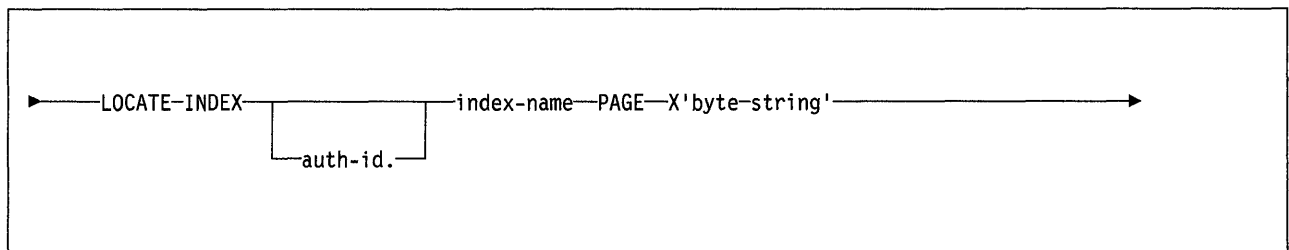
INDEX *index-name*

Names a particular index that is to be used to find the row containing the key. When you are locating by key, the index you specify must be single-column. *index-name* is the qualified or unqualified name of the index.

LOCATE INDEX Statement

The LOCATE INDEX statement locates data to be repaired within an index.

One LOCATE statement is required for each unit of data to be repaired. Several LOCATE statements may appear after each REPAIR statement.



Keyword and Parameter Specifications

index-name

Names the index containing the data you want to repair. *index-name* is the qualified name of the index, in the form *creator-id.index-name*. If you omit the qualifier *creator-id*, the user identifier for the utility job is used.

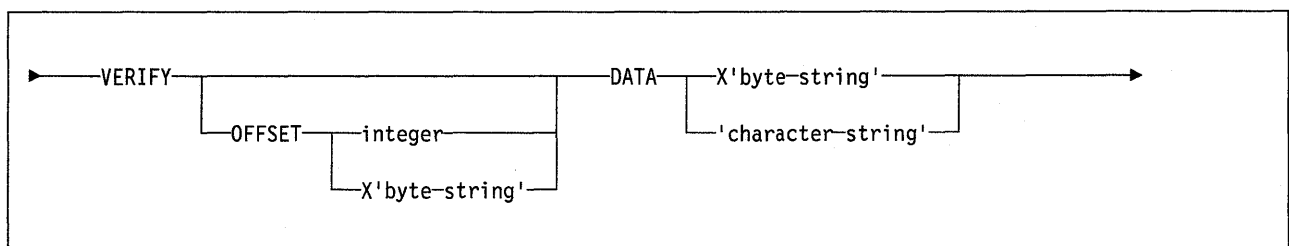
PAGE X'*byte-string*'

Specifies that the data of interest is an entire page. The offsets given in *byte-string* and in later statements are relative to the beginning of the page. The first byte of the page is 0.

byte-string may be 1 to 6 hexadecimal digits. You do not need to enter leading zeros. For the method of writing page numbers in partitioned table spaces, see "Usage Notes" on page 282. Enclose the byte string between apostrophes and precede it with X.

VERIFY Statement

The VERIFY statement tests whether a particular data area contains a specified value. If the data area *does* contain the value, later operations in the same LOCATE block are allowed to proceed. If *any* data area *does not* contain its specified value, *all* later operations in the same LOCATE block are inhibited.



Keyword and Parameter Specifications

OFFSET

Locates the data to be tested by a relative byte address within the row or page.

integer

Gives the offset as an integer. The **default** is **0**, the first byte of the area identified by the previous LOCATE statement.

X'byte-string'

Gives the offset as 1 to 4 hexadecimal digits. You do not need to enter leading zeros. Enclose the byte string between apostrophes and precede it with X.

DATA

Tells what data is assumed to be present before a change is made.

X'byte-string'

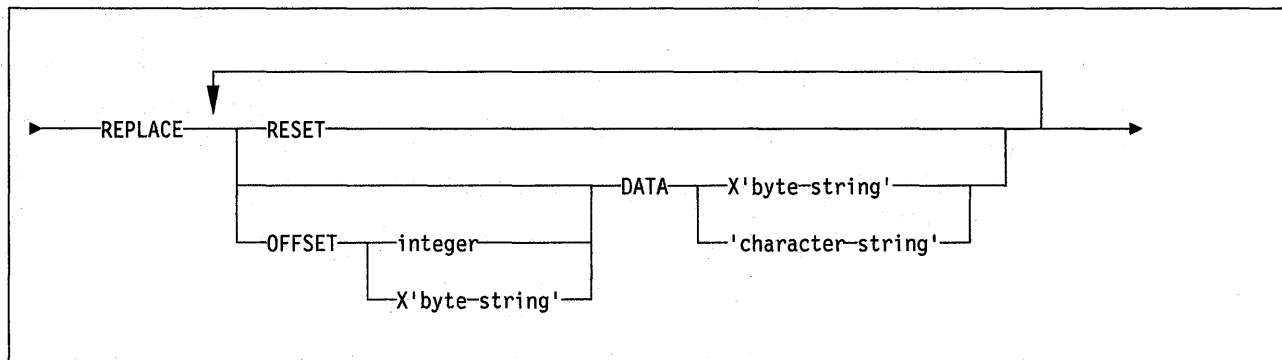
May be an *even number*, from 2 to 32, of hexadecimal digits that must be present. You do not need to enter leading zeros. Enclose the byte string between apostrophes and precede it with X.

'character-string'

May be any character string that must be present.

REPLACE Statement

The REPLACE statement replaces data at a particular location. The statement falls within a LOCATE block. If any VERIFY statement within that block finds a data area that does not contain its specified data, the REPLACE operation is inhibited.



Keyword and Parameter Specifications

RESET

Resets the “inconsistent data” indicator. A page for which this indicator is on is considered “in error,” and the indicator must be reset before you can access the page. Numbers of pages with inconsistent data are reported at the time they are encountered.

The option also resets the “PGCOMB flag bit” in the first byte of the page to agree with the bit code in the last byte of the page.

OFFSET

Tells where data is to be replaced by a relative byte address within the row or page.

integer

Gives the offset as an integer. The **default** is **0**, the first byte of the area identified by the previous LOCATE statement.

X'byte-string'

Gives the offset as 1 to 4 hexadecimal digits. You do not need to enter leading zeros. Enclose the byte string between apostrophes and precede it with X.

DATA

Defines the new data that is to be entered. Only one OFFSET and one DATA specification are acted upon for each REPLACE statement.

X'byte-string'

May be an *even number*, from 2 to 32, of hexadecimal digits to replace the current data. You do not need to enter leading zeros. Enclose the byte string between apostrophes and precede it with X.

'character-string'

May be any character string to replace the current data.

DELETE Statement

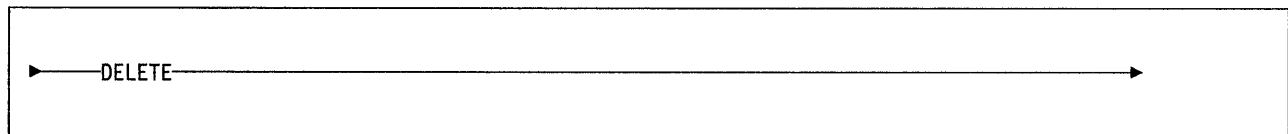
The DELETE statement deletes a single row of data that has been located by a RID or KEY option. The statement falls within a LOCATE block. If any VERIFY statement within that block finds a data area that does not contain its specified data, the DELETE operation is inhibited.

Note: The DELETE statement operates without regard for referential constraints. If you delete a parent row, its dependent rows remain unchanged in the table space.

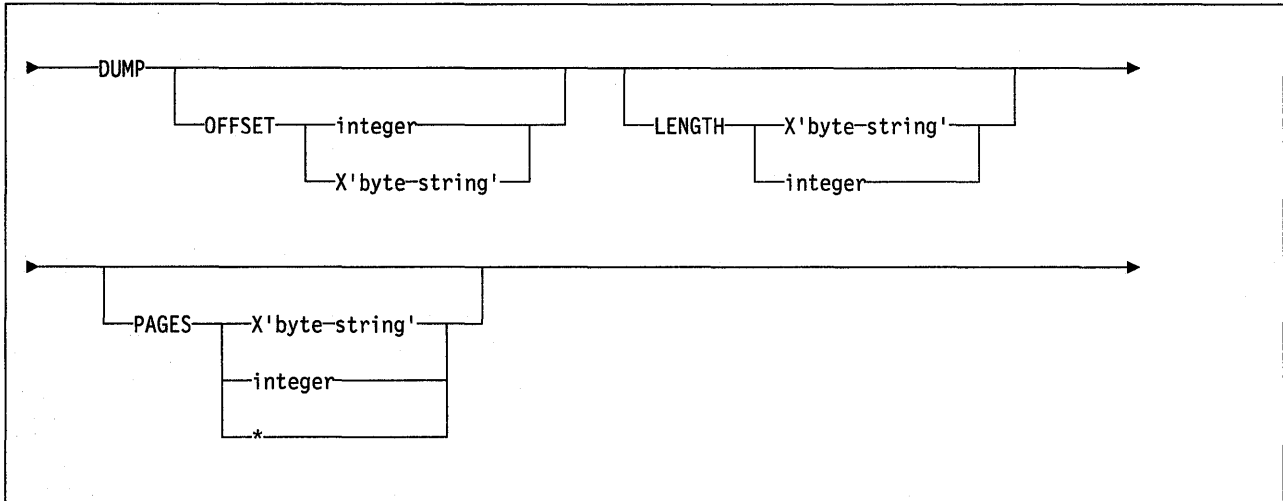
In any LOCATE block, you may use DELETE only once.

You cannot use DELETE if you have used any of these options:

- The LOG NO option on the REPAIR statement
- A LOCATE INDEX statement to begin the LOCATE block
- The PAGE option on the LOCATE TABLESPACE statement in the same LOCATE block
- A REPLACE statement for the same row of data.

**DUMP Statement**

The DUMP statement produces a hexadecimal dump of data identified by offset and length. DUMP statements have no effect on VERIFY or REPLACE operations.



Keyword and Parameter Specifications

OFFSET

Optionally, locates the data to be dumped by a relative byte address within the row or page.

integer

Gives the offset as an integer. The **default** is 0, the first byte of the row or page.

X'byte-string'

Gives the offset as 1 to 4 hexadecimal digits. You do not need to enter leading zeros. Enclose the byte string between apostrophes and precede it with X.

LENGTH

Optionally, tells the number of bytes of data to dump. If you omit both LENGTH and PAGE, the dump continues from OFFSET to the end of the row or page.

If you give a number of bytes (with LENGTH) and a number of pages (with PAGE) the dump contains the same relative bytes from each page. That is, from each page you see the same number of bytes, at the same offset.

PAGES

Optionally, tells a number of pages to dump. You may use this option only if you used PAGE in the preceding LOCATE TABLESPACE control statement.

X'byte-string'

May be 1 to 4 hexadecimal digits. You do not need to enter leading zeros. Enclose the byte string between apostrophes and precede it with X.

integer

Gives the number as an integer.

*

Dumps all pages from the starting point to the end of the table space or partition.

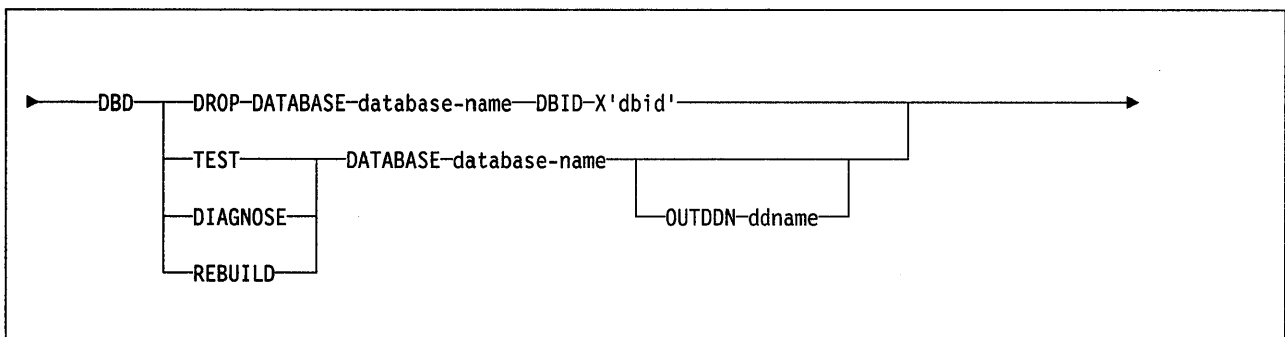
DBD Block

The DBD block allows you to:

- compare the definition of a database in the DB2 catalog with its definition in the DB2 directory
- rebuild the DB2 directory from the information in the DB2 catalog
- drop an inconsistent database definition from the DB2 catalog and the DB2 directory.

DB2 must be operational when you run REPAIR with the DBD block.

REPAIR also assumes that the links in table spaces DSNDB01.DBD01, DSNDB06.SYSDBAUT, DSNDB06.SYSDBASE are intact. Before executing REPAIR with the DBD block, run the DSN1CHKR service aid (page 194) on these table spaces to ensure that the links are not broken.



Keyword Descriptions

TEST

Builds a DBD from information in the DB2 catalog, and compares it with the DBD in the DB2 directory. TEST reports significant differences between the two DBD's.

If the condition code is 0, then the DBD in the DB2 directory is consistent with the information in the DB2 catalog.

If the condition code is not 0, then the information in the DB2 catalog and the DBD in the DB2 directory may be inconsistent. Run REPAIR DBD with the DIAGNOSE option to gather information necessary for resolving any possible inconsistency.

DIAGNOSE

Produces information necessary for resolving an inconsistent database definition. Like the TEST option, DIAGNOSE builds a DBD based on the information in the DB2 catalog and compares it with the DBD in the DB2 directory. In addition, DIAGNOSE reports any differences between the two DBD's, and produces hexadecimal dumps of the inconsistent DBD's.

If the condition code is 0, then the information in the DB2 catalog and the DBD in the DB2 directory is consistent.

If the condition code is 8, then the information in the DB2 catalog and the DBD in the DB2 directory may be inconsistent.

Contact the IBM Support Center for help in resolving any inconsistencies.

REBUILD

Rebuilds the DBD associated with the specified database from the information in the DB2 catalog.

Warning: Use the REBUILD option only under the direction of the IBM Support Center.

DROP

Drops the specified database from both the DB2 catalog and the DB2 directory. Use this keyword if the SQL DROP DATABASE statement fails because the description of the database is not in both the DB2 catalog and the DB2 directory.

Warning: Use the DROP option only under the direction of the IBM Support Center.

DATABASE *dbname*

Names the target database.

DBID *X'dbid'*

Specifies the database descriptor identifier (*dbid*) for the target database.

OUTDDN *ddname*

Names the DD statement for an optional output data set. This data set contains copies of the DB2 catalog records used to rebuild the DBD.

Output

The potential output from the REPAIR utility consists of a modified page or pages in the DB2 table space or index, and a dump of the contents.

Usage Notes

Be extremely careful in using the REPAIR utility to fix invalid data. DB2 does not validate the correctness of replaced data, except as directed by VERIFY statements. Improper use of REPAIR can damage the data even further. Refer to the section on using REPAIR in *Diagnosis Guide and Reference* for more information.

DBD Block: The following is the recommended procedure for using the DBD block:

1. Run the DSN1CHKR service aid on the DSNDB01.DBD01, DSNDB06.SYSDBAUT, and DSNDB06.SYSDBASE table spaces.
2. Run REPAIR DBD with the TEST option to determine if the information in the DB2 catalog is consistent with the DBD in the DB2 directory.
3. If inconsistencies exist (condition code is not zero), use the DIAGNOSE option with the OUTDDN keyword to produce diagnostic information. Contact the IBM Support Center for assistance in analyzing this information.
4. The IBM Support Center may instruct you to replace the existing DBD with the REBUILD option. DO NOT use this option if you suspect that information in the catalog is causing the inconsistency. REBUILD uses information in the catalog to rebuild the DBD; if the catalog is incorrect, the rebuilt DBD will be incorrect.

Check Pending Status: The user is responsible for violations of referential constraints caused by running REPAIR; these violations cause the target table space to be placed in the check pending status. See "CHECK (Utility)" on page 178 for information about resetting this status.

Commit points: If a REPAIR statement is followed by several LOCATE statements, all processing caused by VERIFY, REPLACE, and DUMP statements is committed before the next LOCATE is processed.

Error messages: At each LOCATE statement, the last data page and the new page being located are checked for a few common errors, and messages are issued.

VERIFY, REPLACE, and DELETE: If any data area *does not* contain the value required by a VERIFY statement, all REPLACE and DELETE operations in the same locate block are inhibited. VERIFY and REPLACE statements following the next LOCATE are not affected.

Writing page numbers for partitioned table spaces: If the table space is partitioned, the byte string in "PAGE X'byte-string'" designates the partition number in certain high-order bits and the page number in the low-order bits. To code the partition and page number within the 24-bit (3-byte) string:

1. Find the total number of partitions in the table space (from 1 to 64), and the page size (either 4K or 32K bytes). Both appear in the catalog table SYSIBM.SYSTABLESPACE: The number of partitions is in the PARTITIONS column; the page size is in the PGSIZE column.
2. Using the number of partitions and the page size, find in Figure 35 the number of high-order bits that are used for the partition number. For example, if the total number of partitions is 28 and the page size is 32K, the number of high-order bits used is 8.

Number of Partitions	Page size = 4K	Page size = 32K
1 to 16	4	7
17 to 32	5	8
33 to 64	6	9

Figure 35. Number of High-Order Bits Used for Partition Number

3. Code the partition number, which may range from 0 to 63, in the appropriate number of high-order bits of a 24-bit string. Fill the remainder of the string with zeros. For example, to code 21 in 8 high-order bits, write:

```
00010101 00000000 00000000
```

To code 21 in 9 high-order bits, write:

```
00001010 10000000 00000000
```

4. Code the page number in the low-order bits of a 24-bit string. That is, code it normally as a 24-bit binary number.
5. Add the two strings and convert the result to hexadecimal notation. Use the result as the byte-string after PAGE. For example, if there are 28 partitions and page size is 32K, then code partition 21, page 255, in binary as

```
00010101 000000 11111111
```

Convert that to "1500FF" in hexadecimal. Write the PAGE option as PAGE X'1500FF'.

REPAIR (Utility)

Phases of execution: Though you cannot restart a REPAIR utility job, one of the following phases may be identified if the job terminates.

- UTILINIT: initialization and setup.
- REPAIR: locating and repairing data.
- UTILTERM: cleanup.

Examples

Example 1: Repair the specified page of table space DSN8S21E. Verify that, at the specified offset (50), the damaged data (A00) is found. Replace it with the desired data (D11). Take a dump beginning at offset 50, for 4 bytes, to verify the replacement.

```
REPAIR OBJECT
  LOCATE TABLESPACE DSN8D21A.DSN8S21E PAGE X'02'
  VERIFY OFFSET 50 DATA 'A00'
  REPLACE OFFSET 50 DATA 'D11'
  DUMP OFFSET 50 LENGTH 4
```

Example 2: When loading table space DSNDB04.TS1, you received the following message:

```
DSNU3401 DSNURXBA - ERROR LOADING INDEX, DUPLICATE KEY
  INDEX = EMPINDEX
  TABLE = EMP
  RID OF INDEXED ROW = X'00000201'
  RID OF NON-INDEXED ROW = X'00000503'
```

Delete the nonindexed row and log the change. (The LOG keyword is not required; it will be logged by default.)

```
REPAIR
  LOCATE TABLESPACE DSNDB04.TS1 RID (X'00000503')
  DELETE
```

Example 3: Determine if the DBD's in the DB2 catalog and the DB2 directory are consistent for database DSN8D2AP.

```
REPAIR DBD TEST DATABASE DSN8D2AP
```

Example 4: After running the TEST option on database DSN8D2AP, and determining that the DBD's are inconsistent, determine the differences between the DBD's.

```
REPAIR DBD DIAGNOSE DATABASE DSN8D2AP OUTDDN SYSREC
```


REPORT (Utility)

The REPORT utility reports information about table spaces. REPORT reports:

- recovery history from the SYSIBM.SYSCOPY catalog table
- log ranges from SYSIBM.SYSLGRNG
- volume serial numbers where archive log data sets from the bootstrap data set (BSDS) reside
- names of all table spaces and tables in a table space set.

Environment

See “Chapter 3. Running DB2 Utilities” on page 151, for a description of ways to invoke DB2 utilities.

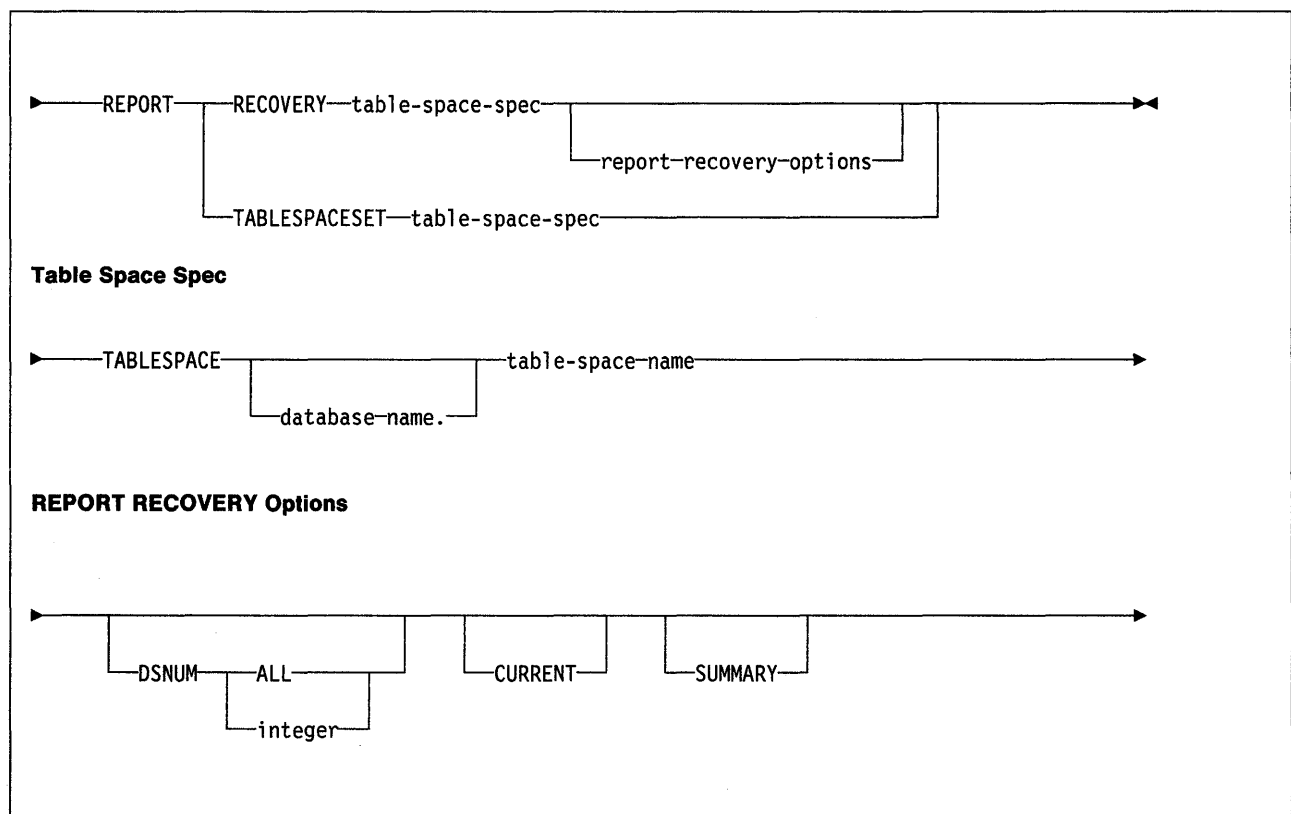
Authorization

To execute this utility, the privilege set defined below must include the RECOVERDB privilege for the database containing the named table space. The RECOVERDB privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following privileges:

- SYSADM authority
- DBADM or DBCTRL authority for the database.

If the utility is being executed on a table space in database DSNDB01 or DSNDB06, the privilege set must include the SYSADM authority defined when DB2 was installed.

The authorization required to invoke a utility job step is discussed under “Authorization” in the description of each utility.



Keyword and Parameter Descriptions

For a description of how utility statements are parsed, see "Utility Statement Parsing" on page 15.

RECOVERY

Indicates that recovery information for the specified table space is to be reported.

TABLESPACESET

Indicates that the names of all table spaces in the table space set are to be reported.

TABLESPACE

For REPORT RECOVERY, names the table space (and, optionally, the data base it belongs to) for which recovery information is to be reported.

For REPORT TABLESPACESET, names a table space (and, optionally, the data base it belongs to) in the table space set.

database-name

Names the database the table space belongs to, and is optional.

tablespace-name

Names the table space.

The following are optional.

DSNUM

Identifies a partition or data set, within the table space, for which information is to be reported; or it reports information for the entire table space.

ALL

Reports information for the entire table space. The **default is ALL**.

integer

Is the number of a partition or data set for which information is to be reported.

For a partitioned table space, the integer is its partition number.

For a nonpartitioned table space, find the integer at the end of the data set name as cataloged in the VSAM catalog. The data set name has this format:

catname.DSNDBx.dbname.tspace.I0001.An

where:

catname = VSAM catalog name or alias

x = C or D

dbname = database name

tspace = table space name

n = data set integer

CURRENT

Specifies that only the entries since the last recoverable point of the table space are to be reported. The last recoverable point is the last full image copy, LOAD REPLACE LOG YES, or REORG LOG YES.

If you do not specify CURRENT, all SYSCOPY and SYSLGRNG entries for that table space are reported.

SUMMARY

Specifies that only a summary of volume serial numbers is to be reported.

If you do not specify SUMMARY, recovery information is reported in addition to the summary of volume serial numbers.

Output

The output from REPORT TABLESPACESET consists of the names of all table spaces in the table space set you specify. It also names all tables in the table spaces, and names all tables dependent upon those tables.

For example, REPORT TABLESPACESET DSN8D21A.DSN8S21D generates the following output:

DSNU587I - REPORT TABLESPACE SET WITH TABLESPACE DSN8D21A.DSN8S21D

TABLESPACE	TABLE	DEPENDENT TABLE
DSN8D21A.DSN8S21D	DSN8210.DEPT	DSN8210.DEPT DSN8210.EMP DSN8210.PROJ
DSN8D21A.DSN8S21E	DSN8210.EMP	DSN8210.PROJ DSN8210.EMPPROJA
DSN8D21A.PROJ	DSN8210.PROJ	DSN8210.PROJ DSN8210.PROJACT
DSN8D21A.ACT	DSN8210.ACT	DSN8210.PROJACT
DSN8D21A.PROJACT	DSN8210.PROJACT	DSN8210.EMPPROJA
DSN8D21A.EMPPROJA	DSN8210.EMPPROJA	

Examples

Example 1: The following statement reports the names of all table spaces in the table space set containing table space DSN8D21A.DSN8S21E.

```
REPORT TABLESPACESET TABLESPACE DSN8D21A.DSN8S21E
```

Example 2: This statement reports recovery information for table space DSN8D21A.DSN8S21D.

```
REPORT RECOVERY TABLESPACE DSN8D21A.DSN8S21D DSNUM ALL
```

RUNSTATS (Utility)

RUNSTATS scans a table space or indexes to gather information about

- Utilization of space
- Efficiency of indexes.

The information is recorded in the DB2 catalog, and is used by the SQL optimizer to select access paths to data during the bind process. It is available to the database administrator for evaluating database design, and determining when table spaces or indexes should be reorganized.

Environment

See “Chapter 3. Running DB2 Utilities” on page 151, for a description of ways to invoke DB2 utilities.

Authorization

To execute this utility, the privilege set defined below must include the STATS privilege for the database containing the named table space. The STATS privilege may have been explicitly granted or may be inherent in another privilege; it is inherent in the following authorities:

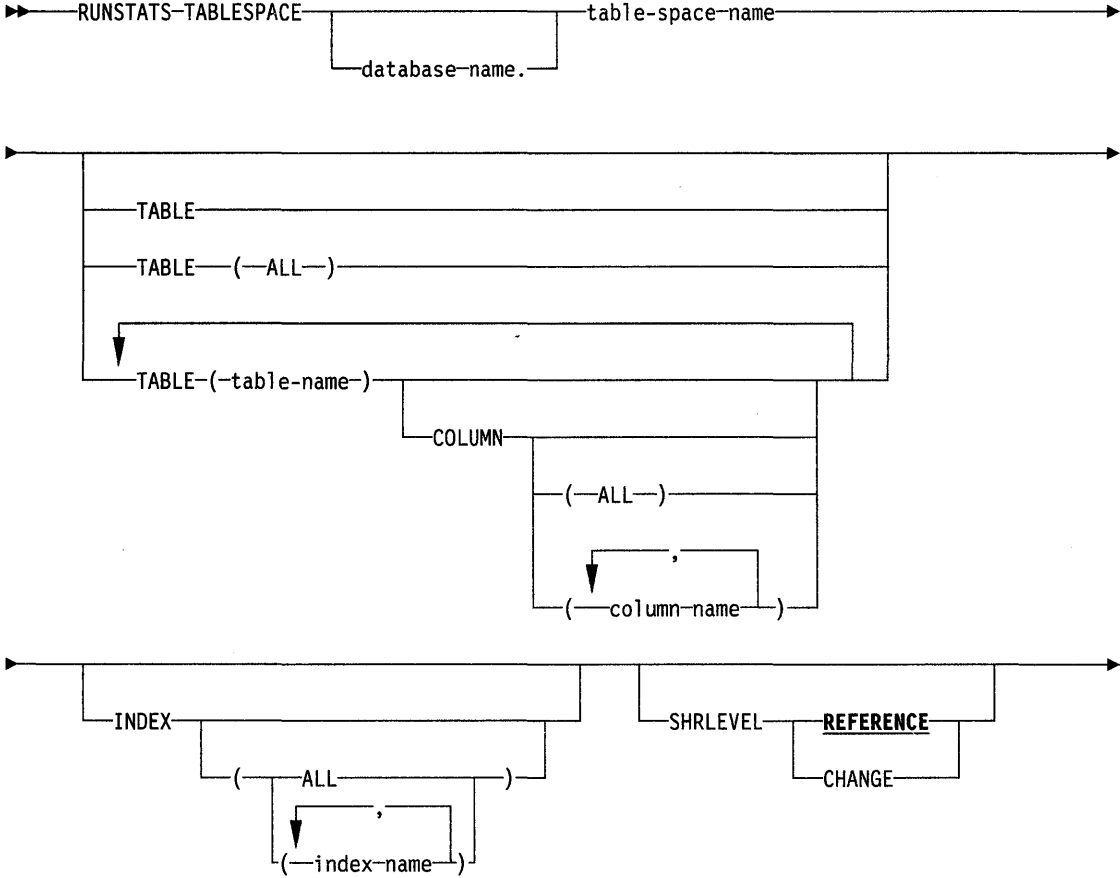
- SYSADM authority
- DBADM, DBCTRL, and DBMAINT authorities for a database.

The authorization required to invoke a utility job step is discussed under “Authorization” in the description of each utility.

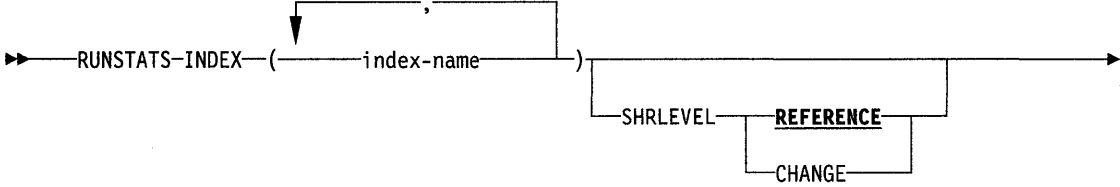
Syntax

There are two formats for the RUNSTATS utility. Format 1 gathers statistics on a table space and, optionally, indexes or tables; format 2 gathers statistics on indexes only. Both formats are shown below.

Format 1



Format 2



Keyword and Parameter Descriptions

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see "Utility Statement Parsing" on page 15.

TABLESPACE

Names the table space (and, optionally, the data base it belongs to) for which information is to be gathered. It must not be a table space in DSNDB01 or DSNDB07.

database-name

Is the name of the database the table space belongs to. The **default** is **DSNDB04**.

tablespace-name

Is the name of the table space about which information is gathered.

INDEX

Names indexes for which information is to be gathered. All the indexes must be associated with the *same* table space. In a format-1 statement, that must be the table space named in the TABLESPACE option. Column statistics are gathered only on the first column of a key, and only if INDEX is used.

(ALL)

Specifies that information is to be gathered for all indexes in the table space. The parentheses are required. The **default** is **ALL**.

(index-name)

Names the index(es) for which information is to be gathered. The parentheses are required.

You may specify a list of index names. If you specify more than one index, separate each name with a comma.

TABLE

Names the table for which column information is to be gathered. All tables must belong to the table space named in the TABLESPACE option.

(ALL)

Specifies that information is to be gathered for all tables in the table space. The **default** is **ALL**.

(table-name)

Names the table(s) for which information is to be gathered. The parentheses are required. If you omit the qualifier, the user identifier for the utility job is used.

If you specify more than one table, you must repeat the TABLE option.

COLUMN

Names columns for which non-indexed column statistics are to be calculated.

You may only specify this option if you name a specific table(s) for which statistics are to be gathered (TABLE *(table-name)*). If you name specific tables and do not specify the COLUMN option, the default is **COLUMN (ALL)**.

(ALL)

Specifies that statistics are to be gathered for all columns in the table.

(column-name)

Names the column(s) for which statistics are to be gathered. The parentheses are required.

You may specify a list of column names; the maximum is 10. If you specify more than one column, separate each name with a comma.

SHRLEVEL

Tells whether other programs that access the table space while RUNSTATS is running must use read-only access, or may change the table space.

REFERENCE

Allows only read-only access by other programs. The default is **REFERENCE**.

CHANGE

Allows other programs to change the table space or index. With SHRLEVEL CHANGE, the utility may take longer to execute, and statistics may be inconsistent.

Output

Index Statistics and Table Space Statistics: RUNSTATS updates the table columns in the DB2 catalog that are listed below.

Columns used during the bind process: These columns are used by the SQL optimizer to select access paths to data. Refer to "DB2 Catalog Tables" in *SQL Reference*, for descriptions of the values of the columns.

- SYSIBM.SYSCOLUMNS
 - HIGH2KEY
 - LOW2KEY
 - COLCARD
- SYSIBM.SYSINDEXES
 - CLUSTERED
 - FIRSTKEYCARD
 - FULLKEYCARD
 - NLEAF
 - NLEVELS
 - CLUSTERRATIO
- SYSIBM.SYSTABLES
 - CARD
 - NPAGES
 - PCTPAGES
- SYSIBM.SYSTABLESPACE
 - NACTIVE

Columns for Tuning Information: The following catalog table columns are updated by RUNSTATS to help database administrators assess the status of a particular table space or index.

SYSIBM.SYSTABLEPART

- CARD** The number of rows in the table space or partition.
- The database administrator can validate design assumptions against this actual count. Over a period of time, it can show the rate of change or growth of the table space.
- NEARINDREF** The number of rows that have been relocated near their original page. (See the note following FARINDREF.)
- FARINDREF** The number of rows that have been relocated far from their original page. (See the following note.)

Note: If an update operation increases the length of a record by

more than the amount of space available in the page it is stored in, the record is moved to another page. Until the table space is reorganized, the record needs an additional page reference when it is accessed. The sum of NEARINDREF and FARINDREF is the total number of such records.

A page is considered "near" the present page if the two page numbers differ by less than 64; otherwise, it is "far from" the present page. A record relocated near its original page tends to be accessed more quickly than one relocated far from its original page.

PERCACTIVE The percentage of space occupied by active rows, containing actual data from active tables. Other space is free, or occupied by dropped records.

A database administrator can use this figure to validate design assumptions, and tell how much of the space allocated to the table space is utilized.

PERCDROP The percentage of space occupied by rows of data from dropped tables.

Space occupied by dropped tables is reclaimed by the REORG utility. Hence, this figure is one indicator of when a table space should be reorganized.

SYSIBM.SYSINDEXPART

CARD The number of table space rows pointed to by index values in one partition of a partitioned index.

These figures, for all the partitions, tell the database administrator how the key ranges specified for each partition have divided the rows among the several partitions.

NEAROFFPOS The number of times it would be necessary to access a different, nearby page when accessing all the records in index order. Each time, it is probable that accessing the next record would require I/O activity. (See the note following FAROFFPOS.)

A page is considered "nearby" the present page if the two page numbers differ by less than 64.

FAROFFPOS The number of times it would be necessary to access a different, far-off page when accessing all the records in index order. Each time, it is almost certain that accessing the next record would require I/O activity. A page is considered "far-off" from the present page if the two page numbers differ by 64 or more.

Note: Together, NEAROFFPOS and FAROFFPOS tell how well the index follows the cluster pattern of the table space. For a clustering index, NEAROFFPOS approaches the number of pages in the table space, and FAROFFPOS approaches 0, as clustering improves. A reorganization should bring them nearer their optimal values. Changes to the data can change NEAROFFPOS and FAROFFPOS, and the changes can help tell when to reorganize.

LEAFDIST 100 times the average distance in page IDs between successive leaf pages during a sequential access of the index.

This value helps to tell how well an index is organized. The value is at its lowest just after the index has been reorganized. Changes increase it; and you can reduce it again by reorganizing the index, either explicitly or as part of a general table space reorganization.

Usage Notes

When to Use RUNSTATS: The SQL optimizer uses the statistics generated by RUNSTATS to determine access paths to data. If no statistics are available, the optimizer makes fixed default assumptions. To ensure the effectiveness of the paths selected, use RUNSTATS whenever:

- A table is loaded
- An index is created
- A table space is reorganized
- There have been extensive updates, deletions, or insertions in a table space.

After using RUNSTATS, rebind any application plans that use the tables or indexes involved, to improve performance.

Changes to a table space may also change its space requirements and performance. A database administrator can use RUNSTATS to assess the current status of the table space and help decide whether to reorganize or redesign the table space.

The database administrator should use caution when running RUNSTATS after another user has updated the statistical columns of the catalog. Since RUNSTATS puts information in these columns, values changed by the user will be replaced.

Phases of Operation: The RUNSTATS utility operates in these phases:

- UTILINIT: initialization and setup
- RUNSTATS: scanning of the table space or index, and catalog update
- UTILTERM: cleanup

Examples

Example 1: Update the catalog statistic columns for table space DSN8S21E and all its associated indexes. Permit other processes to make changes while this utility is executing.

```
RUNSTATS TABLESPACE DSN8D21A.DSN8S21E
INDEX (ALL)
SHRLEVEL CHANGE
```

Example 2: Update the catalog statistics for indexes XEMPL1 and XEMPL2. Do not permit other processes which change the table space associated with XEMPL1 and XEMPL2 (table space DSN8S21E) to execute while this utility is executing.

```
RUNSTATS INDEX (DSN8210.XEMPL1,DSN8210.XEMPL2)
```

Example 3: Obtain statistics on the index XEMPL1.

```
RUNSTATS INDEX (DSN8210.XEMPL1)
```

RUNSTATS (Utility)

Example 4: Update the catalog statistics for all columns in the TCONA and TOPTVAL tables in table space DSN8D21P.DSN8S21C. Update the column statistics for the LINENO and DSPLINE columns in the TDSPTXT table in table space DSN8D21P.DSN8S21C.

```
RUNSTATS TABLESPACE(DSN8D21P.DSN8S21C) TABLE (TCONA)
                                     TABLE (TOPTVAL) COLUMN(ALL)
                                     TABLE (TDSPTXT) COLUMN(LINENO,DSPLINE)
```

STOSPACE (Utility)

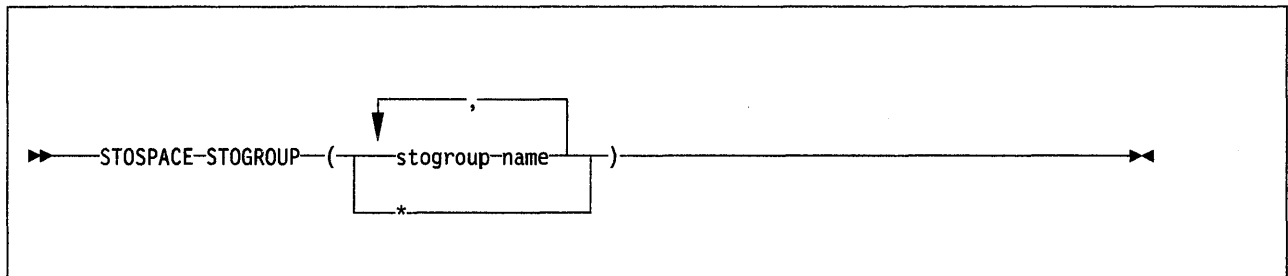
The STOSPACE utility updates DB2 catalog columns that tell how much space is allocated for storage groups and related table spaces and indexes.

Environment

See “Chapter 3. Running DB2 Utilities” on page 151, for a description of ways to invoke DB2 utilities.

Authorization

To execute this utility, the privilege set, consisting of the union of all privileges designated by the authorization IDs of the process, must include SYSADM or SYSOPR authority, or the STOSPACE privilege.



Keyword and Parameter Descriptions

For a description of how utility statements are parsed, and how to read a list of option identifiers and specifications like the one that follows, see “Utility Statement Parsing” on page 15.

STOGROUP

Identifies the groups to be processed.

(*stogroup-name*)

Is the name of a storage group. You may use a list of from one to eight storage group names. Separate items in the list by commas and enclose it in parentheses.

*

Processes all storage groups.

Output

The output from STOSPACE consists of new values in the columns and tables listed. In each case, an amount of space is given in kilobytes (1 kilobyte = 1024 bytes).

- SPACE in SYSIBM.SYSINDEXES shows the amount of space allocated to indexes.
- SPACE in SYSIBM.SYSTABLESPACE shows the amount of space allocated to table spaces.
- SPACE in SYSIBM.SYSSTOGROUP shows the amount of space allocated to storage groups.
- SPCDATE in SYSIBM.SYSSTOGROUP show, in the form *yyddd*, the date when STOSPACE was last used on a particular storage group.

Usage Notes

Meaning of SPACE: The value in a SPACE column is total allocated space, not only space allocated on the current list of volumes in the storage groups. Volumes may be deleted from a storage group even though space on those volumes is still allocated to DB2 table spaces or indexes. Deletion of a volume from a storage group prevents future allocations; it does not withdraw a current allocation.

If the table space or index space is partitioned, and different storage groups have been specified, the SPACE column (of SYSIBM.SYSTABLESPACE or SYSIBM.SYSINDEXES) gives the number of kilobytes of storage allocated to a partition. The partition is determined by the last execution of the STOSPACE utility.

Availability Requirements: For each named storage group, STOSPACE looks at the SYSIBM.SYSTABLESPACE and SYSIBM.SYSINDEXES catalog tables to tell which objects belong to that storage group. For each object, the amount of space allocated is determined from an appropriate VSAM catalog. Hence the table spaces and indexes need not be available to DB2 when STOSPACE is running; only the DB2 catalog and appropriate VSAM catalogs are needed. However, in order to gain access to the VSAM catalog, the utility must have available to it the DBD for the objects involved. This requires that the appropriate database, table spaces, and index spaces not be in the stopped state.

Phases of execution: You can restart a STOSPACE utility job at the beginning of any of the phases listed below. For instructions on restarting a utility job, see "Chapter 3. Running DB2 Utilities" on page 151.

The STOSPACE utility operates in these phases:

1. UTILINIT: initialization and setup
2. STOSPACE: statistics gathering and catalog update
3. UTILTERM: cleanup

Example

Example: Update the DB2 catalog SPACE columns for storage group DSN8G210.

```
STOSPACE STOGROUP DSN8G210
```

Appendix. DB2 Limits

The table below describes certain limits imposed by DB2. For install limits, see Section 2 of *System and Database Administration Guide*.

ITEM	DB2 LIMIT
Maximum number of concurrent DB2 or application agents	Limited by EDM pool size, buffer pool size, and amount of storage used by each DB2 or application agent
Most temporary files active for a single agent	255
Largest table or table space	64 gigabytes
Largest active log space	2 ⁴⁸ bytes
Largest archive log space	2 ⁴⁸ bytes
Most active log copies	2
Most archive log copies	2
Most active log data sets (each copy)	53
Most archive log volumes (each copy)	1000
Number of databases accessible to an application or end user	System storage/EDM pool size
Largest EDM pool	Installation parameter maximum dependent upon available space in memory
Maximum total DASD storage	System storage
Most databases	65279
Most objects in a database	Limiting factors: database descriptor size; most data sets in one ASID (1627).
Largest DBD	65K bytes
Largest OBD	32K bytes
Largest SKCT	System storage/EDM pool size
Largest utility statement block (USM)	32K bytes
Lowest DATE value (shown in ISO format)	0001.01.01
Maximum DATE value (shown in ISO format)	9999.12.31

Figure 36 (Part 1 of 2). DB2 Internal Limits

ITEM	DB2 LIMIT
Lowest TIME value (shown in ISO format)	00.00.00
Maximum TIME value (shown in ISO format)	24.00.00
Lowest TIMESTAMP value	0001-01-01-00.00.00.000000
Maximum TIMESTAMP value	9999-12-31-24.00.00.000000
Maximum number of rows per page	127

Figure 36 (Part 2 of 2). DB2 Internal Limits

Glossary

The following terms and abbreviations are defined as they are used in this book. If you do not find the term you are looking for, refer to the index or to the *IBM Dictionary of Computing*, SC20-1699.

abnormal end of task (abend). Termination of a task, a job, or a subsystem because of an error condition that cannot be resolved during execution by recovery facilities.

abort. To terminate a thread by undoing the uncommitted changes to data back to a prior point of consistency.

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 will fit on the active log.

ANS. American National Standard

application. A program or set of programs that perform a task; for example, a payroll application.

application plan. The control structure produced during the bind process and used by DB2 to process SQL statements encountered during application execution.

archive log. The portion of the DB2 log that contains log records that have been moved from the active log because they no longer fit.

attachment facility. An interface between DB2 and TSO, IMS/VS, CICS, or batch address spaces. An attachment facility allows application programs to access DB2.

authorization ID. A string that designates a set of privileges. It may represent an individual, an organizational group, or a function, but DB2 does not determine this representation.

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.

bind.

automatic bind. Binding done automatically (without a user issuing a BIND command) when an application process begins execution and the bound application plan it needs is invalid.

dynamic bind. Binding done dynamically (as the SQL statements are entered) when SQL statements are prepared through dynamic SQL.

incremental bind. Binding of an SQL statement is done during the execution of an application process because the statement could not be bound during the bind process and VALIDATE(RUN) was specified.

static bind. The process by which the output from the precompiler is converted to a usable control structure called an application plan. This process is the one during which access paths to the data are selected and some authorization checking is performed.

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 reserved to satisfy the buffering requirements for one or more table spaces or indexes.

catalog. A collection of tables that contains descriptions of objects such as tables, views, and indexes.

character string. A sequence of bytes representing bit data, single-byte characters, or a mixture of single and double-byte characters.

check pending. A state of a table space that prevents its use by the COPY and REORG utilities or by SQL data manipulation, because it may contain rows that violate referential constraints.

checkpoint. A point at which DB2 records internal status information on the DB2 log that would be used in the recovery process if DB2 should abend.

CI. control interval

CICS. Represents (in this publication) CICS/OS/VS and CICS/MVS

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.

CICS/MVS. Customer Information Control System/Multiple Virtual Storage

CICS/OS/VS. Customer Information Control System/Operating System/Virtual Storage

CLIST. command list

clustering index. An index that determines how rows are physically ordered in a table space.

cold start. A process by which DB2 restarts without processing any log records.

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.

command. In DB2, a DB2 operator command or a DSN subcommand. Distinct from an SQL statement.

command recognition character (CRC). A character that permits an MVS console operator or IMS/VS subsystem user to route DB2 commands to specific DB2 subsystems.

commit. An operation that terminates a unit of recovery. A commit releases all locks. Data that was changed is now consistent.

commit point. A point in time when data is considered consistent.

concurrency. The shared use of resources by multiple application processes at the same time.

conditional restart. A DB2 restart that is directed by a user-defined conditional restart control record (CRCR).

CRC. command recognition character

CRCR. conditional restart control record

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 used by an application program 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. The isolation level that provides maximum concurrency. With cursor stability, a unit of

work holds locks only on its uncommitted changes and the current row of each of its cursors.

database. A collection of table spaces and index spaces.

database administrator (DBA). An individual responsible for the design, development, operation, safeguarding, maintenance, and use of a database.

database descriptor (DBD). An internal representation of DB2 database definition which reflects the data definition found in the DB2 catalog. The objects defined in a database descriptor are table spaces, tables, indexes, index spaces, and relationships.

database request module (DBRM). A data set member created by the DB2 precompiler that contains information about SQL statements. DBRMs are used in the bind process.

data type. An attribute of columns, literals, and host variables.

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

DBCS. double-byte character set

DBD. database descriptor

DBID. database identifier

DBRM. database request module

DB2 catalog. DB2-maintained tables that contain descriptions of DB2 objects such as tables, views, and indexes.

DB2 Interactive (DB2I). The DB2 facility that provides for the execution of SQL statements, DB2 (operator) commands, programmer commands, and utility invocation.

DB2I. DATABASE 2 Interactive

DCLGEN. declarations generator

DDL. data definition language

deadlock. Unresolvable contention for the use of a resource such as a table or 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.

default value. A predetermined value, attribute, or option that is assumed when no other is explicitly specified.

directory. The system database that contains internal objects such as database descriptors and skeleton cursor tables.

double-byte character set (DBCS). A set of characters used by national languages such as Japanese and Chinese that have more symbols than can be represented by a single-byte. Each character is two bytes in length, and therefore requires special hardware to be displayed or printed.

DSN. (1) The default DB2 subsystem name. (2) The name of DB2's TSO command processor. (3) The first three characters of DB2 module and macro names.

DSN command processor. The DB2 component that processes DB2 subcommands (such as BIND, RUN, etc).

duration. A number that represents an interval of time. See *labeled duration*, *date duration*, and *time duration*.

DXT. Data Extract

EBCDIC. extended binary coded decimal interchange code

ESDS. entry sequenced data set

EUR. IBM European Standards

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.

function. A scalar function or column function. Same as *built-in function*.

GTF. generalized trace facility

host language. Any programming language in which you can embed SQL statements.

host program. An application program written in a host language that contains embedded SQL statements.

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/VS attachment facility. A DB2 subcomponent that uses MVS Subsystem Interface (SSI) protocols and cross-memory linkage to process requests from

IMS/VS to DB2 and to coordinate resource commitment.

IMS/VS Resource Lock Manager (IRLM). An MVS subsystem used by DB2 to control communication and database locking.

IMS/VS. Information Management System/Virtual Storage

in-abort. A status of a unit of recovery. If DB2 fails after a unit of recovery enters abort processing but before completing that processing, it continues to back out the updates to its unit of recovery when it is next restarted. These units of recovery are termed *in-abort*.

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.

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 this unit of recovery is to be committed or rolled back. At emergency restart, if DB2 does not have the information needed to make this decision, its unit of recovery is *indoubt* until DB2 obtains this information from the coordinator.

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 when it is restarted. These units of recovery are termed *inflight*.

IRLM. IMS/VS 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* and *repeatable read*.

ISPF. Interactive System Productivity Facility

ISPF/PDF. Interactive System Productivity Facility/Program Development Facility

JCL. job control language

JIS. Japanese Industrial Standard

K. kilobyte (1024 bytes)

KSDS. key sequenced data set

labeled duration. A number that represents a duration of years, months, days, hours, minutes, seconds, or microseconds.

locking. The process by which integrity of data is ensured. Locking prevents concurrent users from accessing inconsistent data.

log. A collection of records that describe the events that occur during DB2 execution and their sequence. The information thus recorded is used for recovery in the event of a failure during DB2 execution.

log initialization. The first phase of restart processing during which DB2 attempts to locate the current end of the log.

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 will be written.

menu. A displayed list of available functions for selection by the operator. Sometimes called a menu panel.

mixed data string. A character string that may contain both single-byte and double-byte characters.

MVS/XA. Multiple Virtual Storage/Extended Architecture

null. A special value that indicates the absence of information.

OBID. data object identifier

page. A unit of storage within a table space (4K or 32K) or index space (4K). In a table space, a page contains one or more rows of a table.

panel. In computer graphics, a predefined display image that defines the locations and characteristics of display fields on a display surface - sometimes called a menu or menu panel.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data. Synonymous with program library.

partitioned table space. A table space subdivided into parts (based upon index key range), each of which may be processed by utilities independently.

plan name. The name of an application plan.

precompilation. A processing of application programs containing SQL statements that takes place before compilation. SQL statements are replaced with statements that will be 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.

QMF. Query Management Facility

RACF. OS/VS2 MVS Resource Access Control Facility

RBA. relative byte address

RCT. resource control table (CICS attachment facility)

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 their sequence. The information recorded is used for recovery in the event of a failure during DB2 execution.

recovery pending. This condition prevents SQL access to a table space or index space that may need to be recovered.

relative byte address (RBA). The offset of a data record or control interval from the beginning of the storage space allocated to the data set or file to which it belongs.

repeatable read. 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.

RO. read-only access

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 enclosed in parentheses.

service aid. A procedure or program used to determine and correct system problems.

SMF. system management facility

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. Structured Query Language. A language that can be used within host programming languages or interactively to access data and to control access to resources.

SQL escape character. See *escape character*.

SQL string delimiter. A symbol used to enclose an SQL string constant. The SQL string delimiter is the apos-

trope ('), except in COBOL applications, in which case the symbol (either an apostrophe or a quotation mark) may be assigned by the user.

SQL/DS. SQL/Data System

SQLCA. SQL communication area

SQLDA. SQL descriptor area

SSI. MVS subsystem interface

storage group. A named set of DASD volumes on which DB2 data can be stored.

system administrator. The person having the second highest level of authority within DB2. System administrators make decisions about how DB2 is to be used and implement those decisions by choosing system parameters. They monitor the system and change its characteristics to meet changing requirements and new data processing goals.

table. A named data object consisting of a specific number of columns and some number of unordered rows.

table space. A page set used to store the records of one or more tables.

thread. The DB2 structure that describes an application's connection, traces its progress, provides resource function processing capability, and delimits

its accessibility to DB2 resources and services. Most DB2 functions execute under a thread structure.

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.

timestamp. A seven-part value that consists of a date and time expressed in years, months, days, hours, minutes, seconds, and microseconds.

trace. A DB2 facility that provides the ability to monitor and collect DB2 monitoring, auditing, performance, accounting, statistics, and serviceability (global) data.

TSO. Time Sharing Option. A subsystem of MVS.

TSO attachment facility. A DB2 facility consisting of the DSN command processor and DB2I. Applications that are not written for the CICS or IMS/VS environments can run under the TSO attachment facility.

unique index. An index which ensures that no identical key values are stored in a table.

UT. utility-only access

utility. A program used to maintain and manipulate system and user table spaces and indexes.

value. The smallest unit of data manipulated in SQL.

VSAM. Virtual Storage Access Method

Bibliography

- *CICS/MVS Application Programmer's Reference*, SC33-0512
- *CICS/MVS Installation Guide*, SC33-0506
- *CICS/MVS Operations Guide*, SC33-0510
- *CICS/VS Application Programmer's Reference Manual (Command Level)*, SC33-0241
- *CICS/OS/VS: Installation and Operations Guide*, SC33-0071
- *DFSORT Application Programming: Guide*, SC33-4035
- *IBM DATABASE 2 Version 2 General Information*, GC264373
- *IBM DATABASE 2 Version 2 System and Database Administration Guide*, SC264374
- *IBM DATABASE 2 Version 2 Licensed Program Specification*, GC264375
- *IBM DATABASE 2 Version 2 SQL Reference*, SC264376
- *IBM DATABASE 2 Version 2 Application Programming Guide*, SC264377
- *IBM DATABASE 2 Version 2 Messages and Codes*, SC264379
- *IBM DATABASE 2 Version 2 Reference Summary*, SX263771
- *IBM DATABASE 2 Version 2 Diagnosis Guide and Reference*, LY279536
- *IBM System/370 Principles of Operations*, GA22-7000
- *IMS/VS Version 2 Operator's Reference Manual*, SC26-4175
- *MVS/IXA Integrated Catalog Administration: Access Method Services Reference*, GC26-4135
- *MVS/IXA TSO Extensions TSO Guide to Writing a Terminal Monitor Program or Command Processor*, SC28-1136
- *MVS/ESA Operations: System Commands*, GC28-1826
- *OS/VS2 MVS System Programming Library: Service Aids*, GC28-0674
- *Operator's Library: OS/VS2 MVS System Commands*, GC38-0229
- *SQL/DATA SYSTEM Data Base Services Utility for VM/System Product*, SH24-5069
- *TSO Extensions Command Language Reference*, SC28-1370

Index

Special Characters

- * See asterisk
- :
- See colon
- ' in DB2 command 14
- = in DB2 command 14
- " in DB2 command 14

A

- ABEND subcommand (DSN) 60
- ABORT option of RECOVER INDOUBT command 114
- ACCESS
 - option of START DATABASE command 121
 - option of START DB2 command 124
- access path
 - BIND chooses 19
 - RUNSTATS output 288
- ACCTG
 - option of DISPLAY TRACE command 55
 - option of START TRACE command 131
 - option of STOP TRACE command 145
- ACQUIRE
 - option of BIND subcommand 32
 - option of DSNH CLIST 73
 - option of REBIND subcommand 109
- ACTION
 - option of BIND subcommand 30
 - option of DCLGEN subcommand 37
 - option of DSNH CLIST 73
 - option of RECOVER INDOUBT command 113
- ACTIVE
 - option of DISPLAY DATABASE command 47
 - option of DISPLAY THREAD command 52
- active state of utility execution 167
- ADD
 - option of BIND subcommand 30
 - option of DCLGEN subcommand 37
 - option of DSNH CLIST 73
- AFTER option of DISPLAY DATABASE command 47
- AGE option of MODIFY utility 252
- ALL
 - option of /CHANGE command 34
 - option of /DISPLAY command 42
 - option of /START command 119
 - option of /STOP command 137
 - option of CHECK 181
 - option of COPY utility 186
 - option of MERGECOPY utility 248
 - option of MODIFY utility 252
 - option of RECOVER utility 259
 - option of REPORT utility 286

- ALL (*continued*)
 - option of RUNSTATS utility 290
- ALLDUMPS option of DIAGNOSE utility 191
- ALLOCATE
 - option of BIND subcommand 32
 - option of DSNH CLIST 73
 - option of REBIND subcommand 110
- APOST
 - option of DCLGEN subcommand 38
 - option of DSNH CLIST 76
- apostrophe 14
 - in DB2 commands 14
- APOSTSQL option of DSNH CLIST 82
- application plan
 - BIND builds 19
 - creating 18
 - deleting 96
 - naming 30
 - rebinding 107
 - reevaluating 18
- application program
 - /START command 119
 - /STOP command 137
 - application plan 19
 - debugging 60
 - passing parameters to 115
 - processed by DSNH CLIST 71
 - return code from validation 18
- archive log data set 176
- ARCHIVE option of CHANGE LOG INVENTORY 172
- ASMLIB option of DSNH CLIST 73
- ASMHLIB option of DSNH CLIST 73
- ASMLIB option of DSNH CLIST 73
- ASMLOAD option of DSNH CLIST 73
- ASMLIB option of DSNH CLIST 73
- ASMLOAD option of DSNH CLIST 73
- assembler language program
 - See application program
- asterisk
 - operator in DB2 commands 14
- AUDIT
 - &I2@V29.
 - in DISPLAY TRACE 55
 - option of START TRACE command 131
 - option of STOP TRACE command 145
- AUTHID
 - option of DISPLAY TRACE command 56
 - option of START TRACE command 133
 - option of STOP TRACE command 146
- automatic rebind 18

B

BACKOUT option of CHANGE LOG INVENTORY 175
backward log recovery 175
BATCH option of DSNH CLIST 81
BIND
 option of BIND subcommand 31
 option of DSNH CLIST 74, 83
 option of REBIND subcommand 109
 subcommand of DSN 28–33
bind process
 uses RUNSTATS output 288, 291
binding 18
 automatic rebind 18
 dynamic bind 18
 functions 18
 initiating 28
 inputs 18
 output to catalog tables 19
 rebinding 107
 step in DSNH processing 71
 types 18
blank 14
 in DB2 commands 14
BLIB option of DSNH CLIST 74
BLKSIZE parameter
 COPY output 188
 option of MERGECOPY output 249
block size
 COPY output 188
 option of MERGECOPY output 249
BMEM option of DSNH CLIST 74
bootstrap data set (BSDS)
 See BSDS (bootstrap data set)
BSDS (bootstrap data set)
 in DD statements for Print Log Map 254
 recovering 112
 updating 170

C

C option of DCLGEN subcommand 37
CANCEL option of CHANGE LOG INVENTORY 174
CATALOG option of CHANGE LOG INVENTORY 174
CATALOG option of RECOVER utility 261
catalog table
 See catalog tables
catalog tables
 BIND output to 19
catalog, VSAM
 See VSAM, catalog
CHANGE
 option of COPY utility 187
 option of RUNSTATS utility 291
/CHANGE command 34, 35
Change Log Inventory utility 170, 177
 description 155
CHAR
 option of LOAD utility 235

check pending restriction
 with -DISPLAY DATABASE 48
CHECK utility 178, 184
 description 153
 effects of TERM on 148
checking an index 178
CHKPTRBA option of CHANGE LOG INVENTORY 174
CICS attachment facility command
 DSNC 63
 DSNC DISCONNECT 64
 DSNC DISPLAY 65–66
 DSNC MODIFY 67–68
 DSNC STOP 69
 DSNC STRT 70
 option
 DESTINATION 67
 FORCE 69
 PLAN 65
 QUIESCE 69
 STATISTICS 65
 TRANSACTION 65, 67
CICS option of DSNH CLIST 81
CICS translation step in DSNH processing 71
CICSOPT option of DSNH CLIST 74
CICSPRE option of DSNH CLIST 74
CICSXLAT option of DSNH CLIST 75
CLASS
 option of DISPLAY TRACE command 56
 option of START TRACE command 133
 option of STOP TRACE command 146
CLIB option of DSNH CLIST 75
CLIST
 See TSO CLIST
COBOL application program
 See application program
COBOL option of DCLGEN subcommand 37
COB2 option of DCLGEN subcommand 38
COB2CICS option of DSNH CLIST 75
COB2LIB option of DSNH CLIST 75
COB2LOAD option of DSNH CLIST 75
cold start
 specifying for conditional restart 173
colon
 operator in DB2 commands 14
COMM
COMMA
 option of DSNH CLIST 76
command name in DB2 command 12
command option
 in CICS attachment facility
 See CICS attachment facility command, option
 in DB2
 See DB2 command option
 in IMS/VS
 See IMS/VS command option
 in MVS
 See MVS command option

command recognition character (CRC)
 See CRC (command recognition character)

comment
 in DCLGEN output 39
 in DSN subcommands 60
 in SYSIN records 176
 option of DISPLAY TRACE command 55
 option of START TRACE command 131
 option of STOP TRACE command 145

COMMIT
 option of BIND subcommand 32
 option of DSNH CLIST 81
 option of REBIND subcommand 110
 option of RECOVER INDOUBT command 114

commit point
 cursor stability 31, 109
 during REPAIR 283
 terminating utility at 147

compilation
 step in DSNH processing 71

COMPILE
 option of DSNH CLIST 75

conditional restart control record
 effect on restart 124
 in CHANGE LOG INVENTORY utility 174
 status printed by PRINT LOG MAP utility 254

CONLIST
 option of DSNH CLIST 75
 option of DSNU CLIST 90

connection
 displaying status of 42
 DSNH DISPLAY command 65
 information displayed by DISPLAY THREAD 51
 information displayed by MODIFY
 irlmproc,STATUS 101
 made by START command 119
 RETRY option of DSN command 61
 terminating 137

connection-name
 in DISPLAY THREAD 52
 in RECOVER INDOUBT 113

CONSTRAINTS option of LOAD utility 228

CONTINUE
 option of RECOVER utility 260
 option of REORG utility 268

CONTINUEIF option of LOAD utility 229

CONTROL
 option of DSNH CLIST 75
 option of DSNU CLIST 90

control record, conditional restart
 See conditional restart control record

COPTION option of DSNH CLIST 75

copy pending restriction
 turned on by -TERM 188
 turned on by TERM 148
 with COPY 187
 with LOAD 239
 with REORG 270

COPY utility 185, 189
 description 153
 effects of TERM on 148
 example of control statement in utility JCL 161
 examples 158

COPYDDN
 option of COPY utility 186
 option of MERGECOPY utility 249

COPYDSN option of DSNU CLIST 90

copying data sets within a table space in parallel 188

COPY1 option of CHANGE LOG INVENTORY 173

COPY1VOL option of CHANGE LOG INVENTORY 173

COPY2 option of CHANGE LOG INVENTORY 173

COPY2VOL option of CHANGE LOG INVENTORY 173

correlation ID
 identifies thread to be recovered 114

CP option of RUN subcommand 115

CRC (command recognition character) 12

CREATE option of CHANGE LOG INVENTORY 174

creating
 JCL for a utility 166

CRESTART option of CHANGE LOG INVENTORY 173

CS
 option of BIND subcommand 31
 option of DSNH CLIST 78
 option of REBIND subcommand 109

CSRONLY option of CHANGE LOG INVENTORY 175

CURRENT
 option of REPORT utility 286

CURRENT option of DSNU CLIST 91

cursor stability
 in BIND subcommand 31
 in DSNH CLIST 78
 in REBIND subcommand 109

CYLINDER option of DSNH CLIST 82

D

DATA
 option of CHECK 180
 option of LOAD 226
 option of REPAIR 278, 279

data set
 changing space parameters during
 reorganization 271
 disposition in DB2 utility 157
 needed by LOAD 244
 needed by RECOVER. 263
 VSAM 172

data sets
 used by utilities 156

data type
 for INTO TABLE 237
 in field specifications with LOAD 235

database
 displaying status of 45
 option of DISPLAY DATABASE command 45
 reserved names 139

database (*continued*)
 starting 120
 stopping 138
 database request module (DBRM)
 input to BIND 18
 named in BIND command 30
 DATE data type
 DATE option of DSNH CLIST 76
 DATE option of MODIFY utility 252
 DBM1 option of START DB2 command 124
 DBRM
 See database request module (DBRM)
 DBRMLIB option of DSNH CLIST 76
 DB2
 internal limits 297
 starting 123
 stopping 140
 DB2 catalog tables
 See catalog tables
 DB2 command
 DISPLAY DATABASE 45–49
 DISPLAY RLIMIT 50, 128
 DISPLAY THREAD 51–52
 DISPLAY TRACE 53–57
 DISPLAY UTILITY 58–59
 general description 12
 RECOVER BSDS 112
 RECOVER INDOUBT 113–114
 START DATABASE 120–122
 START DB2 123–125
 START RLIMIT 128
 START TRACE 129–136
 STOP DATABASE 138–139
 STOP DB2 140
 STOP RLIMIT 142
 STOP TRACE 143–146
 TERM UTILITY 147–148
 DB2 command option
 ABORT 114
 ACCESS 121, 124
 ACCTG
 in DISPLAY TRACE 55
 in START TRACE 131
 in STOP TRACE 145
 ACTION 113
 ACTIVE 47, 52
 AFTER 47
 AUDIT
 in START TRACE 131, 132
 in STOP TRACE 145
 AUTHID
 in DISPLAY TRACE 56
 in START TRACE 133
 in STOP TRACE 146
 CLASS
 in DISPLAY TRACE 56
 in START TRACE 133
 in STOP TRACE 146

DB2 command option (*continued*)
 COMMENT
 in DISPLAY TRACE 55
 in START TRACE 131
 in STOP TRACE 145
 COMMIT 114
 DBM1 124
 DEST
 in DISPLAY TRACE 55
 in START TRACE 132
 in STOP TRACE 145
 DSNZPARM 123
 FORCE 122, 140
 GLOBAL
 in DISPLAY TRACE 55
 in START TRACE 131
 in STOP TRACE 145
 GTF
 in DISPLAY TRACE 55
 in START TRACE 132
 in STOP TRACE 145
 ID 114
 INDOUBT 52
 LIMIT 47
 LOCKS 47
 MAINT 124
 MODE 140
 MSTR 124
 NID 114
 option of DISPLAY TRACE command 55
 PARM 123
 PERFM
 in DISPLAY TRACE 55
 in START TRACE 131
 in STOP TRACE 145
 PLAN
 in DISPLAY TRACE 56
 in START TRACE 133
 in STOP TRACE 146
 QUIESCE 140
 RES
 in DISPLAY TRACE 55
 in START TRACE 132
 in STOP TRACE 145
 RESTRICT 47
 RMID
 in DISPLAY TRACE 56
 in START TRACE 135
 in STOP TRACE 146
 RO 122
 RW 122
 SMF
 in DISPLAY TRACE 55
 in START TRACE 132
 in STOP TRACE 145
 SPACENAM
 in DISPLAY DATABASE 47
 in START DATABASE 121
 in STOP DATABASE 139

DB2 command option (*continued*)

- SRV
 - in DISPLAY TRACE 55
 - in START TRACE 132
 - in STOP TRACE 145
- STAT
 - in DISPLAY TRACE 55
 - in START TRACE 131
 - in STOP TRACE 145
- TNO 56, 146
- TYPE 52
- USE 47
- UT 122

DB2 precompiler

- effect of OPTIONS option of DSNH CLIST 79
- effect of PLOAD option of DSNH CLIST 79
- effect of PLIB option of DSNH CLIST 79
- effect of PRECOMP option of DSNH CLIST 80
- effect of SOURCE option of DSNH CLIST 82
- effect of SQLDELIM option of DSNH 82
- effect of XREF option of DSNH 83
- produces DBRM 18
- produces members for BIND 30

DB2 UTILITIES panel 158

DB2 utility statement

- CHECK 178–184
- COPY 185–189
- DIAGNOSE 190–193
- LOAD 224–246
- MERGECOPY 247–250
- MODIFY 251–253
- QUIESCE 255–256
- RECOVER 257–265
- REORG 266–272
- REPAIR 273–284
- REPORT 285–287
- RUNSTATS 288–294
- STOSPACE 295–296

DB2I (DB2 Interactive)

- option of DSNU CLIST 90
- used to invoke utilities 158

DCLGEN

- DCLGEN DSN subcommand 36–41
- description 36

DD statements

- for a DB2 utility 161
- used by utilities 156

DEADLOCK option of START irlmproc command 126

DEALLOCATE

- option of BIND subcommand 32
- option of DSNH CLIST 81
- option of REBIND subcommand 110

DECIMAL

- option of DSNH CLIST 76
- option of LOAD utility 235

DECIMAL data type

- input data format 235

declarations generator

- See DCLGEN

DEFAULTIF option of LOAD utility 238

DELETE

- option of CHANGE LOG INVENTORY 172
- option of MODIFY utility 252
- option of REPAIR utility 279

deleting

- IMS units of recovery 34
- log data sets with errors 176

DELIMIT option of DSNH CLIST 76

DEST

- option of DISPLAY TRACE command 55
- option of START TRACE command 132
- option of STOP TRACE command 145

DESTINATION

- option of DSNH MODIFY command 67

DEVT

- option of COPY utility 186
- option of MERGECOPY utility 248

DFSORT

- allocates data sets for REORG 268
- determines values for SORTDEVT in CHECK 180
- determines values for SORTDEVT in LOAD 229
- messages from 243, 271

DIAGNOSE utility 190

- description 153

discard data set for LOAD 228, 240

DISCARD option of LOAD utility 228

DISCARDS option of LOAD utility 228

DISCDSN option of DSNU CLIST 90

/DISPLAY command 42, 44

DISPLAY DATABASE command 45, 49

DISPLAY RLIMIT command 50

DISPLAY THREAD command 51, 52

DISPLAY TRACE command 53–57

DISPLAY UTILITY command 58, 59, 167

displaying

- information about DB2 utilities 167
- trace activity 53

DNAME

- double-byte character
 - in DCLGEN table names 39

DSN command 60–62

DSN subcommand 107, 111

- BIND 28–33
- DCLGEN 36–41
- DSN 60–62
- END 95
- FREE 96–97
- REBIND 107–111
- RUN 115–116
- SPUFI 117

DSN subcommand option

- ACTION 30, 37
- ADD 30, 37
- APOST 38
- BIND 31, 109

DSN subcommand option (*continued*)

- C 37
- COBOL 37
- COB2 38
- CP 115
- CS 31, 109
- FLAG
 - in BIND 32
 - in FREE 97
 - in REBIND 108, 109
- ISOLATION 31, 109
- LABEL 38
- LANGUAGE 37
- LIBRARY
 - in BIND 30
 - in DCLGEN 37
 - in RUN 115
- MEMBER 30
- NAMES 38
- OWNER 30
- PARMS 115
- PLAN
 - in BIND 30
 - in FREE 96
 - in REBIND 108
 - in RUN 115
- PLI 37
- PROGRAM 115
- QUOTE 38
- REPLACE 30, 37
- RETAIN 31
- RETRY 61
- RR 31, 109
- RUN 31, 109
- STRUCTURE 38
- SYSTEM 61
- TABLE 37
- TEST 61
- VALIDATE 31, 109

DSNAME option of CHANGE LOG INVENTORY 173

DSNC command (CICS) 63

DSNC DISCONNECT command (CICS) 64

DSNC DISPLAY command (CICS) 65–66

DSNC MODIFY command (CICS) 67–68

DSNC STOP command (CICS) 69

DSNC STRT command (CICS) 70

DSNH command 71–87

DSNJU003 170

DSNJU004 254

DSNU CLIST 159

- used to invoke utilities 160

DSNU command 88–94

DSNUM

- option of COPY utility 186
- option of MERGECOPY utility 248
- option of MODIFY utility 252
- option of RECOVER utility 259
- option of REPAIR 275

DSNUM (*continued*)

- option of REPORT utility 286
- DSNUPROC JCL procedure 163
 - syntax and parameter explanations 163
- DSNZPARM option of START DB2 command 123
- DSN1CHKR 194
- DSN1COPY
 - authorization 200
 - description 155
 - environment 200
 - examples 206
 - syntax and parameters 201
- DSN1LOGP
 - authorization 209
 - control statement and keywords 214
 - description 155
 - environment 209
 - examples 217
 - on active log data (no BSDS) 211
 - on archive log data (no BSDS) 212
 - syntax and parameters 210
 - with BSDS 211
 - with SUMMARY option 213
- DSN1PRNT
 - authorization 221
 - description 156
 - environment 221
 - sample JCL 223
 - syntax and parameters 222
- DUMP option of DSN1CHKR service aid 195
- DUMP option of REPAIR utility 279
- dynamic bind 18

E

EDIT

- option of DSNU CLIST 91

edit routine

- used by LOAD utility 224
- used by REORG utility 268

editing

- JCL for a utility 161

END DSN subcommand 95

END option of DIAGNOSE utility 191

ENDRBA option of CHANGE LOG INVENTORY 173

ENFORCE option of LOAD utility 228

ENTRY option of DSNH CLIST 76

equal sign in DB2 command 14

ERRDDN option of LOAD utility 228

error

- active log data set 176

ERROR RANGE option of RECOVER utility 260

escape character in SQL

- with APOST and QUOTE 38

EXEC statement

- for a DB2 utility 161, 166

executing

- SQL statements 18

executing (*continued*)
 utilities by the DSNU CLIST 88
EXPLAIN
 option of BIND subcommand 32
 option of DSNH CLIST 77
 option of REBIND subcommand 110
EXTERNAL
 option of LOAD utility 236, 237

F
 F irlmproc,ABEND command (MVS IRLM) 98
 F irlmproc,START command (MVS IRLM) 99–100
 F irlmproc,STATUS command (MVS IRLM) 101–102
 F irlmproc,STOP command (MVS IRLM) 103–104
 fall back
 incremental copies 188
 with RECOVER 264
 field procedure
 used by REORG utility 268
 with LOAD 244
FLAG
 option of BIND subcommand 32
 option of DSNH CLIST 77
 option of FREE subcommand 97
 option of REBIND subcommand 109
FLOAT
 option of LOAD utility 236
FLOAT data type
 input data format 236
FORCE
 option of DSNC STOP command 69
 option of START DATABASE command 122
 option of STOP DB2 command 140
FORMAT option of DSN1CHKR service aid 195
FORMAT option of LOAD utility 227
FORTLIB option of DSNH CLIST 77
FORTLOAD option of DSNH CLIST 77
FORTTRAN application program
 See application program
 forward log recovery 175
FORWARD option of CHANGE LOG INVENTORY 175
FREE
 DSN subcommand 96–97
 free space
 left by REORG 269
FULL option of COPY utility 187

G
 Generalized Trace Facility (GTF)
 See GTF (Generalized Trace Facility)
GLOBAL
 option of DISPLAY TRACE command 55
 option of START TRACE command 131
 option of STOP TRACE command 145
GRAPHIC
 option of LOAD utility 236

GTF (Generalized Trace Facility)
 option of DISPLAY TRACE command 55
 option of START TRACE command 132
 option of STOP TRACE command 145
GTRACE
 option of MODIFY irlmproc,START command 99
 option of MODIFY irlmproc,STOP command 103

H
 HASH option of DSN1CHKR service aid 195
 help
 getting help while using the DB2 UTILITIES
 panel 160
 HELP PF key 160
 HOST option of DSNH CLIST 77

I
 ID option of RECOVER INDOUBT command 114
 image copy
 full 185
 incremental 185
 parallel 185
IMAGECOPY
IMS option of DSNH CLIST 81
IMS/VS
 events, tracing 149
IMS/VS command
 /CHANGE 34–35
 /DISPLAY 42–44
 /SSR 118
 /START 119
 /STOP 137
 /TRACE 149–150
IMS/VS command option
 ALL
 in /CHANGE 34
 in /DISPLAY 42
 in /START 119
 in /STOP 137
 LOG 149
 OASN 34, 42
 RESET 34
 SUBS 149
SUBSYS
 in /CHANGE 34
 in /DISPLAY 42
 in /START 119
 in /STOP 137
 SUBSYS ALL 42
IMSPRE option of DSNH CLIST 78
 in-abort unit of recovery 175
 in-commit unit of recovery 175
INCLUDE SQL statement
 with DCLGEN output 39
 inconsistent data indicator 278

INDDN option of LOAD utility 226
 index
 checking consistency with data 178
 option of CHECK utility 179
 option of RECOVER utility 261
 option of REORG utility 267
 option of REPAIR utility 275, 277
 option of RUNSTATS utility 290
 recovering 261
 INDEXES option of REPAIR 275
 INDOUBT option of DISPLAY THREAD command 52
 indoubt thread
 recovering 113–114
 indoubt unit of recovery
 See unit of recovery, indoubt
 INDSN option of DSNU CLIST 90
 inflight unit of recovery 175
 input data set for LOAD 240
 INPUT option of DSNH CLIST 78
 INTEGER
 option of LOAD utility 237
 INTEGER data type
 input data format 237
 INTO TABLE
 option of LOAD 230
 invoking
 the DSNU CLIST 160
 utilities
 by DB2I 158
 by DSNU CLIST 160
 by supplied JCL 163
 by your own JCL 166
 IRLMID option of START irImproc command 127
 IRLNMN option of START irImproc command 127
 ISOLATION
 option of BIND subcommand 31
 option of DSNH CLIST 78
 option of REBIND subcommand 109
 ISPF
 Primary Option Menu 158
 ITRACE
 option of MODIFY irImproc,START command 99
 option of MODIFY irImproc,STOP command 103
 option of START irImproc command 127

J

JCL
 creating for DB2 utility 166
 example for running utility 161
 for DSNUPROC 88
 generated by DB2I 161
 generating to run a DB2 utility 160
 supplied, to invoke utilities 163
 JOB statement
 for a DB2 utility 161

K

KEY option of REPAIR utility 276
 keyword
 in CICS attachment facility commands
 See CICS attachment facility command, option
 in DB2 commands 12
 See also DB2 command option
 in DSN command and subcommands
 See DSN subcommand option
 in IMS/VS commands
 See IMS/VS command option
 in MVS commands
 See MVS command option
 in TSO CLISTS
 See TSO CLIST, options
 in utility statements 15
 See also utility option
 negation 14

L

LABEL
 option of DCLGEN subcommand 38
 LANGUAGE
 option of DCLGEN subcommand 37
 LEAVE option of DSNH CLIST 80, 81, 82
 LENGTH
 option of REPAIR 280
 LIB option of DSNUPROC 163
 LIBRARY
 option of BIND subcommand 30
 option of DCLGEN subcommand 37
 option of RUN subcommand 115
 limit
 DB2 297
 SQL 297
 LIMIT option of DISPLAY DATABASE command 47
 LINECOUNT option of DSNH CLIST 78
 LINK option of DSNH CLIST 78
 link-editing step in DSNH processing 71
 LIST
 keyword in DSNH CLIST 75
 option of DSNU CLIST 90
 LLIB option of DSNH CLIST 78
 LOAD
 LOAD utility 224, 246
 description 153
 effects of TERM on 148
 loading
 data 224
 LOCAL
 option of START irImproc command 127
 LOCATE INDEX option of REPAIR utility 277
 LOCATE TABLESPACE option of REPAIR utility 276
 LOCKS option of DISPLAY DATABASE command 47
 log
 option of /TRACE command 149
 option of LOAD utility 226

log (*continued*)
 option of REORG utility 267
 option of REPAIR utility 274
 recovery
 backward 175
 forward 175
 truncation 177
 log data set
 adding 170
 deleting 170
 printing map of 254
 printing names of 254
 with I/O error 176
 LOG PRINT UTILITY (DSNJU004) 254
 log utility statement
 CHANGE LOG INVENTORY (DSNJU003) 170–177
 PRINT LOG MAP (DSNJU004) 254
 LOPTION option of DSNH CLIST 79

M

MACRO option of DSNH CLIST 79
 MAINT option of START DB2 command 124
 MAP option of DSN1CHKR service aid 196
 MAPDDN option of LOAD utility 228
 MAXCSA option of START irlmproc command 127
 media damage, avoiding with RECOVER 263
 MEMBER option of BIND subcommand 30
 MERGECOPY utility 247, 250
 description 153
 effects of TERM on 148
 message
 from BIND 30
 effect of FLAG 32
 for plan name beginning with DSN 30
 with VALIDATE 31
 from CHECK 182
 from DCLGEN 38
 from DFSORT 243, 271
 from DISPLAY THREAD with ACTIVE 52
 from DISPLAY TRACE 56
 from DISPLAY UTILITY 58, 59
 from DSN during attention processing 61
 from DSNH CLIST 77
 from DSNU 93
 from FREE 97
 from LOAD 226
 from MERGE 248
 from MODIFY irlmproc,START 99
 from MODIFY irlmproc,STATUS 101
 from REBIND 109
 from RECOVER 263
 from REORG 270
 from RUN 116
 MODE
 option of STOP DB2 command 140
 MODIFY irlmproc command
 ABEND 98

MODIFY irlmproc command (*continued*)
 START 99–100
 STATUS 101–102
 STOP 103–104
 MODIFY utility 251, 253
 description 153
 effects of TERM on 148
 monitoring
 utility status 167
 MSTR option of START DB2 command 124
 MVS command option
 DEADLOK 126
 GTRACE 99, 103
 IRLMID 127
 IRLMNM 127
 ITRACE 127
 in MODIFY irlmproc,START 99
 in MODIFY irlmproc,STOP 103
 LOCAL 127
 MAXCSA 127
 NODUMP 98
 PC 127
 PTBTRACE 99, 103
 SCOPE 127
 START irlmproc 126–127
 STOP irlmproc 141
 TRACE 99, 103
 MVS commands affecting the IRLM
 MODIFY irlmproc,ABEND 98
 MODIFY irlmproc,START 99–100
 MODIFY irlmproc,STATUS 101–102
 MODIFY irlmproc,STOP 103–104

N

NAME
 option of CHECK utility 180
 NAMES
 option of DCLGEN subcommand 38
 negation of keyword 14
 network-id 114
 NEWCAT option of CHANGE LOG INVENTORY 173
 NEWCOPY option of MERGECOPY utility 249
 NEWLOG option of CHANGE LOG INVENTORY 172
 NID option of RECOVER INDOUBT command 114
 NOCHECKPEND option of REPAIR 275
 NOCOPYPEND option of REPAIR 275
 NODUMP option of MODIFY irlmproc,ABEND
 command 98
 NOPASSWD option of CHANGE LOG INVENTORY 175
 NORCVRPEND option of REPAIR 275
 NULLIF option of LOAD utility 238

O

OASN

- option of /CHANGE command 34
- option of /DISPLAY command 42

OBJECT option of REPAIR utility 274

OFFSET

- option of REPAIR...DUMP 280
- option of REPAIR...REPLACE 278
- option of REPAIR...VERIFY 278

ONE option of DSNH CLIST 79

ONLY option of REORG utility 268

operand in DB2 command 12

optional parameter

- in syntax diagrams 8

OPTIONS option of DSNH CLIST 79

OUTNAME option of DSNH CLIST 79

OWNER

- option of REBIND subcommand 108

P

P irlImproc command 141

PAGE

- option of LOCATE INDEX 277
- option of RECOVER utility 260
- option of REPAIR utility 276

page number

- writing
- for REPAIR 283

PAGES

- option of REPAIR 280

parameter in utility statement 15

parameters, passing to application program 115

parentheses

- in DB2 commands 14

PARAM

- option of START DB2 command 123

PARMS

- option of DSNH CLIST 79
- option of RUN subcommand 115

parsing rules 11–15

- commands 11
- utility statements 15

PART

- option of LOAD utility 232
- option of REORG utility 267

partial recovery 264

PASS option of DSNH CLIST 79

PASSWORD

- on archive log data set 176
- option of CHANGE LOG INVENTORY 174

password, VSAM

- See VSAM, password

PAUSE option of REORG utility 268

PC option of START irlImproc command 127

PCLOAD option of DSNH CLIST 79

PENDING option of CHECK 180

PERFM

- option of DISPLAY TRACE command 55
- option of START TRACE command 131
- option of STOP TRACE command 145

PERIOD

- option of DSNH CLIST 76

phase

- DSNH processing 71
- utility 91

phase of utility execution

- for CHECK 183
- for COPY 189
- for LOAD 242
- for MERGECOPY 249
- for MODIFY 253
- for RECOVER 263
- for REORG 270
- for REPAIR 284
- for RUNSTATS 293
- for STOSPACE 296

PHASE option of DSNU CLIST 91

phases

- DB2 utility 167

PL/I application program 71

- See also application program

macro processing step for DSNH 71

PLAN

- option of BIND subcommand 30
- option of DISPLAY TRACE command 56
- option of DSNH CLIST 79
- option of FREE subcommand 96
- option of REBIND subcommand 108
- option of RUN subcommand 115
- option of START TRACE command 133
- option of STOP TRACE command 146

PLI option of DCLGEN subcommand 37

PLIB option of DSNH CLIST 79

PLILIB option of DSNH CLIST 80

PLILOAD option of DSNH CLIST 80

POSITION option of LOAD utility 234

PRECOMP option of DSNH CLIST 80

precompilation

- step in DSNH processing 71

precompiled statement

- allowable contents 18
- checked by BIND 18

precompiler

- See DB2 precompiler

Print Log Map utility 254

- description 155

PRINT option of DSNH CLIST 80

PROGRAM option of RUN subcommand 115

PSECSPAC option of DSNH CLIST 80

PSPACE option of DSNH CLIST 80

PTBTRACE

- option of MODIFY irlImproc,START command 99
- option of MODIFY irlImproc,STOP command 103

Q

QUIESCE

- option of DSNB STOP command 69
- option of STOP DB2 command 140

QUIESCE utility 255

- description 154

quotation mark in DB2 command 14

QUOTE

- option of DCLGEN subcommand 38
- option of DSNH CLIST 76

QUOTESQL option of DSNH CLIST 82

R

RCTERM option of DSNH CLIST 80

rebinding

- application plans 107
- automatic 18
- recommended after LOAD 240

RECDSN option of DSNU CLIST 91

recognition character 12

RECOVER BSDS command 112

RECOVER INDOUBT command 113, 114

RECOVER utility 257, 265

- description 154

- effects of TERM on 148

recovering

- bootstrap data sets 112
- data

by RECOVER 257

effect of MERGECOPY 249

REORG makes image copies invalid 188

with RECOVER 272

indexes 261

indoubt threads 113–114

partially 264

RECOVERY

backward log 175

forward log 175

log 175

option of MODIFY utility 251

option of REPORT 286

partial

See partial recovery

unit of

See unit of recovery

recovery pending restriction

with -DISPLAY DATABASE 48

with RECOVER 264

REFERENCE

option of COPY utility 187

option of RUNSTATS utility 291

RELEASE

option of BIND subcommand 32

option of DSNH CLIST 81

option of REBIND subcommand 110

REORG utility 266, 272

- description 154

REORG utility (*continued*)

- effects of TERM on 148

REPAIR utility 273, 284

- description 154

- effects of TERM on 148

repeatable read

- in BIND subcommand 31

- in DSNH CLIST 78

- in REBIND subcommand 109

REPLACE

option of BIND subcommand 30

option of DCLGEN subcommand 37

option of DSNH CLIST 73

option of LOAD utility 226

option of REPAIR utility 278

REPORT utility 285

- description 154

RES

option of DISPLAY TRACE command 55

option of START TRACE command 132

option of STOP TRACE command 145

RESET

option of /CHANGE command 34

option of REPAIR 278

resetting

indoubt units of recovery 34

restart control record, conditional

See conditional restart control record

RESTART option of DSNU CLIST 91

restarting

a utility by an EXEC statement 166

a utility through UTPROC 164

a utility using DB2I 162

cannot restart CHECK 183

CICS attachment facility 69

connections between IMS and a subsystem 119

COPY utility 188

during QUIESCE 256

LOAD utility 242

MERGECOPY utility 249

REORG utility 268, 270

REPAIR utility 284

status of DB2 resources in 123

terminated utility job steps 148

utilities 91

RESTRICT option of DISPLAY DATABASE

command 47

RESUME

option of LOAD utility 226

RETAIN

option of BIND subcommand 31

option of DSNH CLIST 81

RETRY option of DSN command 61

RID option of REPAIR utility 276

RMID (resource manager identifier)

option of DISPLAY TRACE command 56

option of START TRACE command 135

option of STOP TRACE command 146

RO option of START DATABASE command 122
 RR
 option of BIND subcommand 31
 option of DSNH CLIST 78
 option of REBIND subcommand 109
 RUN 116
 DSN subcommand 115–116
 option of BIND subcommand 31
 option of DSNH CLIST 81, 83
 option of REBIND subcommand 109
 RUNIN option of DSNH CLIST 81
 running
 interactive debugging programs 115
 step in DSNH processing 71
 utilities with DB2 153–169
 RUNOUT option of DSNH CLIST 82
 RUNSTATS utility 288, 294
 description 154
 effects of TERM on 148
 recommended after LOAD 240
 RW option of START DATABASE command 122

S

scanning rules 11–16
 commands 11
 utility statements 15
 SCOPE option of CHECK 180
 SCOPE option of START irlmproc command 127
 separator in DB2 command 12
 service aid
 SET INDEX option of REPAIR utility 274
 SET TABLESPACE option of REPAIR utility 274
 shift-in character
 in LOAD utility 234
 shift-out character
 in LOAD utility 234
 SHRLEVEL
 option of COPY utility 187
 option of RUNSTATS utility 290
 SIZE option of DSNUPROC 163
 SMALLINT
 option of LOAD utility 237
 SMALLINT data type
 input data format 237
 SMF (System Management Facility)
 option of DISPLAY TRACE command 55
 option of START TRACE command 132
 option of STOP TRACE command 145
 Sort
 See DFSORT
 SORTDEVT
 option of CHECK utility 180
 option of LOAD utility 229
 option of RECOVER utility 262
 option of REORG utility 268
 SORTLIB
 used by utilities 156

SORTNUM
 option of CHECK utility 180
 option of LOAD utility 229
 option of RECOVER utility 263
 option of REORG utility 268
 SORTWKnn
 used by utilities 156
 SOURCE option of DSNH CLIST 82
 space parameter for data set, changing 271
 space, free
 See free space
 SPACENAM
 option of DISPLAY DATABASE command 47
 option of START DATABASE command 121
 option of STOP DATABASE command 139
 SPACEUN option of DSNH CLIST 82
 SPUFI
 DSN subcommand 117
 SQL (Structured Query Language)
 limits 297
 SQL statement
 INCLUDE
 with DCLGEN output 39
 SQL/DS
 option of LOAD utility 227, 233
 SQLDELIM option of DSNH CLIST 82
 SRC (subsystem recognition character)
 as part of a command 12
 for more than one subsystem 124
 SRV
 option of DISPLAY TRACE command 55
 option of START TRACE command 132
 option of STOP TRACE command 145
 /SSR command 118
 /START command 119
 START DATABASE command 120, 122
 START DB2 command 123, 125
 START irlmproc command (MVS IRLM) 126–127
 START RLIMIT command 128
 START TRACE command 129, 136
 starting
 connections between IMS and a subsystem 119
 databases 120
 DB2 123
 IRLM component 126
 trace activity 129
 STARTRBA option of CHANGE LOG INVENTORY 173
 STAT
 option of DISPLAY TRACE command 55
 option of START TRACE command 131
 option of STOP TRACE command 145
 states
 of utility execution 167
 STATISTICS option of DSNH DISPLAY command 65
 status of utility
 monitoring 167
 STEPLIB
 used by utilities 156

STOGRROUP
 option of STOSPACE utility 295
 /STOP command 137
STOP DATABASE command 138, 139
STOP DB2 command 140
STOP irImproc command (MVS IRLM) 141
STOP RLIMIT command 142
STOP TRACE command 143, 146
STOPDB
 stopped state of utility execution 167
 stopping
 See terminating
STOSPACE utility 295, 296
 description 154
 effects of TERM on 148
 string
 delimiter in SQL 38
STRUCTURE
 option of DCLGEN subcommand 38
SUBMIT
 parameter on DSNU CLIST 91
SUBS option of /TRACE command 149
SUBSYS
 option of /CHANGE command 34
 option of /DISPLAY command 42
 option of /START command 119
 option of /STOP command 137
SUBSYS ALL option of /DISPLAY command 42
 subsystem recognition character (SRC)
 See SRC (subsystem recognition character)
SUFFIX option of DSNH CLIST 82
SUMMARY option of REPORT utility 287
SYMLIST
 option of DSNH CLIST 75
 option of DSNU CLIST 90
 synonyms for commands 13
 syntax
 conventions 7—11
 diagrams
 how to read 7—11
SYSCOPY
 option of COPY utility 186
 option of MERGECOPY utility 249
 used by COPY utility 156
SYSDISC
 option of LOAD utility 228
 used by LOAD utility 156
SYSERR
 used by LOAD utility 156
SYSERR option of LOAD utility 228
SYSIBM catalog tables
 See catalog tables
SYSIBM.SYSDBRM catalog table 19
SYSIBM.SYSPLAN catalog table 19
SYSIBM.SYSPLANDEP catalog table 19
SYSIBM.SYSSTMT catalog table 19
SYSIN
 used by utilities 156

SYSMAP
 used by LOAD utility 156
SYSMAP option of LOAD utility 228
SYSPRINT
 used by utilities 156
SYSREC
 option of LOAD utility 226
 option of REORG utility 267
 used by LOAD and REORG 156
SYSTEM
 option of DSN command 61
 option of DSNH CLIST 83
 option of DSNU CLIST 91
System Management Facility (SMF)
 See GTF (Generalized Trace Facility)
SYSTEM option of DSNUPROC 163
SYSTEMDB option of CHANGE LOG INVENTORY 172
SYSUT1
 option of LOAD utility 227
 option of REORG utility 267
 used by utilities 156

T

table
 catalog
 See catalog tables
 dropped 272
 option of DCLGEN subcommand 37
TABLE option of RUNSTATS utility 290
table space
 copying 185
 merging copies 247
 partitioned
 writing page numbers for REPAIR 283
 segmented
 in COPY utility 188
 in LOAD utility 239
 in REORG utility 269
TABLESPACE
 option of CHECK utility 180
 option of COPY utility 186
 option of MERGECOPY utility 248
 option of MODIFY utility 251
 option of QUIESCE 255
 option of RECOVER utility 259, 261
 option of REORG utility 267
 option of REPAIR utility 274
 option of REPORT utility 286
 option of RUNSTATS utility 289
TABLESPACESET option of REPORT 286
TBLSPACE option of CHANGE LOG INVENTORY 174
TERM option of DSNH CLIST 81, 82, 83
-TERM UTILITY command 147, 148, 169
 during REORG 270
 effect on RECOVER 263
 effect on rerunning LOAD 243
 effect on rerunning REORG 271

- TERM UTILITY command (*continued*)
 - effect on restarting COPY 188
- terminating
 - a utility 169
 - connections between IMS and a subsystem 137
 - databases 138
 - DB2 140
 - IRLM 141
 - state of utility execution 167
 - trace activity 143
 - utilities 147
- TEST option of DSN command 61
- THREAD
 - option of DISPLAY THREAD command 51
- TIME data type
- TIME option of DSNH CLIST 83
- TIMESTAMP data type
- TNO
 - option of DISPLAY TRACE command 56
 - option of STOP TRACE command 146
- TOCOPY option of RECOVER utility 261
- TORBA option of RECOVER utility 260
- TOSEQNO option of RECOVER utility 261
- TOVOLUME option of RECOVER utility 261
- trace
 - displaying 53
 - option of MODIFY irlmproc,START command 99
 - option of MODIFY irlmproc,STOP command 103
 - starting 129
 - stopping 143
- /TRACE command 149, 150
- tracing
 - events, tracing 149
- TRACK
 - option of DSNH CLIST 82
- TRANSACTION
 - option of DSNH DISPLAY command 65
 - option of DSNH MODIFY command 67
- TSO CLIST
 - DSNH 71–87
 - DSNU 88–94
 - options
 - ACQUIRE 73
 - ACTION 73
 - ADD 73
 - ALLOCATE 73
 - APOST 76
 - APOSTSQL 82
 - ASMLIB 73
 - ASMHLIB 73
 - ASMLOAD 73
 - ASMLIB 73
 - ASMLOAD 73
 - BATCH 81
 - BIND 74, 83
 - BLIB 74
 - BMEM 74
 - CICS 81
 - CICSOPT 74
 - CICSPRE 74

- TSO CLIST (*continued*)
 - options (*continued*)
 - CICSXLAT 75
 - CLIB 75
 - COB2CICS 75
 - COB2LIB 75
 - COB2LOAD 75
 - COMMA 76
 - COMMIT 81
 - COMPILE 75
 - CONLIST 75, 90
 - CONTROL 75, 90
 - COPTION 75
 - COPYDSN 90
 - CS 78
 - CURRENT 91
 - CYLINDER 82
 - DATE 76
 - DBRMLIB 76
 - DB2I 90
 - DEALLOCATE 81
 - DECIMAL 76
 - DELIMIT 76
 - DISCDSN 90
 - EDIT 91
 - ENTRY 76
 - EXPLAIN 77
 - FLAG 77
 - FORTLIB 77
 - FORTLOAD 77
 - HOST 77
 - IMS 81
 - IMSPRE 78
 - INDSN 90
 - INPUT 78
 - ISOLATION 78
 - LEAVE 80, 81, 82
 - LINECOUNT 78
 - LINK 78
 - LIST 75, 90
 - LLIB 78
 - LOAD 79
 - LOPTION 79
 - MACRO 79
 - ONE 79
 - OPTIONS 79
 - OUTNAME 79
 - PARMS 79
 - PASS 79
 - PCLOAD 79
 - PERIOD 76
 - PHASE 91
 - PLAN 79
 - PLIB 79
 - PLILIB 80
 - PLILOAD 80
 - PRECOMP 80
 - PRINT 80
 - PSECSPAC 80

TSO CLIST (continued)

options (continued)

PSPACE 80
QUOTE 76
QUOTESQL 82
RCTERM 80
RECDSN 91
RELEASE 81
REPLACE 73
RESTART 91
RETAIN 81
RR 78
RUN 81, 83
RUNIN 81
RUNOUT 82
SOURCE 82
SPACEUN 82
SQLDELIM 82
SUBMIT 91
SUFFIX 82
SYMLIST 75, 90
SYSTEM 83, 91
TERM 80, 81, 82, 83
TIME 83
TRACK 82
TSO 81
TWO 79
UID 92
UNIT 93
USE 73
UTILITY 90
VALIDATE 83
VOLUME 93
WORKUNIT 83
WSECSPAC 83
WSPACE 83
XLIB 83
XREF 83

TSO CLISTS

DSNU 160

TSO option of DSNH CLIST 81

TTNAME

tutorial panels 160

TWO option of DSNH CLIST 79

TYPE

option of DIAGNOSE utility 191

option of DISPLAY THREAD command 52

U

UID

option of DSNU CLIST 92

UID keyword in DSNUPROC 163

UNIT

option of CHANGE LOG INVENTORY 174

option of DSNU CLIST 93

unit of recovery

displaying an outstanding 42

unit of recovery (continued)

IMS/VS

outstanding 43

resetting by /CHANGE 34

indoubt

in conditional restart 175

resetting by /CHANGE 34

largest and smallest 257

UNLDDN option of REORG utility 267

UNLOAD

option of LOAD utility 227

option of REORG utility 268

USE

option of BIND subcommand 32

option of DISPLAY DATABASE command 47

option of DSNH CLIST 73

option of REBIND subcommand 109

UT option of START DATABASE command 122

UTILINIT utility phase 167

utilities

data set disposition for 157

data sets used 156

descriptions 153

invoking by DB2I 158

invoking by DSNU CLIST 160

invoking by supplied JCL 163

invoking by your own JCL 166

monitoring status 167

phases of execution 167

problems during execution 168

running 153-169

statements 160

states of execution 167

terminating 169

UTILITIES panel 158

utility

See also DB2 utility statement

change log inventory 170

CLISTS 71, 88

displaying status of 58

executing by DSNU CLIST 88

identifier 147

log print 254

name

RUNSTATS 288

names

CHECK 178

COPY 185

LOAD 224

MERGECOPY 247

MODIFY 251

RECOVER 257

REORG 266

REPAIR 273

STOSPACE 295

option descriptions 15

option of DISPLAY UTILITY command 58

option of DSNU CLIST 90

utility (*continued*)

- phases 167
- print log map 254
- statements
 - parsing 15
- states 167
- statistics 288
- terminating execution 147
- utility option
 - AGE, in MODIFY 252
 - ALLDUMPS, in DIAGNOSE 191
 - ARCHIVE, in CHANGE LOG INVENTORY 172
 - BACKOUT, in CHANGE LOG INVENTORY 175
 - CANCEL, in CHANGE LOG INVENTORY 174
 - CATALOG
 - in CHANGE LOG INVENTORY 174
 - in RECOVER 261
 - CHKPTRBA, in CHANGE LOG INVENTORY 174
 - CONTINUE, in RECOVER 260
 - CONTINUEIF, in LOAD 229
 - COPYDDN
 - in COPY 186
 - in MERGECOPY 249
 - COPY1, in CHANGE LOG INVENTORY 173
 - COPY1VOL, in CHANGE LOG INVENTORY 173
 - COPY2, in CHANGE LOG INVENTORY 173
 - COPY2VOL, in CHANGE LOG INVENTORY 173
 - CREATE, in CHANGE LOG INVENTORY 174
 - CRESTART, in CHANGE LOG INVENTORY 173
 - CSRONLY, in CHANGE LOG INVENTORY 175
 - DATA
 - in CHECK 180
 - in LOAD 226
 - in REPAIR 278, 279
 - DATE, in MODIFY 252
 - DEFAULTIF, in LOAD 238
 - DELETE
 - in CHANGE LOG INVENTORY 172
 - in MODIFY 252
 - DEVT
 - in COPY 186
 - in MERGECOPY 248
 - DISCARD, in LOAD 228
 - DISCARD, in LOAD 228
 - DSNAME, in CHANGE LOG INVENTORY 173
 - DSNUM
 - in COPY 186
 - in MERGECOPY 248
 - in MODIFY 252
 - in RECOVER 259
 - in REPAIR 275
 - in REPORT 286
 - DUMP, in REPAIR 279
 - END, in DIAGNOSE 191
 - ENDRBA, in CHANGE LOG INVENTORY 173
 - ENFORCE, in LOAD 228
 - ERRDDN, in LOAD 228
 - ERROR RANGE, in RECOVER 260

utility option (*continued*)

- FORMAT, in LOAD 227
- FORWARD, in CHANGE LOG INVENTORY 175
- FULL, in COPY 187
- INDDN, in LOAD 226
- INDEX
 - in CHECK 179
 - in RECOVER 261
 - in REORG 267
 - in REPAIR 275, 277
 - in RUNSTATS 290
- INDEXES, in REPAIR 275
- INTO TABLE, in LOAD 230
- KEY, in REPAIR 276
- LENGTH, in REPAIR 280
- LOCATE INDEX, in REPAIR 277
- LOCATE TABLESPACE, in REPAIR 276
- LOG
 - in LOAD 226
 - in REORG 267
 - in REPAIR 274
- MAPDDN, in LOAD 228
- NAME, in CHECK 180
- NEWCAT, in CHANGE LOG INVENTORY 173
- NEWCOPY, in MERGECOPY 249
- NEWLOG, in CHANGE LOG INVENTORY 172
- NOCHECKPEND, in REPAIR 275
- NOCOPYPEND, in REPAIR 275
- NOPASSWD, in CHANGE LOG INVENTORY 175
- NORCVRPEND, in REPAIR 275
- NULLIF, in LOAD 238
- OBJECT, in REPAIR 274
- OFFSET, in REPAIR
 - DUMP statement 280
 - REPLACE statement 278
 - VERIFY statement 278
- ONLY, in REORG 268
- PAGE
 - in RECOVER 260
 - in REPAIR 276, 277
- PAGES, in REPAIR 280
- PART
 - in LOAD 232
 - in REORG 267
- PASSWORD, in CHANGE LOG INVENTORY 174
- POSITION, in LOAD 234
- RECOVERY, in MODIFY 251
- REPLACE
 - in LOAD 226
 - in REPAIR 278
- RESET, in REPAIR 278
- RESUME, in LOAD 226
- RID, in REPAIR 276
- SCOPE, in CHECK 180
- SET INDEX, in REPAIR 274
- SET TABLESPACE, in REPAIR 274
- SHRLEVEL
 - in COPY 187
 - in RUNSTATS 290

utility option (*continued*)

SORTDEVT
in CHECK 180
in LOAD 229
in RECOVER 262
in REORG 268

SORTNUM
in CHECK 180
in LOAD 229
in RECOVER 263
in REORG 268

STARTRBA, in CHANGE LOG INVENTORY 173

STOGROUP, in STOSPACE 295

SYSTEMDB, in CHANGE LOG INVENTORY 172

TABLE, in RUNSTATS 290

TABLESPACE
in CHECK 180
in COPY 186
in MERGECOPY 248
in MODIFY 251
in QUIESCE 255
in RECOVER 259, 261
in REORG 267
in REPAIR 274
in REPORT 286
in RUNSTATS 289

TABLESPACESET, in REPORT 286

TBLSPACE, in CHANGE LOG INVENTORY 174

TOCOPY, in RECOVER 261

TORBA, in RECOVER 260

TOSEQNO, in RECOVER 261

TOVOLUME, in RECOVER 261

TYPE, in DIAGNOSE 191

UNIT, in CHANGE LOG INVENTORY 174

UNLDDN, in REORG 267

UNLOAD, in REORG 268

VERIFY, in REPAIR 277

VSAMCAT, in CHANGE LOG INVENTORY 175

WHEN, in LOAD 232

WORKDDN
in CHECK 180, 181
in LOAD 227
in MERGECOPY 248
in RECOVER 262
in REORG 267

UTILTERM utility phase 167

UTPRINT
used by utilities 156

UTPROC option of DSNUPROC 164

V

VALIDATE
option of BIND subcommand 31
option of DSNH CLIST 83
option of REBIND subcommand 109

validation routine
used by LOAD utility 224
used by REORG utility 268

VALUE

VARCHAR
option of LOAD utility 238

VARCHAR data type

VARGRAPHIC
option of LOAD utility 238

VERIFY option of REPAIR utility

VOLUME option of DSNH CLIST 93

VSAM
catalog
in COPY utility 186
in MERGECOPY utility 248, 259
in RECOVER utility 252
in REORG utility 269
in STOSPACE utility 296

data set 172

password
in CHANGE LOG INVENTORY utility 170
in DCLGEN 37
in PRINT LOG MAP utility 254

VSAMCAT option of CHANGE LOG INVENTORY 175

W

WHEN
option of LOAD utility 232
with LOAD 226, 243

work data set
for LOAD 240

WORKDDN
option of CHECK utility 180, 181
option of LOAD utility 227
option of MERGECOPY utility 248
option of RECOVER utility 262
option of REORG utility 267

WORKUNIT option of DSNH CLIST 83

WSECSPAC option of DSNH CLIST 83

WSPACE option of DSNH CLIST 83

X

XLIB option of DSNH CLIST 83

XREF option of DSNH CLIST 83

SC26-4378-0

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you have applied any technical newsletters (TNLs) to this book, please list them here: _____

Comments (please include specific chapter and page references) :

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

If you want a reply, please complete the following information:

Name _____ Date _____

Company _____ Phone No. (_____) _____

Address _____

Thank you for your cooperation. No postage is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail them directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

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, NY

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Programming Publishing
P.O. Box 49023
San Jose, CA 95161-9023



Fold and tape

Please do not staple

Fold and tape



SC26-4378-0

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you have applied any technical newsletters (TNLs) to this book, please list them here: _____

Comments (please include specific chapter and page references) :

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

If you want a reply, please complete the following information:

Name _____ Date _____

Company _____ Phone No. (_____) _____

Address _____

Thank you for your cooperation. No postage is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail them directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and tape

Please do not staple

Fold and tape

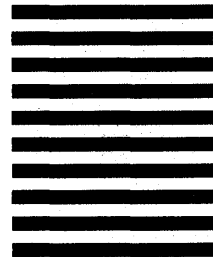


NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Programming Publishing
P.O. Box 49023
San Jose, CA 95161-9023



Fold and tape

Please do not staple

Fold and tape



SC26-4378-0

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you have applied any technical newsletters (TNLs) to this book, please list them here: _____

Comments (please include specific chapter and page references) :

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

If you want a reply, please complete the following information:

Name _____ Date _____

Company _____ Phone No. (_____) _____

Address _____

Thank you for your cooperation. No postage is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail them directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and tape

Please do not staple

Fold and tape

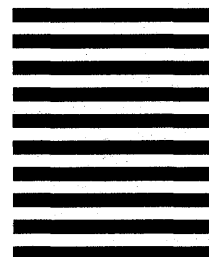


NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Programming Publishing
P.O. Box 49023
San Jose, CA 95161-9023



Fold and tape

Please do not staple

Fold and tape



SC26-4378-0

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you have applied any technical newsletters (TNLs) to this book, please list them here: _____

Comments (please include specific chapter and page references) :

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

If you want a reply, please complete the following information:

Name _____ Date _____

Company _____ Phone No. (_____) _____

Address _____

Thank you for your cooperation. No postage is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail them directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and tape

Please do not staple

Fold and tape



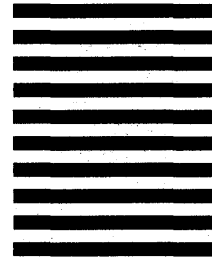
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Programming Publishing
P.O. Box 49023
San Jose, CA 95161-9023



Fold and tape

Please do not staple

Fold and tape

