HP SYSTEM DICTIONARY/XL SDMAIN

HP 3000 MPE/iX Computer Systems

Edition 1

HEWLETT®
PACKARD

Manufacturing Part Number: 32256-90001
E1287

U.S.A. November 1987

Notice

Restricted Rights Legend

Acknowledgments

UNIX is a registered trademark of The Open Group.

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304 U.S.A.

Preface

Manual Organization

This manual describes the SDMAIN program for the HP System Dictionary/XL software system that operates on the 900 Series HP 3000 computer family. It is a reference document for all persons involved in creating and maintaining a dictionary database on the 900 Series HP 3000. As such, it assumes a working knowledge of the 900 Series HP 3000 computer system. If the subject of data dictionaries is new to you, it might be helpful to read the Hewlett Packard primer entitled *Managing Your Information Network: A Data Dictionary Primer* that explains fundamental concepts of data dictionaries.

In addition to the SDMAIN program, HP System Dictionary/XL provides intrinsics for users who wish to access HP System Dictionary/XL programmatically. A set of four HP System Dictionary/XL utilities is also available for performing dictionary tasks such as creating IMAGE databases from definitions residing in HP System Dictionary/XL. A COBOL Definition Extractor utility is available to generate COBOL II source code from definitions residing in HP System Dictionary/XL. The manual is organized as follows:

Chapter 1	Introducing the System Dictionary/XL SDMAIN Program Presents an overview of the HP System Dictionary/XL SDMAIN program.
Chapter 2	Running the SDMAIN Program Describes how to execute the HP System Dictionary/XL SDMAIN program.
Chapter 3	The System Dictionary Command Language Provides a conceptual overview of the kinds of functions and tasks that the Command Language performs. This section also provides a listing of the specific commands that belong in each category of commands.
Chapter 4	System Dictionary Commands Presents the syntax and parameter descriptions of the Command Language. This chapter also provides a short example of each command.
Appendix A	SDMAIN Error Messages Provides a listing of the HP System Dictionary/XL error messages that are returned on a System Dictionary SDMAIN error, one or more probable causes of each error, and an action for each listed cause.
Appendix B	SDMAIN Command Abbreviations Provides a table listing the HP System Dictionary/XL SDMAIN commands and their abbreviation.
Glossary	Provides a glossary of the major terms associated with HP System Dictionary.

Audience

The intended users of this manual are those individuals that wish to access the SDMAIN Command Language to create, maintain, and report entries in the dictionary. Knowledge of data dictionary concepts is assumed. Knowledge of the HP 3000 MPE XL operating system is also helpful.

Resources

In addition to this manual, you may need to consult the following manuals:

Managing Your Information Network: A Data Dictionary Primer

HP System Dictionary/XL Utilities Reference Manual

HP System Dictionary/XL General Reference Manual, Volume 1

HP System Dictionary/XL General Reference Manual, Volume 2

HP System Dictionary/XL Intrinsics Reference Manual

HP System Dictionary/XL COBOL Definition Extractor Reference Manual

TurboIMAGE/XL Reference Manual

HP IMAGE Reference Manual

SQL Reference Manual

HP SQL Database Administration Guide

VPLUS Reference Manual

QUERY/V Reference Manual

KSAM Reference Manual

HP 3000 General Information Manual

MPE XL Commands Reference Manual

MPE XL Intrinsics Reference Manual

Program Design and Optimization

Programmer's Utilities and Tools

Native Language Support Reference Manual

EDIT/V Reference Manual

Pascal/XL Reference Manual

Pascal/XL Programmer's Guide

COBOL II/XL Reference Manual

COBOL II/XL Programmer's Guide

HP FORTRAN 77 Reference Manual

HP FORTRAN 77/XL Reference Manual Supplement

HP FORTRAN 77/XL Programmer's Guide

HP FORTRAN 77/XL Programmer's Guide Supplement

SPL Reference Manual

Future Upgrades

In the future, Hewlett-Packard will prefix all names of objects it adds to the core set with "HP" . To avoid potential name conflicts, do not create any entity types, relationship classes, attributes, scopes, domains, or versions prefixed with "HP" .

Conventions

NOTATION	DESCRIPTION
nonitalics	Words in syntax statements which are not in italics must be entered exactly as shown. Punctuation characters other than brackets, braces and ellipses must also be entered exactly as shown. For example:

EXIT;

italics Words in syntax statements which are in italics denote a parameter which must be replaced by a user-supplied variable. For example: CLOSE *filename*

[ ] An element inside brackets in a syntax statement is optional. Several elements stacked inside brackets means the user may select any one or none of these elements. For example:

[A]

[B] User *may* select A or B or neither.

{ } When several elements are stacked within braces in a syntax statement, the user must select one of those elements. For example:

{A}

{B} User *must* select A or B or C.

{C}

... A horizontal ellipsis in a syntax statement indicates that a previous element may be repeated. For example:

[,*itemname*]...;

In addition, vertical and horizontal ellipses may be used in examples to indicate that portions of the example have been omitted.

, A shaded delimiter preceding a parameter in a syntax statement indicates that the delimiter *must* be supplied whenever (a) that parameter is included or (b) that parameter is omitted and any *other* parameter which follows is included. For example:

itema [, *itemb*][, *itemc*]

means that the following are allowed:

itema

itema,itemb

itema,itemb,itemc

itema,,itemc

Å When necessary for clarity, the symbol Å may be used in a syntax statement to indicate a required blank or an exact number of blanks. For example:

SET[(*modifier*)] Å (*variable*);

underlining Brackets, braces or ellipses appearing in syntax or format statements which must be entered as shown will be underlined. For example:

LET *var* [[*subscript*]] = *value*

Output and input/output parameters are underlined. A notation in the description of each parameter distinguishes input/output from output parameters. For example:

CREATE (*parm1*, parm2, flags, error)

shading Shading represents inverse video on the terminal's screen. In addition, it is used to emphasize key portions of an example.

[[   ]]   The symbol [[   ]] may be used to indicate a key on the terminal's keyboard. For example, [[RETURN]] indicates the carriage return key.

[[CONTROL]]*char*   Control characters are indicated by [[CONTROL]] followed by the character. For example, [[CONTROL]]Y means the user presses the control key and the character Y simultaneously.

1 Introducing the System Dictionary SDMAIN Program

Product Overview

HP System Dictionary SDMAIN is a program that allows you to access the System Dictionary product. You can:

- Enter and maintain data in HP System Dictionary
- Report on and make queries about the data residing in HP System Dictionary

This access is achieved by means of a **command language**. This manual describes the HP System Dictionary Command Language--both its syntax and usage. Examples are given showing how the commands are used within common types of applications.

If you are a new user of HP System Dictionary, you will need an understanding of what a data dictionary is along with its purpose and function.

What is a Data Dictionary?

Just as an ordinary dictionary is a collection of definitions of words, a data dictionary is a collection of definitions and descriptions of data that resides on a computer system. In a dictionary, the smallest unit of information is a word, whereas, in a data dictionary, the smallest unit of information is a **data element**.

The data dictionary does not contain the data itself, but contains **metadata** --data about data. This metadata can be descriptions and definitions of various kinds. It can describe such things as:

Data:	Names and definitions of data elements
Data Relationships:	How data is related to other data
Data Responsibility:	Who is responsible for what data
Organizational Structure:	The information flow, who uses the data
Location Information:	Where files, programs, and reports reside
Security Information:	Who has access to what data

A typical example of a piece of metadata is a data element called "SSN" which represents a piece of data--a social security number. The social security number *itself* does not reside in the dictionary, but a description of that piece of data does. For instance, the data dictionary might tell you the name of the data element, the storage length, the display or output length, the type of data (numeric or character), its sign if the element is numeric, what database or program that data resides in, and possibly what departments in the organization use and maintain that data.

The data dictionary, therefore, serves many purposes. You can use it as a quick directory to the information that resides on a computer system--where to go to get pieces of data. You can also, however, use it as one of the primary means for ensuring consistency of data definitions and preventing data redundancy. This means that programmers and developers may be required to check the data dictionary for data elements that already exist on their system before they design a new program. Therefore, if a data element already exists on the system describing a social security number (for example, "SSN"), the dictionary reports this information and does not allow the programmer to add a new element with the same name. This helps an organization to save time in program development by using data definitions that already exist. It also saves data storage space by preventing data redundancy and helps to standardize data definitions within an organization.

For a more detailed introduction to what data dictionaries are and how an organization can use a data dictionary, see the HP primer entitled *Managing Your Information Network: A Data Dictionary Primer*

What is System Dictionary?

System Dictionary provides dictionary capabilities to the 900 Series, HP 3000 computer systems. All of the dictionary functions that are discussed above, plus others that are discussed later in this manual are provided by System Dictionary.

The main areas of functionality in System Dictionary are:

1 Defining the Dictionary Environment

2 Defining the Dictionary Structure

3 Defining and Maintaining Data

4 Reporting on Dictionary Contents

You can access all of these functions through the HP System Dictionary Command Language. Chapter 3 provides an overview of each of these functional areas. Chapter 4 provides details about each command.

In addition to SDMAIN that is described in this document, HP System Dictionary provides the following utilities:

- **Dictionary/V to HP System Dictionary Conversion Utility (SDCONV)** - translates data definitions from a Dictionary/V database to the System Dictionary database.
- **TurboIMAGE/V and IMAGE/V Database Definition Loader (SDDBD)** - loads information about an IMAGE/V database structure into System Dictionary from an IMAGE/V root file.
- **VPlus/V Forms Definition Loader (SDVPD)** - loads information about VPlus/V forms into System Dictionary from a VPlus/V forms file.
- **TurboIMAGE/V and IMAGE/V Database Creation Utility (SDDBC)** - creates IMAGE schemes and root files from definitions in System Dictionary.
- **COBOL Definition Extractor Utility (SDCDE)** - creates COBOL copylibs from definitions in System Dictionary.

These utilities are described in the documents entitled *HP System Dictionary/XL Utilities Reference Manual, HP System Dictionary/XL General Reference Manual, Volume 2.* The basic System Dictionary concepts are described in the manual entitled *HP System Dictionary/XL General Reference Manual, Volume 1.*

What is the SDMAIN Program?

The program that provides the interface to System Dictionary is the Dictionary Maintenance Program called SDMAIN. You must run the SDMAIN program to access the Command Language commands that create, maintain, and report entries in the dictionary. The SDMAIN program provides access to the dictionary without requiring a program to be written for each specific task.

How to use this Manual

This manual is designed to progress from a general description of HP System Dictionary and its various functional areas to the specific description of the HP System Dictionary Command Language.

- Chapter 2 explains how to execute the SDMAIN program.

- Chapter 3 provides a conceptual overview of the kinds of functions and tasks that can be performed by the Command Language, with a listing of the specific commands that belong in each category.
- Chapter 4 presents the syntax and parameter descriptions of the Command Language, with a short example of each command.
- Appendix A lists all of the SDMAIN error messages.
- Appendix B provides a summary of the abbreviations for the command words in the system.

2 Running the SDMAIN Program

Overview

System Dictionary provides a Dictionary Maintenance program called SDMAIN to create, maintain, and report entries in the dictionary. This chapter explains how to run SDMAIN, define files and file equations used in conjunction with the program, and how to define SDMAIN's run options.

To use System Dictionary and run the SDMAIN program, System Dictionary must already be installed on your system and a dictionary initialized. Initialization procedures are described in Part 1 of the *HP System Dictionary/XL General Reference Manual Volume 2*. If HP System Dictionary is *not* already installed on your system, see your System Manager.

Defining Input, Log, Macro, and Output Files

The first step in using the dictionary is to define the files that you use.

SDMAIN uses the following formal file designators:

- SDIN = the input file
- SDLOG = the log file
- SDMACR= the macro file
- SDOUT = the output file

SDMAIN opens all the files with file equates allowed. You can, therefore, redirect the SDMAIN files to your own files, if you wish, by means of MPE file equations.

The Input File (SDIN)

SDMAIN accepts input from the file SDIN. The default for SDIN is $STDINX, or your terminal (for sessions). This means that input is accepted from the terminal unless otherwise specified. You can, however, redirect the input to another file. This causes SDMAIN to accept the input from the specified file. The file must be an unnumbered file with a record length of 80 bytes or less (if over 80 bytes, SDMAIN reads only the first 80 and ignores the remainder). Note that if the specified input file does not include an EXIT command, SDMAIN is redirected to accept input from $STDINX when the end of file is reached.

Redirecting the Input File. If you would like input to come from a file other than SDIN (or $STDINX, your terminal), you can redirect the input to a file of your own choosing. In order to redirect SDIN, enter the following MPE file equation prior to the RUN command:

```
      :FILE SDIN = command file name
```

After entering this command, the next time you run SDMAIN, input is accepted from *command file name*. See the examples following "Redirecting the Log File" for an example.

The Log File (SDLOG)

When you execute SDMAIN, all valid commands are added, or logged, to a temporary log file called SDLOG. Valid commands are those that do not contain any syntax errors.

Examples of types of commands that are not logged are:

- Commands containing syntax errors
- Subsystem commands, such as Edit/V responses you enter through the use of the EDIT command
- The REDO command and its editing subcommands (The resulting edited dictionary commands, however, are logged as they are executed.)
- Certain commands you enter within a START/SAVE pair

  The START/SAVE command pair is a System Dictionary pair of reporting commands. Only *executable commands* within this pair are logged. Therefore, no COMMENT, HELP or SHOW commands that can occur within this pair are logged. As with the REDO command, once you issue the SAVE command, the START/SAVE pair is executed and all executable commands within that pair are logged to SDLOG.
- If you enter a [[CONTROL]] Y before you enter the full command, the entire command is not logged.
- Macro calls and INCLUDE commands are not logged. However, the expanded macro and the commands in the include file are logged as they are executed.

Redirecting the Log File. To redirect SDLOG to another file, enter the following MPE command:

```
    :FILE SDLOG = log file name
```

When SDLOG is redirected to another file, the specified file saves all input entered for that session. If the file does not exist, then SDMAIN builds it as a temporary fixed-length ASCII file with a record size of 80 bytes. If the file already exists, then SDMAIN asks for permission to overwrite it with the following prompt:

```
filename.groupname.accountname  already exists.  Is it okay to overwrite (N/Y)?
```

If you respond NO to the above prompt, logging is disabled and no commands are logged.

If SDLOG is not redirected (or if it is redirected to a temporary file), it is only a temporary file and must be saved if you want SDLOG to become a permanent file. To do this, issue the MPE SAVE command and then rename the resulting file:

```
    :SAVE SDLOG
    :RENAME SDLOG, LOGFILE
```

If you fail to rename the saved file to something other than SDLOG, and do not redirect SDLOG, SDMAIN asks you for permission to overwrite it during the next run. If you answer NO, logging is disabled and no commands are written to the log file. You can use the OPTIONS command to change the log during a run.

You can disable logging in several ways, The first is to redirect SDLOG to the file $NULL. Next, as mentioned above, a response of NO to the overwrite prompt disables logging. Finally, the OPTIONS command can turn logging ON and OFF during a run.

The ability to redirect the log file is a valuable feature. This capability, in conjunction with the ability to redirect the input file, allows you to create and save a file of commands that you can reuse at a later date. This becomes especially helpful if, for instance, a certain sequence of commands is executed on a regular basis. If, for example, once a month, all data elements within a particular data set are updated, the sequence of commands necessary to accomplish this update can be kept in a command file and simply re-executed whenever needed.

Examples

There are several ways to combine SDLOG and SDIN. In both of the following cases, the input file SDIN is redirected to take input from a command file. In the first example, the log file is redirected to another file called cmdfile, which is then saved and used as input for the redirected input file:

:FILE SDLOG=cmdfile	Redirects SDLOG to cmdfile.
:RESET SDIN	Cancels any existing file equations for SDIN. Do this if SDIN is already directed to a file.

:RUN SDMAIN.PUB.SYS	Runs SDMAIN and enters the commands. The commands that are now entered, go to the file cmdfile.
:	
:SAVE cmdfile	Saves cmdfile as a permanent file.
:FILE SDIN=cmdfile	SDIN now takes commands from this file.
:RESET SDLOG	Cancels the previous file equation for SDLOG.
:RUN SDMAIN.PUB.SYS	Commands are now taken from cmdfile.

Notice that the MPE RESET command is used here to cancel any existing file equations for SDIN and SDLOG.

The following example saves and renames the SDLOG file. The redirected SDIN file then uses the renamed file.

:RUN SDMAIN.PUB.SYS	Commands are written to SDLOG.
:	
:SAVE SDLOG	Saves SDLOG file.
:RENAME SDLOG,cmdfile	Renames SDLOG file to cmdfile.
:FILE SDIN=cmdfile	SDIN takes commands from this file.
:RUN SDMAIN.PUB.SYS	Commands are now taken from cmdfile.

Both of the above methods give you the same result.

The Macro File (SDMACRO)

SDMAIN allows you to define macros (see the discussion of macros later in this Chapter). The definitions are placed in a file that the system loads when SDMAIN is run. The default is for there to be no macro file (if no file named SDMACRO exists) and so no macros available when the system is run. The file must be an unnumbered file with a record length of 80 bytes or less (if over 80 bytes, only the first 80 are read and the remainder are ignored).

Redirecting the Macro File. To redirect SDMACRO to a file, use the following MPE file equation:

```
:FILE SDMACRO = macro file name
```

Now, the next time SDMAIN is run, the macros that have been defined in *macro file name* are loaded for use. If no file equate is provided and there is no file named SDMACRO, no macros are available for the next run. You can use the OPTIONS command to change the macro file during a run.

The Output File (SDOUT)

SDMAIN's reporting information is written to the output file called SDOUT. The default for this file is $STDLIST, which for sessions is your terminal. Note that only output from reporting commands is sent to the file. All other output (that is, HELP, SHOW, and error messages) is still sent to $STDLIST.

Redirecting the Output File. To redirect SDOUT to a file, use the following MPE file equation:

```
:FILE SDOUT = output file name
```

If the file does not exist, then SDMAIN builds it as a permanent ASCII file with a record size of 132 bytes. If the file already exists, then SDMAIN asks for permission to overwrite it:

```
*filename.groupname.accountname*  already exists.  Is it okay to overwrite (N/Y)?
```

If you respond NO to the above prompt, output is directed to $STDLIST. You can use the CONFIGURE command to change the output file during a run.

SDMAIN's Temporary Edit File

When you run SDMAIN, it creates an edit file named EDTXT*xxx* (where *xxx* is a number). EDTXT*xxx* is created as a permanent file, but deleted when you exit the system (as long as you do not abort the program). Therefore, if you issue an MPE LISTF command while running SDMAIN, this file is listed. If, however, you already have a file with the same name and number as the EDTXT*xxx* file, SDMAIN increments the EDTXT number and uses that number.

For example, if you already have a file named EDTXT0, then when the edit file is created, SDMAIN finds EDTXT0, increments the number to 1, and creates an edit file with the name EDTXT1. There is, therefore, no conflict between the two files, and your file, EDTXT0, is not deleted.

SDMAIN's Run Options

When SDMAIN is run, you can specify one of several run options through use of the PARM parameter. This parameter allows you to specify how errors that you may encounter are dealt with. The basic syntax of the RUN command with the PARM parameter is:

```
     :RUN SDMAIN.PUB.SYS; PARM = n
```

The options available for the PARM value are:

-1	Parse only. Only command syntax is checked. The dictionary remains unchanged.
0	Execute and do not terminate regardless of the number of errors. This is the default option.
1	Execute and terminate on the first error.
>1	Execute and terminate if the specified number of errors are encountered.

Parse-Only Option

To run SDMAIN with the Parse-Only option, enter the following command:

```
     :RUN SDMAIN.PUB.SYS;PARM = -1
```

All commands are checked for correct syntax, but the contents, structure, etc., of the dictionary remains unchanged. This option can prove helpful when instructing employees how to use System Dictionary. In this way, all commands can be entered without the dictionary contents being changed.

Perhaps the most important use of the Parse-Only option, however, occurs when running System Dictionary in batch mode. This allows you to run the job and catch syntax errors before actually altering dictionary contents or structure. Once the commands have been parsed, or checked for syntax errors, you can then resubmit the job and execute the commands.

Whenever SDMAIN is run in batch mode, without the Parse-Only option, the contents of the dictionary are altered for all valid commands, but not for any invalid commands. Therefore, the dictionary can be left in an incomplete state since all valid commands were executed, while all invalid commands were not executed. Also, for run options of 1 or greater, no commands encountered after the given number of errors specified by the run option are executed. This becomes tedious, because once this occurs and the error is corrected, the valid commands that were executed prior to the error need to be deleted. Otherwise, these commands are executed for the second time once the batch job is resubmitted.

Execute and Terminate Options

The Execute and Terminate options (0, 1, or greater) allow you to specify the number of errors that can occur before the SDMAIN program terminates. The default option is 0--execute and do not terminate, regardless of the number of errors. Option 1 causes the system to terminate the program when the first error is encountered. When any number greater than 1 is entered, the program terminates once this number of errors is encountered.

Job Control Word (JCW)

The Job Control Word called JCW is a system parameter that indicates the termination condition of the program. JCW is set by the system according to the condition under which the SDMAIN program is terminated. The two possible settings for JCW are:

0	Program terminated normally with an EXIT command.
32768	Program terminated abnormally before an EXIT command.

The Job Control Word allows a batch mode user to test for the termination condition and act accordingly.

The JCW is set to 32768 (FATAL) whenever the system must terminate before executing an EXIT command. This can occur as a result of several conditions:

- The number of errors encountered equaled or exceeded the number specified by the PARM value.
- A fatal MPE file system error was encountered.
- An SDMAIN system bug was discovered.

Running the SDMAIN Program

You can run the SDMAIN program in either session mode or batch mode. After logging on and defining the files that you will be using, you are ready to run the System Dictionary Maintenance program in session mode.

Session Mode

To run SDMAIN in session mode, enter the following MPE command:

```
      RUN SDMAIN.PUB.SYS
```

Now press [[RETURN]]. SDMAIN responds with the banner, followed by a prompt character:

HP System Dictionary SDMAIN HP32256v.uu.ff - (C)Hewlett-Packard Co. 1985

SDMAIN Prompt Characters. SDMAIN uses both a command (single) prompt and a continuation (double) prompt:

>	Command prompt. SDMAIN expects a command.
>>	Command continuation prompt. SDMAIN expects a subcommand or a keyword clause to complete the command.

Once the command prompt (>) appears, SDMAIN is ready to receive commands. The only SDMAIN commands that are allowed at this point are:

COMMAND	Makes a comment

CONFIGURE	Sets the output configuration
DEFINE	Defines the SDMAIN environment
EXIT	Terminates the program
FORMAT	Specifies a title and header for a report
HELP	Gets a list of valid commands
INCLUDE	Redirects input from another file
OPTIONS	Sets prompting, logging, and macro options
REDO	Redoes the previous command
SHOW	Shows the SDMAIN environment
SHOWMACRO	Shows a list of the macros currently defined
SHOWREDO	Displays a list of commands currently on the redo history stack

If you enter any other command, SDMAIN issues an error message.

Batch Mode

You can also execute SDMAIN in batch mode. If you have not already created a job file, you can create one by using the MPE text editor system. To begin creating your job file, enter the following command:

```
EDITOR
```

You may then begin entering your job file. When executing in batch mode, the EDIT command, REDO command, and attribute prompting are not allowed. One example of a job file is shown below. In this example, both the input file and the output file are redirected:

!JOB SDJOB,MARSHA.ACCT	Gives job name, user name and account name
!FILE SDIN=COMMANDS	Redirects SDIN input file
FILE SDOUT=OUTPUT	Redirects SDOUT output file
!CONTINUE	Overrides any job or session errors. Allows continuation of the job if the following command produces a fatal error.
!RUN SDMAIN.PUB.SYS;PARM=5	Issues SDMAIN RUN command
!IF JCW=FATAL THEN	Tests for FATAL JCW condition
!TELL MARSHA.ACCT Fatal Dictionary Error	Notifies user with TELL command if FATAL
!ELSE	Otherwise...
!TELL MARSHA.ACCT Dictionary Job Completed	Tells the user the dictionary job is completed
!ENDIF	Ends IF statement

!EOJ	End of job

Notice that the RUN command specifies a PARM parameter of 5. Therefore, if 5 or more errors are encountered, the program terminates. The JCW condition is checked and you are then notified that a fatal dictionary error occurred. You must use the CONTINUE command. If you do not include it, the job stream terminates on the FATAL error and you are not informed of the errors that occurred.

You now need to correct the errors and delete the previously executed valid commands in the input file before you resubmit the job.

Another way to proceed would be to use the Parse-Only option, catch all the syntax errors, and then resubmit the job. You can still test for the termination condition and be notified by the TELL command of either a FATAL error or a completed job.

For more information on creating jobs, see the *MPE XL Commands Reference Manual*.

Terminating SDMAIN

To terminate the SDMAIN program, enter:

```
    >EXIT.
```

The EXIT command closes the dictionary and then terminates the program, returning you to MPE. If any structure changes are made while running SDMAIN in customization mode, the dictionary is restructured before the program is terminated. This could take some time if a large number of dictionary occurrences are affected.

3 The System Dictionary Command Language

Overview

The HP System Dictionary Command Language enables you to create, maintain, and report on entries in System Dictionary through the SDMAIN program which is discussed in the previous chapter.

This chapter discusses characteristics of the command language itself, plus describes the four general categories of commands. A description of each of these four categories is followed by a list of the commands that you can use in that category. The four categories of commands are:

- Dictionary environment commands
- Dictionary structure commands
- Dictionary data maintenance commands
- Dictionary reporting commands

A complete description of each command is located in Chapter 4 of this manual.

The HP System Dictionary Command Language

The HP System Dictionary Command Language is a free formatted language that allows you to create, maintain, and report on entries in HP System Dictionary. The general format of the language is:

```
COMMAND  SUBCOMMAND  OBJECT-CLAUSE;

KEYWORD-CLAUSE1; KEYWORD-CLAUSE2; ... KEYWORD-CLAUSEn.
```

Free formatting means that the command, subcommand, object-clause, and keyword-clause may appear on the same line or on different lines. However, the order of these language elements is important. They are, therefore, positional. In other words, a subcommand must follow a command, an object-clause must follow a subcommand, and a keyword-clause must follow an object-clause. However, among themselves, keyword clauses are non-positional. That is, you may enter them in any order, as long as they follow the object-clause. For example, you may enter **keyword-clause1** after **keyword-clause2**. You cannot, however, enter them before the object-clause.

Table 3-1 shows the System Dictionary command language elements.

Table 1: System Dictionary Language Elements

Language Element	Description
COMMAND	The SDMAIN-defined name that specifies the action to be taken EXAMPLE: CREATE ENTITY ship-date;ENTITY-TYPE=element.
SUBCOMMAND	The SDMAIN-defined name that specifies the general target of the action. EXAMPLE: CREATE ENTITY ship-date;ENTITY-TYPE=element.

Table 1: System Dictionary Language Elements

Language Element	Description
OBJECT-CLAUSE	The user-defined name of the object. This is the specific target of the action specified by the command. EXAMPLE: CREATE ENTITY ship-date;ENTITY-TYPE=element.
KEYWORD-CLAUSE	A keyword clause can be either a single keyword or a keyword followed by an equal sign (=) that is followed by either nothing, a single value, or a list of values separated by commas. The keywords are SDMAIN-defined, while their values are either SDMAIN-defined or user-defined.* Keyword clauses are separated by semicolons. EXAMPLE: CREATE ENTITY ship-date;ENTITY-TYPE=element.

* There are two exceptions to this punctuation rule:

ATTRIBUTE-LIST and SUB-REPORT

Both of these keywords must be followed by an equal sign and the keyword value list must be enclosed in parentheses to avoid conflicts. An example of this is:

REPORT ENTITY item-name;ENTITY-TYPE=element;

ATTRIBUTE-LIST=(max-record-size=256,min-record-size=128).

Table 3-2 lists the System Dictionary Command Language punctuation characters.

Table 2: Command Language Punctuation Characters

CHARACTER	DESCRIPTION
.	Terminates commands. The period signifies the end of a string of characters denoting a command. It is required for all commands except the COMMENT, EXIT, HELP, REDO, RESTRUCTURE, SHOW, SHOWMACRO and START commands, where it is optional.
,	Separates items in a list. The comma separates objects in an object list and values in a keyword value list.
;	Separates clauses. The semicolon separates object clauses from keyword clauses and keyword clauses from each other.
=	Specifies a keyword and a keyword value list, or an attribute and an attribute list, i.e., ENTITY-TYPE=element; ATTRIBUTE-LIST=(element-type=9,byte-length=8).
< >	Used with the ATTRIBUTE-LIST or SUB-REPORT keyword clauses to specify the beginning and end of a keyword value list and within any value clause allowing selection criteria to denote precedence among Boolean operators.

Table 2: Command Language Punctuation Characters

CHARACTER	DESCRIPTION
"	Delimits character string values. Use two double quotes ("") to represent a single double quote (") within a string. Required around variable length attribute values, the description attribute of stored reports, and the TITLE and PAGE-HEADER parameters of the FORMAT command. Optional around passwords, character attribute values, and alias attribute values, as long as the value is a valid dictionary name. If the string contains any invalid characters (See "User Defined Names" later in this Chapter) or has more than 32 characters, the value must be enclosed in quotes.
ÅPP	Specifies a blank. You must use blanks to separate a command and subcommand, a command and object clause, and a subcommand and object clause. Any number of blanks may appear between any name and any punctuation character.
[[Return]]	Specifies a carriage return. Treated the same as a blank. Wherever a blank is legal, a [[Return]] is legal. A [[Return]] is not legal inside a scope password.

User Input Rules

When running SDMAIN, you can enter all commands, subcommands, etc., in either uppercase or lowercase letters. However, all lowercase letters are upshifted when they are processed. Therefore, names such as "TaxRate," "taxrate," and "TAXRATE," all refer to the same object.

NOTE Passwords and character strings (attribute values) are case-sensitive. They are NOT shifted (up OR down), and are read exactly as entered.

User-Defined Names

All user-defined dictionary names must be 32 characters or less. The valid characters for user-defined names are shown below:

All alphanumeric characters (all lowercase characters are upshifted)

All special characters EXCEPT:

```
, ; : . ) " = > < ^ !
```

Scope Password

Scope passwords must be 32 characters or less. All characters are valid in a scope password. However, if the password does not qualify as a valid user-defined name, it must be enclosed in double quotes, for example, "password" .

Special Character Responses

When entering commands in SDMAIN, you may realize that you have typed the wrong command or have misspelled a command name. In such cases, you will want to have the option to begin typing again and not execute the command.

Each of the following responses to an SDMAIN prompt allow you to correct your error:

[[Control]] X Causes the line on which the [[Control]] X was typed to be ignored. The system displays three exclamation points (! ! !) and then waits for you to re-enter the line.

[[Control]] Y Returns to the highest prompt level. In response to the >> prompt, the command terminates without executing. In response to the > prompt, no action is taken and the > prompt is reissued. If a START command is active, it is cancelled by the [[Control]] Y. If input has been redirected using SDIN, the INCLUDE command, or a macro, all redirected levels are exited and input is reset to $STDINX.

The HELP Command

A HELP command provides a quick reference to SDMAIN commands. You can use the HELP command to receive the following information:

- A list of all available commands
- A list of valid subcommands for a particular command
- The syntax for a particular command
- The definition of a macro

HELP Command Environments

What you see after you issue the HELP command depends upon the state of the dictionary when you ask for help. There are three different criteria that affect the responses to the HELP command:

- The mode in which the dictionary is open
- Whether or not a START command has been issued
- Whether the dictionary is a compiled or master dictionary

Before the Dictionary is Opened. Before the dictionary is opened, you cannot use any of the dictionary structure, data maintenance, or reporting commands, but you can use the environment commands such as DEFINE, CONFIGURE, SHOW, etc.

After the Dictionary is Opened. The dictionary may be opened in any of five different modes. These modes are:

- Customization
- Exclusive-update
- Shared-update
- Read-only
- Read-allow-read

In addition, a sixth mode is possible if the dictionary is open in either exclusive update or shared update mode, and a START/SAVE command pair is active. These modes determine what commands you are able to use within SDMAIN,and what commands are displayed when you specify the HELP command.

Customization Mode. When the dictionary is open in customization mode, only environment commands and structure commands are allowed. You have exclusive access to the dictionary and no one else can use it.

Exclusive-Update Mode. When the dictionary is open in exclusive-update mode, you may use the data maintenance commands to create and maintain data occurrences. You can also use the dictionary environment and dictionary reporting commands. You have exclusive access and no one else can have the use of the dictionary while you are using it.

Shared-Update Mode. In shared-update mode, you may use the data maintenance commands to create and maintain data occurrences. You can also use the dictionary environment and dictionary reporting commands. As the name implies, others may also share the use of the dictionary.

Read-Only Mode. When the dictionary is open in read-only mode, you may use the dictionary environment and dictionary reporting commands. The read-only mode also allows other users to share the use of the dictionary.

Read-Allow-Read Mode. When the dictionary is open in read-allow-read mode, you may use the dictionary environment and dictionary reporting commands. The read-allow-read mode allows other users to access the dictionary for reading only.

The START/SAVE Reporting Command Pair. When a START/SAVE command pair is active, you may use only commands involved with building a stored report. Restrictions on whether other users can access the dictionary depend on the open mode:

- **Exclusive Update** --No one else can access the dictionary
- **Shared Update** --Other users can access the dictionary

Compiled or Master Dictionary. If you are working with a compiled dictionary, you are not allowed to modify the dictionary. You may use only the environment and reporting commands.

HELP Examples

In addition to a complete list of available commands, you can also use the HELP command to get the following types of information:

- A list of valid subcommands for a command. (See Example 1 below.)
- A detailed description of a specific command. (See Example 2 below.)
- A list of a user-defined macro.

You can use abbreviations for both the commands and subcommands. The examples below show the kinds of information displayed in the first two cases.

Example 1 - HELP CREATE.

```
>HELP CREATE
```

The output of the above command (entered in the shared-update mode) is as follows:

```
CREATE   -   Create a new object

subcommands:

  DOMAIN       <D  > - Create a new domain

  ENTITY       <E  > - Create a new entity

  RELATIONSHIP <R  > - Create a new relationship

  SCOPE        <S  > - Create a new scope

  SYNONYM      <SYN> - Create a new synonym

  VERSION      <V  > - Create a new version

Enter 'HELP CREATE subcommand-name' for more information
```

Example 2 - HELP CREATE ENTITY.

```
>HELP CREATE ENTITY
```

The output of the above command (entered in Shared-Update mode) is as follows:

CREATE ENTITY - Create a new entity

```
SYNTAX:   CREATE<C> ENTITY<E>   entity-name
          [;INTERNAL<INT> = internal-name]
           ;ENTITY-TYPE<ET> = entity-type-name
          [;ATTRIBUTE-LIST<AL> = ([attribute-name1=[attribute-value1]]
                           [,attribute-name2=[attribute-value2]]
                                     .
                                     .
                                     .
                           [,attribute-nameN=[attribute-valueN]])]
          [;COMMON<C> = common-entity-name]
```

Executing MPE Command

You can issue some MPE commands without exiting the SDMAIN program. To use MPE commands from within SDMAIN, enter a colon (:) followed by the command, in response to the > prompt. From this point on, all syntax follows the MPE rules and conventions. For example, the ampersand (&) is the continuation character, and there is no terminating period. After the MPE command has been processed, control is returned and the SDMAIN prompt is again displayed.

Note that you cannot execute another program from within SDMAIN by using the MPE RUN command. For a list of MPE commands that you can issue from within SDMAIN, see the COMMAND intrinsic shown in the *MPE XL Intrinsics Reference Manual*. If an MPE command cannot be executed from within SDMAIN, you receive an error message.

Dictionary Environment Commands

The first step after issuing the RUN SDMAIN.PUB.SYS command is to open the dictionary and define the environment. The environment of System Dictionary consists of four parts:

- Dictionary environment
- Output environment
- Input environment
- SDMAIN environment

The dictionary environment commands define the four environments as shown below. Any user of the dictionary can use these commands with the dictionary open in any mode. The specific syntax for each of these commands is shown in Chapter 4, "System Dictionary Commands" .

COMMENT	Enters comments into the job stream for documentary purposes.
CONFIGURE	Defines the output environment.
DEFINE	Opens the dictionary and defines the dictionary environment.
EXIT	Terminates the SDMAIN program.
FORMAT	Defines the output environment.
HELP	Provides a quick reference to SDMAIN commands.

INCLUDE	Defines the input environment.
OPTIONS	Defines the SDMAIN environment.
REDO	Allows error correction or changes to the last command issued.
SHOW	Shows a list of all the current environment information.
SHOWMACRO	Shows a list of all the currently defined macros for the system.
SHOWREDO	Displays a list of commands currently on the redo history stack.

Dictionary Structure Commands

System Dictionary is based on the Entity-Relationship model. (See the *HP System Dictionary/XL General Reference Manual, Volume 1* for a complete description.) Its structure consists of **entity types**, **relationship types**, **relationship classes**, and **attributes**. System Dictionary supports a predefined set of **entity types**, **relationship types**, **relationship classes**, and **attributes** known as the **core set**. (See the *HP System Dictionary/XL General Reference Manual, Volume 1* for the complete listing of core set structures.)

Subsystem Support

The purpose of the core set is to provide a standard set of entity types, relationship types, relationship classes, and attributes that are commonly used. This supports the description of the following:

- MPE files
- KSAM files
- Image
- HP IMAGE
- HP SQL
- Vplus forms
- MPE accounting structure
- Network Spooler devices and device classes
- Rapid/V (with some conversion)
- Network directory
- COBOL data definitions
- Pascal data definitions (subset of Pascal data types)
- RPG programs

Additionally, you may customize your dictionary to describe your particular subsystems and applications.

Customizing the Dictionary

Once you begin working in the dictionary, you may discover that a certain entity type needed in your particular installation is not provided in the core set. You may create this entity type by means of the structure command CREATE ENTITY-TYPE. For example, if you need to know which documents are filed in which filing cabinet, you may create the entity type **filing-cabinet** and associate it with (relate it to) the entity type **document**. In order to do this, a new relationship type called **file-cabinet contains document** would also have to be created. The ability to make such changes to the System Dictionary structure is provided by the structure commands. To use these commands, the dictionary must be opened in customization mode and in either the DA scope or a scope with extend capability.

You cannot modify a compiled dictionary. If you specify any of the dictionary structure commands with a compiled dictionary, System Dictionary issues an error message. Table 3-3 lists the structure commands and their subcommands. The specific syntax for each of these commands is shown in Chapter 4, "System

Dictionary Commands" .

Table 3: Dictionary Structure Commands and Subcommands

COMMAND	SUBCOMMAND *
ADD	ENTITY-TYPE-ATTRIBUTE | RELATIONSHIP-TYPE-ATTRIBUTE
CREATE	ATTRIBUTE | ENTITY-TYPE | RELATIONSHIP-CLASS | RELATIONSHIP-TYPE
DELETE	ATTRIBUTE | ENTITY-TYPE | RELATIONSHIP-CLASS | RELATIONSHIP-TYPE
MODIFY	ATTRIBUTE | ENTITY-TYPE | ENTITY-TYPE-ATTRIBUTE | RELATIONSHIP-CLASS | RELATIONSHIP-TYPE | RELATIONSHIP-TYPE-ATTRIBUTE
REMOVE	ENTITY-TYPE-ATTRIBUTE | RELATIONSHIP-TYPE-ATTRIBUTE
RENAME	ATTRIBUTE | ENTITY-TYPE | RELATIONSHIP-CLASS
RESTRUCTURE	

| | |

| RESTRUCTURE | |

| | |

----

* Where | means OR.

The following is a list of the dictionary structure commands and a description of each.

ADD Associates attributes to entity types or relationship types. The attributes and the entity types or relationship types must already exist in the dictionary.

CREATE Creates new entity types, relationship classes, relationship types, and attributes.

DELETE Deletes entity types, relationship classes, relationship types, and attributes.

MODIFY Changes the characteristics of entity types, relationship classes, relationship types, relationship type-attribute pairs, entity type-attribute pairs, and attributes.

REMOVE Disassociates attributes from entity types or relationship types. The opposite of the ADD command.

RENAME Renames an entity type, a relationship class, or an attribute.

RESTRUCTURE Restructures a dictionary, incorporating any structure changes that you have made. Restructuring is done automatically when the EXIT command is issued, when you change the open or name mode, or when you open a new dictionary.

Example

The following example provides an illustration of how to open the dictionary, create a new attribute and entity-type, and then associate the new attribute to the new entity-type.

>DEFINE DICTIONARY = sysdic;
>>SCOPE = dictionary-administrator;
>>PASSWORD = da;
>>OPEN MODE = customization.
	Opens the dictionary with the Dictionary Administrator scope and customization open mode.

>CREATE ENTITY-TYPE terminal.	Creates a new entity-type called terminal.

>CREATE ATTRIBUTE terminal-type;
>>TYPE=c;
>>LENGTH=30.
	Creates a new attribute called terminal-type as a character string containing 30 bytes.

>ADD ENTITY-TYPE-ATTRIBUTE terminal; >>ATTRIBUTE = terminal-type.	Associates the attribute terminal-type to entity-type terminal.
>EXIT	Exits the SDMAIN program, restructuring the dictionary to incorporate the added structure.

Dictionary Data Maintenance Commands

Once you have run SDMAIN, opened the dictionary, and defined the environment, you can then begin entering and updating dictionary data. To do this, however, you must open the dictionary with a scope that grants you the scope rights that you need. For example, if you need to use versions, you must have a scope that grants version capability. (See the *HP System Dictionary/XL General Reference Manual, Volume 1* for a discussion of scope rights.)

In addition, you must open the dictionary in either shared-update or exclusive-update open mode. These are the only two modes in which you are allowed to enter and modify data, manage security (create scopes, etc.), use version capability, and use the domain feature. You cannot modify a compiled dictionary. If you specify any of the dictionary data maintenance commands with a compiled dictionary, System Dictionary issues an error message. Table 3-4 lists the data maintenance commands and their subcommands. The specific syntax for each of these commands is shown in Chapter 4, "System Dictionary Commands."

Table 4: Dictionary Data Maintenance Commands and Subcommands

COMMAND	SUBCOMMAND *
ADD	SCOPE-DOMAIN | SCOPE-ENTITY | SCOPE-RELATIONSHIP
COPY	ENTITY | RELATIONSHIP | VERSION
CREATE	DOMAIN | ENTITY | RELATIONSHIP | SCOPE | SYNONYM | VERSION
DELETE	DOMAIN | ENTITY | RELATIONSHIP | SCOPE | SYNONYM | VERSION
EDIT	ENTITY | RELATIONSHIP
MODIFY	DOMAIN | ENTITY | RELATIONSHIP | SCOPE | VERSION | SCOPE-ENTITY | SCOPE-RELATIONSHIP
REMOVE	SCOPE-DOMAIN | SCOPE-ENTITY | SCOPE-RELATIONSHIP
RENAME	DOMAIN | ENTITY | SCOPE | VERSION

Table 4: Dictionary Data Maintenance Commands and Subcommands

COMMAND	SUBCOMMAND *
RENUMBER	:
RESEQUENCE	
SETVERSION	

* Where | means OR

The following is a list of the dictionary data maintenance commands and a description of each.

ADD Associates domains, entities, and relationships to a security scope and specifies the access rights of the scope to the added entities and relationships.

COPY Creates a new entity (or version) and copies the source object's attributes and relationships (and entities) to the new entity (or version).

CREATE Creates new entities, relationships, synonyms, scopes, versions, and domains.

DELETE Deletes entities, relationships, synonyms, scopes, versions, and domains.

EDIT Edits variable length attribute text belonging to entities and relationships.

MODIFY Modifies entities, relationships, scopes, scope-entities, scope-relationships, versions, and domains.

REMOVE Removes domains, entities, and relationships from a scope. The opposite of the ADD command.

RENAME Renames entities, scopes, versions, and domains.

RENUMBER Assigns new numbers to the relationship position attribute of the relationships of a relationship type.

RESEQUENCE Changes the order of relationships. Resequences the relationships of a relationship type.

SETVERSION Sets a version to one of three statuses: **test**, **production**, or **archival**.

Data Creation

Before you actually begin creating and manipulating dictionary data, you should be familiar with the core set and the various entity types, relationship types, etc., that are available. (Refer to the *HP System Dictionary/XL General Reference Manual, Volume 1* for a complete description of the core set.) Once you have done this, you are then ready to begin creating those entities and relationships that you need in your dictionary.

One important aspect of data creation is the creation of entity names. Refer to the *HP System Dictionary/XL General Reference Manual, Volume 1* for discussions on entity naming, aliases, and synonyms.

Data Maintenance

After you have created entities, relationships, scopes, versions, etc., you can modify or delete them using dictionary commands such as DELETE, EDIT, MODIFY, RENAME, etc. The dictionary must be open in exclusive-update mode in order to delete, modify, or rename scopes, domains, and versions.

Example

The example below shows the logical sequence of data creation, data association, and data maintenance using many of the available data maintenance commands listed in table Table 3-4.

>DEFINE DICTIONARY = sysdic;	Opens the dictionary.
>>SCOPE = dictionary-administrator;	
>>PASSWORD = da;	
>>OPEN-MODE = shared-update;	
>>DOMAIN = domain1;	
>>STATUS = test.	
>CREATE SCOPE personnel;	Creates a new security scope called **personnel**, with **secure**,
>>SCOPE-RIGHTS = s,c,v;	**create**, and **version** capabilities, and with **p** as the password.
>>PASSWORD = p.	
>CREATE DOMAIN personnel-domain;	Creates a new domain called **personnel-domain** with a new version called **testversion** linked to the common version **version1**.
>>VERSION = testversion; COMMON = version1.	
>DEFINE	Reopens SYSDIC (which is the default if no dictionary is
>>SCOPE = personnel;	specified) with the new scope, version, and domain that have just been created.
>>PASSWORD = p;	
>>OPEN-MODE = exclusive-update;	
>>VERSION = testversion;	
>>DOMAIN = personnel-domain.	
>CREATE ENTITY employee-record;	Creates a new entity called **employee-record** of entity-type **record**.
>>ENTITY-TYPE = record;	
>>ATTRIBUTE-LIST = (byte-length = 55,	
>>DESCRIPTION="This record	
>>contains employee data").	
>CREATE ENTITY last-name;	Creates new entities called **last-name**, **first-name**, **date-**

hired, and **social-security-no** of entity-type **element**.

```
>>ENTITY-TYPE = element;
>>ATTRIBUTE-LIST = (element-type = X,
>>BYTE-LENGTH = 10,
>>COBOL-ALIAS = emp-last-name).
>CREATE ENTITY first-name;
>>ENTITY-TYPE = element;
>>ATTRIBUTE-LIST = (element-type = X,
>>BYTE-LENGTH = 20,
>>COBOL-ALIAS = emp-first-name).
>CREATE ENTITY date-hired;
>>ENTITY-TYPE = element;
>>ATTRIBUTE-LIST = (element-type = 9,
>>BYTE-LENGTH = 6,
>>BLANK = yes,
>>DESCRIPTION = "date in mmddyy format").
>CREATE ENTITY social-security-no;
>>ENTITY-TYPE = element;
>>ATTRIBUTE-LIST = (element-type = 9,
>>BYTE-LENGTH = 9,
>>COBOL-ALIAS = emp-ss-no).
>CREATE RELATIONSHIP employee-record,
last-name;
>>RELATIONSHIP-TYPE = record, element;
>>RELATIONSHIP-CLASS = contains.
```

Creates relationships of type **record contains element by relating last-name**, **first-name**, **date-hired**, and **social-security-no**, **to employee-record**.

```
>CREATE RELATIONSHIP employee-record, first-name;
>>RELATIONSHIP-TYPE = record, element;
>>RELATIONSHIP-CLASS = contains.
>CREATE RELATIONSHIP employee-record, date-hired;
>>RELATIONSHIP-TYPE = record, element;
>>RELATIONSHIP-CLASS = contains.
>CREATE RELATIONSHIP employee-record,
>>social-security-no;
```

```
>>RELATIONSHIP-TYPE = record, element;

>>RELATIONSHIP-CLASS = contains.

>MODIFY ENTITY last-name; ENTITY-TYPE = element;   Modifies attribute byte-length for entity last-
                                                  name.

>>ATTRIBUTE-LIST = (BYTE-LENGTH = 20).

>CREATE SYNONYM surname; ENTITY-TYPE = element;    Creates a synonym called surname for el-
                                                  ement last-name.

>>ENTITY = last-name.

>RESEQUENCE employee-record, social-security-no;   Resequences the order of elements in the relation-
                                                  ship record

>>RELATIONSHIP-TYPE = record, element;            contains element by putting social-security-no before

>>RELATIONSHIP-CLASS = contains;                  last-name.

>>BEFORE-ENTITY = last-name.

>RENAME VERSION testversion,V00-00.               Renames the version from testversion to V00-00.

>SETVERSION V00-00; status = production.          Sets the version V00-00 to production status.

>EXIT.                                            Terminates the program.
```

Dictionary Reporting Commands

System Dictionary provides the capability to report on both the dictionary structure and the dictionary data. Two options exist when retrieving information about the dictionary structure: (1) you can retrieve all structure information, or (2) you can retrieve only the names of specified entries. However, you can produce complex reports when retrieving information on dictionary data. Data reporting allows retrieval of information on both data definitions (such as entity definitions) and relationship information. You can specify and store reports on dictionary data for future use. After a report is stored, you can execute it at any time by using the EXECUTE command.

To report on dictionary structure, the dictionary may be open in any open mode. However, to report on dictionary data, the dictionary must be open in read-only, shared-update, or exclusive-update mode. The System Dictionary reporting commands allow great flexibility through use of various selection criteria. These criteria allow selection of fields to be returned through use of relational and Boolean operators. For example, you can retrieve all values that start with an E or a G and end with a D. These selection criteria are discussed later in this chapter.

You cannot modify a compiled dictionary. The only dictionary reporting commands you can specify with a compiled dictionary are DISPLAY, EXECUTE, and REPORT. If you specify any of the other dictionary reporting commands, System Dictionary issues an error message. Table 3-5 lists the dictionary reporting commands and subcommands in System Dictionary. A detailed description of each command is located in Chapter 4, "System Dictionary Commands."

Table 5: Dictionary Reporting Commands and Subcommands

COMMAND	SUBCOMMAND *
COPY	REPORT

Table 5: Dictionary Reporting Commands and Subcommands

COMMAND	SUBCOMMAND *
DELETE	REPORT
DISPLAY	ATTRIBUTE | DOMAIN | ENTITY-TYPE | RELATIONSHIP-CLASS | ENTITY-TYPE-ATTRIBUTE | RELATIONSHIP-TYPE | RELATIONSHIP-TYPE-ATTRIBUTE | REPORT | SCOPE | VERSION
EDIT	REPORT
EXECUTE	
MODIFY	REPORT
RENAME	REPORT
REPORT	ENTITY | RELATIONSHIP
SAVE	
START	

* Where | means OR The following is a list of the dictionary reporting commands and a description of each.

COPY Makes copies of reports that have been stored.

DELETE Deletes a previously stored report from the dictionary.

DISPLAY Generates reports on dictionary structure, in contrast to the REPORT command, which generates reports on dictionary data. There are two available options: (1) the report lists structure names only, or (2) the report lists structure names, plus all other associated information.

EDIT Edits a stored report definition or description.

EXECUTE Processes a stored report.

MODIFY Allows the addition of a description to a report or the modification of an existing report description or scope owner.

RENAME Allows the name of a stored report to be changed.

REPORT Generates reports on dictionary data, in contrast to the DISPLAY command, which generates reports on dictionary structure.

SAVE Stores a report definition and also marks the end of this definition.

START Marks the beginning of a report definition.

Selection Criteria

The SDMAIN reporting commands allow a variety of selection criteria that provide flexibility when producing reports. Unless otherwise noted, you can use any of the selection criteria discussed below with any parameter for the commands that return information from the dictionary (the DISPLAY, EXECUTE, and REPORT commands). These criteria allow qualification of fields of the returned items using relational and Boolean operators.

Any keyword (or attribute qualifier in the ATTRIBUTE-LIST keyword) that is omitted (except for the keyword flags) means that any value for that item is acceptable to you. For example, if displaying attributes and TYPE = R is specified, but no LENGTH keyword is declared, the returned attribute is accepted if it is a real value, regardless of its length. However, if LENGTH = 4 is specified, then only real values of length 4 are returned.

Wild Card Characters. You can enter a string of characters with "wild card" characters embedded. The "wild card" characters have the following meanings:

^	Match any 0 or more characters
!	Match any SINGLE character

You can use the characters at any point within the character string. Additionally, you can use them in any combination. For example, to specify that one or more characters are allowed in a particular position, you can use an "!" for one character followed by a "^" for the additional characters.

Relational and Boolean Operators. Relational and Boolean operators are available and operate on character or numeric values according to the following rules:

expression --> [expression connector] primary

primary --> [operator] value | (expression)

where | means OR

operator An optional field that can contain one of the following

relational operators:

=	Equal to
<>	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

If an operator is omitted, the default "= " (equal to) is used.

value A character string (with possible embedded wild card characters) or numeric value to be compared to values retrieved from the dictionary.

If the value specified is rejected as a valid name in the dictionary (contains an illegal character, blanks, or is longer than 32 characters), then it must be enclosed within quotation marks. (See "User-Defined Names" in Chapter 3 for a description of valid and invalid characters.)

connector A Boolean connection between qualifying values. The following connectors are allowed:

AND	Selected field must contain both values.
OR	Selected field must contain one or both values.

Operator Precedence. The order in which the selection criteria isprocessed is:

1 Operator qualifying the value

2 Expressions enclosed in parentheses

3 AND

4 OR

Examples. The following are examples showing the use of selection criteria.

PROD^ and ^NO Values that begin with PROD and end with NO are selected.

PROD^ or ^ACCT^ Values that begin with PROD or contain ACCT somewhere in them are selected.

A^Z Values that begin with an A and end with a Z with any number of any characters between are selected.

M!! Values that consist of M followed by any two characters are selected.

A!^ Values that consist of A followed by one or more characters are selected.

<= B or >= Y Values that are less than or equal to B or are greater than or equal to Y are selected.

45 or > 60 Values that are equal to 45 or are greater than 60 are selected.

E^ OR G^ AND ^D Values that begin with an E or that begin with a G and end with a D are selected.

(E^ OR G^) AND ^D Values that begin with an E or a G and end with a D are selected.

(10) OR (<15 AND >12) Values that are equal to 10 or are between 12 and 15 are selected.

Example

The following example illustrates the use of System Dictionary reporting commands:

```
>DEFINE DICTIONARY                   Opens the dictionary in exclusive-update mode in the
>>SCOPE = data-administrator;         latest production version of the common domain.
>>PASSWORD = da;
>>OPEN-MODE = exclusive-update.
>CONFIGURE OUTPUT = rptfile.     Configures the output environment with rptfile as the output file.
>DISPLAY ENTITY-TYPE terminal.      Displays the entity-type terminal.
>DISPLAY ATTRIBUTE a^;         Lists the names of all attributes beginning with the letter "a" .
>>NAME-ONLY.
>START              Marks the beginning of a report definition. Command execution is disabled until the
                    SAVE command is encountered, marking the end of the report definition.
>CONFIGURE                Specifies a 15-space right margin and a 10-line top margin, when
>>RIGHT = 15;             output is directed somewhere other than the terminal.
>>TOP = 10.
>REPORT ENTITY;           The reporting command that defines what information appears on the report.
>>ENTITY-TYPE = element.  In this case, the report provides information on all entities of entity-type
                           element.
>SAVE element-report;      Saves the report with the name element-report and an appropriate
```

>>DESCRIPTION = "   Report to provide complete data description. Command execution is again enabled.

>>about occurrences of entity-type element.".

>EXECUTE element-report.   Executes the stored report.

Dictionary Attribute Prompting Facility

SDMAIN provides an attribute prompting facility that prompts for attribute values whenever you issue a CREATE, MODIFY, or REPORT command without the ATTRIBUTE-LIST parameter. For the CREATE command, all possible attributes that can have values assigned to them are displayed for the command in the menu format. For the MODIFY command, all possible attributes that can be changed are displayed, and for the REPORT command, all attribute values that can be qualified (to limit what is retrieved) are displayed.

Attribute prompting does not occur for the CREATE ENTITY or CREATE RELATIONSHIP command if the entity or relationship being created already exists in the dictionary. Attribute prompting does not occur for the MODIFY ENTITY or MODIFY RELATIONSHIP command if the entity or relationship being modified does not exist. Attribute prompting is not performed when input is from a file or when you are executing in batch mode.

Prompting can be turned ON or OFF by using the OPTIONS command. When you run SDMAIN interactively, prompting is initially ON. While prompting is ON, the prompt menu is issued only when the ATTRIBUTE-LIST parameter is omitted from a command that allows it (so ATTRIBUTE-LIST = () disables prompting for the command without providing any attribute values). When you do not run SDMAIN interactively, you cannot turn prompting ON. Attribute prompting consists of a menu of all the attributes you can choose to provide values for and a prompt for indicating which attributes you choose. You then enter the numbers of the attributes desired and the system prompts for values for those attributes. After you enter all the values, the system appends a string with an equivalent ATTRIBUTE-LIST clause, to the command entered and executes the command. Each of these steps is described in detail in the following discussion.

The Menu

The first thing you see when using the attribute prompting facility is the menu. It consists of a one-line header describing the use of the attribute values to be entered and a list of attributes, each with a number. You can choose any combination of attributes for any command. Number 1 on all menus is the all option which prompts for all of the displayed attributes. Numbering for the remaining prompts, however, can vary (and often does) from menu to menu. A sample menu for creating an entity of entity-type RECORD is displayed below:

```
Choose the attribute(s) to be assigned a value

 1 all               8 VPLUS-ALIAS          15 ENTRY-TEXT

 2 SENSITIVITY       9 STANDARD-ALIAS       16 HEADING-TEXT

 3 ID-NUMBER         10 HP SQL-ALIAS        17 PATH-REPORT

 4 BYTE-LENGTH       11 DEFAULT             18 HP-CONDITION-VALUE

 5 COBOL-ALIAS       12 DESCRIPTION         19 HP SQL-SELECT-COMMAND

 6 IMAGE-ALIAS       13 EDIT-MASK

 7 PASCAL-ALIAS      14 ENTITY-LONG-NAME
```

Menu Choices

After the attribute menu is displayed, the prompting facility displays the menu prompt (>>) and waits to receive a selection of numbers from the menu. After you enter values for the specified attributes, the menu prompt is again displayed until you enter either a period (.) or a [[RETURN]]. The following table shows the valid responses to an attribute menu prompt:

INPUT **ACTION**

A number Prompts for a value for the indicated attribute. EXAMPLE: 2

A list of numbers Prompts for a value for each of the indicated attributes separated by blanks or commas. EXAMPLE: 2,3 or 6

all option On all menus, menu option 1 is the all option. When you enter 1, the system prompts for values for each of the attributes in the menu. When the prompting is completed, this option proceeds immediately with the execution of the command instead of returning to the menu prompt. EXAMPLE: 1.

A range of numbers Prompts for a value for each of the attributes (including the attributes associated with the beginning and ending number) within the specified range. A range is of the form: *beginning number : ending number* where the ending attribute number is greater than the beginning attribute number. EXAMPLE: 2 : 4

H[ELP] Displays a short description of the syntax of user input at the menu and attribute prompts.

A blank line or a period Terminates prompting and begins execution of the command.

Attribute Values

After you choose an attribute, the system prompts for a value for the attribute and provides the current or default value in parentheses after the attribute name prompt (for example, SENSITIVITY (PRIVATE)>>). When using the CREATE command, the default value is displayed in parentheses the first time you choose that attribute. If you choose that attribute again before the command is executed, the value you previously entered at this attribute prompt is displayed in parentheses. When using the MODIFY command, the value currently assigned in the dictionary is displayed in parentheses. If you select that attribute again, the value you entered previously for this attribute prompt is displayed. The value that you enter must follow the same rules that apply in command mode. Values you enter during attribute prompting are validated to insure they are legal for the attribute type. No check is made against the edit value of the attribute. That check is done during execution of the command. The following values have special meaning when entered in response to value prompting:

INPUT **ACTION**

[[Return]] The attribute is skipped as if it were never selected. If the attribute has not been previously selected during this command, the attribute is NOT included in the ATTRIBUTE-LIST; otherwise, the value assigned at the previous prompt is used.

.(a period) The value of the current attribute is set as if you entered a [[Return]] and prompting of the current range (if one is active) is stopped. If it is a simple range, no more attributes in that range are prompted for, and the system moves on to the next attribute. If the selection is the all option, no more attributes are prompted for and the command executes immediately.

/ (a slash) For the CREATE and MODIFY command the attribute is set to its default value. For the REPORT command the attribute is set to accept any value. Any value previously assigned by prompting is deleted. This is the equivalent of entering attribute= without a value in the ATTRIBUE-LIST clause of the command.

Executing the Command

After all attribute values have been entered, the system creates an ATTRIBUTE-LIST clause that corresponds to the information provided. The system appends this clause to the end of the command so that it is present if you use REDO to edit or reenter the command. The system then processes the command.

Example

The following example creates element quantity. It assigns read sensitivity, an element-type of I, a display-length of 5, a byte-length of 2, and a count of 10. No value is assigned to the decimal or units fields even though a prompt is issued for values (the first because a [[Return]] is entered and the second because a . is entered to stop prompting for the range 4:13). Note also the multiple lines of prompting for description. You are prompted for multiple lines as long as quoted string has been started. Finally, a [[Return]] is entered at the menu prompt and the command executes to create the element quantity with the provided attribute values.

```
>CREATE ENTITY quantity;

>>ENTITY-TYPE = element.

Choose the attribute(s) to be assigned a value

 1 all              11 BLANK            21 DESCRIPTION

 2 SENSITIVITY      12 JUSTIFY          22 EDIT-MASK

 3 ID-NUMBER        13 SYNCHRONIZE      23 ENTITY-LONG-NAME

 4 ELEMENT-TYPE     14 COBOL-ALIAS      24 ENTRY-TEXT

 5 DISPLAY-LENGTH   15 IMAGE-ALIAS      25 HEADING-TEXT

 6 DECIMAL          16 PASCAL-ALIAS     26 PATH-REPORT

 7 BYTE-LENGTH      17 VPLUS ALIAS      27 HP-CONDITION-VALUE

 8 COUNT            18 STANDARD-ALIAS   28 HP SQL-SELECT-COMMAND

 9 UNITS            19 HP SQL-ALIAS

 10 SIGN            20 DEFAULT
>>2

SENSITIVITY (PRIVATE)>>read

>>4:13, 21

ELEMENT-TYPE ( )>>I

DISPLAY-LENGTH (0)>>5

DECIMAL (0)>>

BYTE-LENGTH (0)>>2

COUNT (1)>>10
```

```
  UNITS ( )>>.
  DESCRIPTION>>"Quantity is an array holding the quantity ordered by a
  DESCRIPTION>>company over the last ten orders.  It is used to help
  DESCRIPTION>>figure the sales discount allowed on the next sale."
  >>
  >
>EXIT                         Terminates the program.
```

4 System Dictionary Commands

Overview

This section includes descriptions of all SDMAIN commands with their syntax, parameters, and examples. These commands are listed alphabetically for your ease of use.

The HELP command provides a quick reference to SDMAIN commands. Using this command, you can either get a list of all available commands, or a detailed description of a particular command.

Abbreviations for commands are allowed at any point. Appendix B lists each command and its abbreviation.

ADD ENTITY-TYPE-ATTRIBUTE

Adds attributes to an entity type's attribute list.

Syntax

```
  A[DD] E[NTITY-]T[YPE-]A[TTRIBUTE] entity-type-name
     ;A[TTRIBUTE] = attribute-name1
             [,attribute-name2 ]
                 .
                 .
                 .
             [,attribute-nameN ]
             .
```

Parameters

entity-type-name Name of the entity type to which attributes are added.

attribute-name(N) Name of the attribute to be added.

Description

All attributes and the entity type specified must already exist in the dictionary.

You can specify any number of attributes, but the maximum number of attributes associated with an entity type may not be more than 128. No attribute may appear more than once in an entity type's attribute list. If you specify more than one attribute, the operation continues until the end of the attribute list, even if an error occurs with any previous attribute.

Open Mode: Customization

Scope DA scope or any scope with extend capability

Example

The following example adds the attributes terminal-type and manufacturer to the entity type terminal.

```
>ADD ENTITY-TYPE-ATTRIBUTE terminal;
>>ATTRIBUTE = terminal-type, manufacturer.
>
```

ADD RELATIONSHIP-TYPE-ATTRIBUTE

Adds attributes to a relationship type's attribute list.

Syntax

```
A[DD] R[ELATIONSHIP-]T[YPE-]A[TTRIBUTE] entity-type-name1
                                  ,entity-type-name2
                                 [,entity-type-name3 ]
                                 [,entity-type-name4 ]
                                 [,entity-type-name5 ]
                                 [,entity-type-name6 ]
    [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
     ;A[TTRIBUTE] = attribute-name1
               [,attribute-name2 ]
                    .
                    .
                    .
               [,attribute-nameN ]
               .
```

Parameters

entity-type-name(N) Name of the entity type involved in the relationship type to which attributes are added.

relation-class-name Name of the relationship class.

attribute-name(N) Name of the attribute to be added.

Description

All attributes and the relationship type you specify must already exist in the dictionary.

You can specify any number of attributes, but the maximum number of attributes associated with a relationship type may not be more than 128. No attribute may appear more than once in a relationship type's attribute list. If you specify more than one attribute, the operation continues until the end of the attribute list, even if an error occurs with any previous attribute.

Open Mode: Customization

Scope: DA scope or any scope with extend capability

Example

The following example adds the attributes size and functionality to the relationship type program contains module.

```
>ADD RELATIONSHIP-TYPE-ATTRIBUTE program, module;
>>RELATIONSHIP-CLASS = contains;
>>ATTRIBUTE = size, functionality.
>
```

ADD SCOPE-DOMAIN

Allows a scope to have access to a domain.

Syntax

```
A[DD] S[COPE-]D[OMAIN] scope-name
      D[OMAIN] = domain-name.
```

Parameters

scope-name Name of the scope given access to the domain.

domain-name Name of the domain to associate with the scope.

Description

Explicit access can be given only if it is not already allowed by the sensitivity of the domain.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or the domain's owner scope

Example

The following example allows the scope candidate to have access to the domain extra-domain.

```
>ADD SCOPE-DOMAIN candidate;
>>DOMAIN = extra-domain. >
```

ADD SCOPE-ENTITY

Allows a scope to have access to an entity.

Syntax

```
A[DD] S[COPE-]E[NTITY] scope-name
  ;E[NTITY] = entity-name
  ;E[NTITY-]T[YPE] = entity-type-name
 [;S[COPE-]A[CCESS] = scope-access ]
 .
```

Parameters

scope-name Name of the scope given access to the entity.

entity-name Name of the entity to associate with the scope.

entity-type-name Name of the type of the entity.

scope-access The type of access the scope has to the entity. The two types of access that are allowed are:

read The scope can only read the entity information.

modify The scope may modify the entity. Read access is automatically assigned.

The default is read.

Description

Explicit access can be given only if it is not already allowed by the sensitivity attribute of the entity. In addition, the scope to be given access must have sufficient capability. That is, at least read capability to be allowed read access and at least create capability to be allowed modify access.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or the entity's owner scope

Example

The following example allows the scope purchasing to have modify access to the element product-number.

```
>ADD SCOPE-ENTITY purchasing;
>>ENTITY = product-number;
>>ENTITY-TYPE = element;
>>SCOPE-ACCESS = modify.
>
```

ADD SCOPE-RELATIONSHIP

Allows a scope to have access to a relationship.

Syntax

A[DD] S[COPE-]R[ELATIONSHIP] *scope-name*

;R[ELATIONSHIP] = *entity-name1*

,entity-name2

[,*entity-name3*]

[,*entity-name4*]

[,*entity-name5*]

[,*entity-name6*]

;R[ELATIONSHIP-]T[YPE] = *entity-type-name1*

,*entity-type-name2*

[,*entity-type-name3*]

[,*entity-type-name4*]

[,*entity-type-name5*]

[,*entity-type-name6*]

[;R[ELATIONSHIP-]C[LASS] = *relation-class-name*]

[;S[COPE-]A[CCESS] = *scope-access*]

.

Parameters

scope-name Name of the scope given access to the relationship.

entity- name(N) Name of the entity involved in the relationship to associate with the scope.

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

scope-access The type of access the scope has to the relationship. The two types of access that are allowed are:

read The scope may only read the relationship information.

modify The scope may modify the relationship. Read access is automatically assigned.

The default is read.

Description

Explicit access can be given only if it is not already allowed by the sensitivity attribute of the relationship. In addition, the scope to be given access must have sufficient capability. That is, at least read capability to be allowed read access and at least create capability to be allowed modify access.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or the relationship's owner scope

Example

The following example allows the scope manufacturing to have modify access to the relationship stockroom processes inventory of relationship type module processes module.

```
>ADD SCOPE-RELATIONSHIP manufacturing;
>>RELATIONSHIP = stockroom,inventory;
>>RELATIONSHIP-TYPE = module,module;
>>RELATIONSHIP-CLASS = processes;
>>SCOPE-ACCESS = modify.
>
```

COMMENT

Allows the addition of descriptions and/or instructions into the command stream for documentation purposes.

Syntax

```
    COM[MENT] [text ] [.]
```

You can specify zero or more characters after the COMMENT command. The comment must be contained on one line. If the comment needs to span more than one line, you must begin each line with the word COMMENT. A period following the command is optional.

The system ignores all lines preceded by COMMENT and simply writes them to the log file before prompting for the next line. The only time that comments are not written to the log file is during processing of a START/SAVE reporting command pair. Thus, once you issue the START command, no comments are logged until after you issue the SAVE command.

Open Mode: Any

Scope: Any

Example

The following example adds three lines of comments to the command stream and log file.

```
    >COMMENT    The following commands define all elements that

    >COMMENT    exist in the SALES-REP data set and adds them

    >COMMENT    to that data set.

    >
```

CONFIGURE

Defines the output environment, such as screen and paper sizes, spacing information, page breaks, and output destination.

Syntax

```
    CON[FIGURE]

         [;S[CREEN-]L[ENGTH] = screen-length ]

         [;S[CREEN-]W[IDTH] = screen-width ]

         [;P[AGE-]L[ENGTH] = page-length ]

         [;P[AGE-]W[IDTH] = page-width ]

         [;LEFT = left-margin ]

         [;RIGHT = right-margin ]

         [;TOP = top-margin ]

         [;BOT[TOM] = bottom-margin ]

         [;OUT[PUT] = output-dest ]

         [;S[INGLE-]S[PACE]]
```

```
[;D[OUBLE-]S[PACE]]
[;PAGE          ]
.
```

Parameters

screen-length Number of rows on the terminal.

screen-width Number of columns on the terminal.

page-length Number of rows on a page of paper.

page-width Number of columns on a page of paper.

left-margin Number of spaces in the left margin.

right-margin Number of spaces in the right margin.

top-margin Number of blank lines in the top margin.

bottom-margin Number of blank lines in the bottom margin.

output-dest Destination of dictionary output. If file SDOUT is already redirected, the current output file is closed and SDOUT is now redirected to *output-dest.*

If you enter the command in response to a command prompt, the file is defined as the output file until you redefine it or for the duration of the session (whichever comes first). If the file is declared in a stored report, it is defined as the output file for the duration of that stored report only. After the completion of the command, the output file reverts to the same file as before the execution of the stored command.

There are three options for this parameter:

TERMINAL The output appears on the screen (or to the batch job listing if

running in batch mode)

PRINTER The output is sent to the line printer (dev=LP)

file-name The output is written to the specified file. If the file does not exist, a new file by that name is created. If the file exists, the new data is appended to the end of the file. You can also use file equate and back-referencing to define this file, that is, *file1.

Because "TERMINAL" and "PRINTER" are reserved words, they have special meaning in SDMAIN and define either the terminal or the printer, respectively, as the output destination. Accordingly, output cannot be redirected to a file with either of these names.

SINGLE-SPACE Specifies single spacing of dictionary output between items. Information about each item is always single-spaced after the item name. This is the default spacing option.

DOUBLE-SPACE Specifies double spacing of dictionary output between items. Information about each item is always single-spaced after the item name.

PAGE Specifies placement of each item on a new page. When the end of an item is encountered, the remainder of the page is filled with blank lines and the next item is positioned at the top of the following page. Information about each item is always single-spaced after the item name.

Description

Because all parameters are optional, if you do not specify any keywords, all parameters are reset to their default values (the original parameter values when the program was initially run). If you specify one or more keywords, then those parameters you do not specify are not affected.

All numeric values must be in the range 0 .. 32767.

When configuring the right and left margins and screen and paper widths, there must be at least 53 characters per row of printable space between the margins to allow for the printing of the statically formatted reports produced by the DISPLAY commands. This allows 32 characters for names, along with up to 21 characters for the line headers.

If you use the CONFIGURE command inside a START/SAVE reporting command pair, you may enter values for all keywords. However, if CONFIGURE is processed by a Nested EXECUTE command, the OUTPUT and PAGE-LENGTH keywords are ignored and the values for these fields are carried over from the previous level.

If you redefine the dictionary environment (for example, switch scopes, open a new dictionary), it is not necessary to reissue the CONFIGURE command if this information remains the same for the new dictionary environment.

You can use the page and screen length and top and bottom margin clauses to manipulate paging of output. For example, if you do not want output to page on the terminal (stop at the end of each screen), simply set the screen length to a large value such as 10,000, (the maximum value is 32767). The output then pauses only after that many lines have scrolled by on the screen (in this case, 10,000) for a given command.

To send the output to a file (with no page spacing) set the top and bottom margins to 0. In this case, only the final page of each command is blank filled so that the next command starts on a new page (determined by page-length).

To eliminate all spacing (including the blank-fill at the end of the command), set the top and bottom margins to 0 and the page length to 1.

You can use other combinations to provide the desired output spacing.

Open Mode: Any

Scope: Any

Example

The following example sets the paper page width at 102, the right and left margins at 12 spaces each, specifies double spacing between items, and sends the output to the printer.

```
>CONFIGURE PAGE-WIDTH =102;

>>LEFT = 12;

>>RIGHT = 12;

>>DOUBLE-SPACE;

>>OUTPUT = PRINTER.

>
```

Example

The following example double spaces all output (every line is double spaced, not just between items as in example 1) and sends the output to the printer.

```
>CONFIGURE PAGE-LENGTH = 2;
>>TOP = 1;
>>BOTTOM = 0;
>>OUTPUT = PRINTER.
>
```

COPY ENTITY

Creates a new entity and copies the source entity's attributes to the new entity.

Syntax

```
CO[PY] E[NTITY] source-entity-name
            ,target-entity-name
     ;E[NTITY-]T[YPE] = entity-type-name
    [;INT[ERNAL] = internal-name ]
    .
```

Parameters

source-entity-name	Name of the entity to be copied.
target-entity-name	External name of a new entity that has attribute values copied to it.
entity-type-name	Name of the type of the entity to be copied.
internal-name	Internal name of the target entity. If not specified, the internal name is the same as the *target-entity-name*.

Description

The target entity cannot already exist.

If the source entity is linked to a common entity, then the target entity is also linked to that common entity.

Note that this command does not copy the values of attributes of type alias. Since alias attributes are alternate names used in external subsystems, a new set of alternate names is needed for the new entity.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability and read access to the source entity. The scope with which the dictionary is open is the owner of the new entity.

Example

The following example copies the attributes of the entity ship-date to the new entity purchase-date.

```
>COPY ENTITY ship-date, purchase-date;
>>ENTITY-TYPE = element.
```

>

COPY RELATIONSHIP

Creates a new entity and copies the source entity's attributes and relationships to the new entity.

Syntax

```
  CO[PY] R[ELATIONSHIP] source-entity-name
              ,target-entity-name
     ;R[ELATIONSHIP-]T[YPE] = entity-type-name1
                   ,entity-type-name2
                  [,entity-type-name3 ]
                  [,entity-type-name4 ]
                  [,entity-type-name5 ]
                  [,entity-type-name6 ]
     [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
     [;POS[ITION] = relation-position ]
     [;INT[ERNAL] = internal-name ]
     .
```

Parameters

source-entity-name Name of the entity whose attributes and relationships are to be copied.

target-entity-name External name of a new entity that has attribute values and relationship associations copied to it.

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

relation-position An integer (from 1 to 6) that specifies the position of the entity type involved in the relationship type corresponding to the source entity. Default is for the source entity to correspond to the first entity type in the relationship type.

internal-name Internal name of the target entity. If not specified, the internal name is the same as the *target-entity-name*.

Description

The target entity cannot already exist.

If the source entity or source entity's relationships are linked to an entity or relationships in the common domain, then the target entity or target entity's relationships are also linked to the same entity or relationships in the common domain.

Note that this command does not copy the values of attributes of type alias. Since alias attributes are alternate names used in external subsystems, a new set of alternate names are needed for the new entity and relationships.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability and read access to the source entity. The scope with which the dictionary is open is the owner of the new entity and relationships.

Example

The following example creates a new entity personnel-record of entity type record. This entity has the same attributes as employee-record. It also creates relationships of elements related to personnel-record identical to those related to employee-record.

```
>COPY RELATIONSHIP employee-record, personnel-record;
>>RELATIONSHIP-TYPE = record, element;
>>RELATIONSHIP-CLASS = contains.
>
```

COPY REPORT

Creates a new report and copies the source report's definition and description to the new report.

Syntax

```
CO[PY] REP[ORT]  source-report-name
           ,target-report-name
  [;INT[ERNAL] = internal-name ]
  .
```

Parameters

source-report-name Name of the report to be copied.

target-report-name External name of a new report that has the same definition and description as the *source-report-name.*

internal-name Internal name of the target report. If not specified, the internal name is the same as the *target-report-name.*

Description

The target report cannot already exist.

Open Mode: Shared-update or exclusive-update

Scope: DA or any scope with create capability. The scope with which the dictionary is open is the owner of the new report.

Example

The following example creates the new report sales-report and copies the description and definition from the existing report accounting-report to the new report.

```
>COPY REPORT accounting-report, sales-report.
>
```

COPY VERSION

Creates a new version and copies all the entities and relationships of the source version to the new version.

Syntax

```
CO[PY] V[ERSION] source-version-name
                       ,target-version-name
        [;INT[ERNAL] = internal-name ] .
```

Parameters

source-version-name Name of the version to be copied.

target-version-name External name of the new version that has entities and relationships copied to it.

internal-name Internal name of the target version. If not specified, the internal name is the same as the *target-version-name.*

Description

The target version cannot already exist.

If the source version is linked to a version in the common domain, the target version is linked to that same version. The new version's status is always automatically set to test, even if the source version's status is production or archival. Refer to the SETVERSION command for information about changing the status of a version.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with version capability. The scope with which the dictionary is open is the owner of the new version.

Example

The following example copies all entities and relationships in version test1 to the new version test2.

```
 >COPY VERSION test1, test2
 >
```

CREATE ATTRIBUTE

Creates a new attribute.

Syntax

```
    C[REATE] A[TTRIBUTE] attribute-name
          [;INT[ERNAL] = internal-name ]
           ;T[YPE] = attribute-data-type
          [;LEN[GTH] = attribute-length ]
          [;E[DIT-]V[ALUE] = attr-edit-value1
                      [,attr-edit-value2 ]
                            .
```

.

.

[,*attr-edit-valueN*]]

.

Parameters

attribute-name External name of the attribute to be created.

internal-name Internal name of the attribute to be created. If not specified, the internal name is the same as the *attribute-name*.

attribute-data-type The attribute data type. Valid data types are:

alias

boolean

character

floating

integer

variable

attribute-length Length of the attribute. Legal values are:

Data Type	Length
alias	32
boolean	1
character	1 - 255
floating	4 or 8
integer	2 or 4
variable	0 (undefined)

If the attribute-data-type is a character, floating, or integer, the length is required. Otherwise, this clause is optional and the length is set at the only possible value.

attr-edit-value(N) The edit values for the attribute. The first value you specify is the default value used whenever you specify an occurrence using this attribute and do not provide a value. If there is only one value, it is only a default and no edit check is performed. However, if there is more than one value, the values you specify during CREATE ENTITY, CREATE RELATIONSHIP, MODIFY ENTITY, and MODIFY RELATIONSHIP commands must match one of the values specified here. If the value you specify through one of these commands is not one of the edit values, an error is returned. Edit values are not allowed for data types alias and variable.

The edit values are a list of values separated by commas. If the type is boolean, only the values true and false are allowed. If the type is character, any character string less than or equal to the length is allowed. (If you use any character not allowed in a user-defined name, you must enclose the value in quotes.) Finally, if the type is either floating or integer, any single number or a number range of the form "number : number" is allowed as an entry in the list. If you specify a range as the first entry in the list, the first number of the range is the default.

Description

To avoid possible name conflicts with any future extensions to the core set, do not create any attributes with a name starting with the characters "HP" .

Open Mode: Customization

Scope: DA scope or any scope with extend capability. The scope with which the dictionary is open is the owner of the attribute.

Example

The following example creates the attribute terminal-type of data type character, with a length of 30. Only the values TTY and GRAPHICS are allowed as values for the attribute with TTY as the default.

```
>CREATE ATTRIBUTE terminal-type;

>>TYPE = character;

>>LENGTH = 30;

>>EDIT-VALUE = TTY, GRAPHICS.

>
```

CREATE DOMAIN

Creates a new domain.

Syntax

```
C[REATE] D[OMAIN] domain-name

    [;INT[ERNAL] = internal-name ]

     ;V[ERSION] = version-name

    [;V[ERSION-]INT[ERNAL] = intern-version-name ]

    [;C[OMMON] = common-version-name ]

    [;SEN[SITIVITY] = sensitivity ]

    .
```

Parameters

domain-name External name of the domain to be created.

internal-name Internal name of the domain to be created. If not specified, the internal name is the same as the *domain-name*.

version-name External name of the version to create for this domain.

intern-version-name Internal name of the version created for this domain. If not specified, the internal name is the same as the *version-name*.

common-version-name Name of the common domain version to which this local domain version is linked. This keyword is not allowed if the dictionary is open in the common domain. If COMMON is omitted, no link to a version in the common domain is established.

sensitivity Specifies whether other scopes are allowed to access the domain. Valid values are:

private	Only the DA and owner scope can access the domain.
public	Any scope may access the domain.

The default is public.

Description

The version created with the domain has its status set to test. Refer to the SETVERSION command for information about changing the status of a version.

To avoid possible name conflicts with any future extensions to the core set, do not create any domains with a name starting with the characters "HP" .

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with domain capability. The scope with which the dictionary is open is the owner of the domain.

Example

The following example creates the new domain testdomain1 as a public domain with a version testversion that is linked to the version version1 in the common domain.

```
>CREATE DOMAIN testdomain1;

>>VERSION = testversion;

>>COMMON = version1.

>
```

CREATE ENTITY

Creates a new entity.

Syntax

C[REATE] E[NTITY] *entity-name*

;E[NTITY-]T[YPE] = *entity-type-name*

[;INT[ERNAL] = *internal-name*]

[;A[TTRIBUTE-]L[IST]=([

attribute-name1 =[*attribute-value1*]]

[,*attribute-name2* =[*attribute-value2*]]

.

.

.

[,*attribute-nameN* =[*attribute-valueN*]])]

[;C[OMMON] = *common-entity-name*]

.

Parameters

entity-name External name of the entity to be created.

entity-type-name Name of the type of the entity to be created.

internal-name Internal name of the entity to be created. If not specified, the internal name is the same as the *entity-name.*

attribute-name(N) Name of the attribute to be assigned a value.

attribute-value(N) The value to be assigned to the attribute. If the attribute is a variable length attribute, you must specify the value text within quotes. Also, if the attribute is of type alias or character, and the attribute value includes invalid characters (See "User-Defined Names" in Chapter 3) or is greater than 32 characters in length, you must specify the attribute value within quotes.

common-entity-name Name of the entity in the common domain whose attributes are to be shared by the local domain entity being created. This keyword is not allowed if the dictionary is open in the common domain. If you specify this parameter, then the attribute list can contain only the following attributes:

sensitivity Specifies the access rights to the entity

id-number A user-specified identification number that is never checked for uniqueness by System Dictionary

If you specify attributes other than these in the attribute list, then you must not use this parameter.

Description

System Dictionary automatically assigns values for the following attributes:

scope-owner

date-created

date-changed

scope-changed

If you do not specify the attribute name, or if you specify the name but do not assign a value, the default value from the attribute's edit values is used. If the attribute does not have any edit values, then the System Dictionary default value for the attribute's data type is used.

alias	:	No alias is assigned
boolean	:	FALSE
character	:	ASCII blanks

floating : Floating point zero

integer : Binary zero

variable : No value is assigned

If the attribute is of type boolean the only values allowed are true and false (may be abbreviated T and F, respectively).

The valid values for the sensitivity attribute are:

private — Only the owner scope is allowed access to the entity, unless it assigns access to other scopes by associating the entity to the scope by means of the ADD SCOPE-ENTITY command. This is the default.

read — Any scope with read capability may read the entity. The owner scope may, in addition, assign modify access to other scopes by associating the entity to the scope by means of the ADD SCOPE-ENTITY command.

modify — Any scope with read capability may read the entity. Any scope with create capability may read and modify the entity.

You should only specify the attributes sensitivity and id-number for the first version of an entity to be created. If you do not specify sensitivity for the first version of an entity, it defaults to private. If you specify sensitivity or id-number for subsequent versions of an entity, they are ignored. If the entity is to be linked to an entity in the common domain, the sensitivity cannot be greater than the sensitivity of the entity in the common domain.

To link the entity to an entity in the common domain, the current version must be linked to a version in the common domain.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability. The scope with which the dictionary is open is the owner of the entity.

Example

The following example creates the element %account with element-type of 9 and byte-length of 8. In addition, it has the cobol-alias ACCT-NO and a description.

```
>CREATE ENTITY %account;
>>ENTITY-TYPE = element;
>>ATTRIBUTE-LIST = (element-type = 9, byte-length = 8,
>>cobol-alias = ACCT-NO,
>>description = "Each customer has a unique account no.
>>All of the customers' transactions are sorted by
>>date and product number.").
>
```

CREATE ENTITY-TYPE

Creates a new entity type.

Syntax

```
C[REATE] E[NTITY-]T[YPE] entity-type-name
     [;INT[ERNAL] = internal-name ]
     .
```

Parameters

entity-type-name External name of the entity type to be created.

internal-name Internal name of the entity type to be created. If not specified, the internal name is the same as the *entity-type-name*.

Description

Every entity type has the following attributes associated with it after it is created:

scope-owner	scope-changed
date-created	sensitivity
date-changed	id-number

These attributes, called **special attributes**, are associated by default and you can never delete them from the entity type's attribute list.

To avoid possible name conflicts with any future extensions to the core set, do not create any entity types with a name starting with the characters "HP" .

Open Mode: Customization

Scope: DA scope or any scope with extend capability. The scope with which the dictionary is open is the owner of the entity type.

Example

The following example creates the new entity type terminal.

```
>CREATE ENTITY-TYPE terminal.
>
```

CREATE RELATIONSHIP

Creates a new relationship.

Syntax

```
C[REATE] R[ELATIONSHIP] entity-name1
                  ,entity-name2
                 [,entity-name3 ]
                 [,entity-name4 ]
                 [,entity-name5 ]
                 [,entity-name6 ]
     ;R[ELATIONSHIP-]T[YPE] = entity-type-name1
```

```
                  ,entity-type-name2
                  [,entity-type-name3 ]
                  [,entity-type-name4 ]
                  [,entity-type-name5 ]
                  [,entity-type-name6 ]
         [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
         [;A[TTRIBUTE-]L[IST]=([
attribute-name1 =[attribute-value1 ]]

[,attribute-name2 =[attribute-value2 ]]
                        .
                        .
                        .

[,attribute-nameN=[attribute-valueN ]])]
         [;C[OMMON] = common-entity-name1
             ,common-entity-name2
             [,common-entity-name3 ]
             [,common-entity-name4 ]
             [,common-entity-name5 ]
             [,common-entity-name6 ]]
             .
```

Parameters

entity-name(N) Name of the entity that establishes a relationship. You can specify up to six entity names. The order of the entities entered should correspond to the order of the entity types in the relationship type.

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

attribute-name(N) Name of the attribute to be assigned a value.

attribute-value(N) The value to be assigned to the attribute. If the attribute is a variable length attribute, you must specify the value within quotes. Also, if the attribute is of type alias or character, and the attribute value includes invalid characters (See "User-Defined Names" in Chapter 3) or is greater than 32 characters in length, you must specify the value within quotes.

common-entity- Name of the entity involved in the relationship in *name(N)* the common domain whose attributes are to be shared by the local domain relationship being created. This keyword is not allowed if the dictionary is open in the common domain. If you use this parameter, the attribute list can contain only the following attributes:

sensitivity Specifies the access rights to the entity

relationship position The logical order of a child entity (the second entity in the relationship) relative to all other child entities for the same parent entity of the same relationship type

If you specify attributes other than these in the attribute list, then you must not use this parameter.

Description

System Dictionary automatically assigns values for the following attributes:

scope-owner

date-created

date-changed

scope-changed

If you do not specify the attribute name or you specify the name, but do not assign a value, the default value from the attribute's edit values is used. If the attribute does not have any edit values, then the System Dictionary default value for the attribute's data type is used.

alias	:	No alias is assigned
boolean	:	false
character	:	ASCII blank
floating	:	Floating point zero
integer	:	Binary zero
variable	:	No value is assigned

If the attribute is of type boolean, the only values allowed are true and false (may be abbreviated T and F, respectively).

The valid values for the sensitivity attribute are:

private Only the owner scope is allowed access to the relationship, unless it assigns access to other scopes by associating the relationship to the scope by means of the ADD SCOPE-RELATIONSHIP command. This is the default.

read Any scope with read capability may read the relationship. The owner scope may, in addition, assign modify access to other scopes by associating the relationship to the scope by means of the ADD SCOPE-RELATIONSHIP command.

modify Any scope with read capability may read the relationship. Any scope with create capability may read and modify the relationship.

You should only specify the attribute sensitivity for the first version of a relationship to be created. If you do not specify sensitivity for the first version of a relationship, it defaults to private. If you specify sensitivity for subsequent versions of a relationship, it is ignored. If the relationship is to be linked to a relationship in the common domain, the sensitivity cannot be greater than the sensitivity of the relationship in the common domain.

The relationship-position attribute is treated specially by the system. The system uses this attribute as a sequencing number to order relationships with the same parent entity (i.e. to represent the order of elements in a record). Accordingly, two relationships of a type with the same entity in the first position cannot have the same value for relationship-position. To avoid conflicts, if you omit either relationship-position completely or do not specify a value, the system assigns a relationship-position value to place the new relationship at the end of the current list. That is, the system assigns a value greater than the largest currently assigned to a relationship with the same parent entity. Note that this is different from other default values as the default depends on the values assigned to other relationships and is different for each relationship.

To link the relationship to a relationship in the common domain, the current version must be linked to a version in the common domain.

If you assign the relationship an alias, the alias is associated with the second entity in the relationship. Only the second entity in the relationship may have an alias, regardless of how many entities form the relationship. Therefore, relationship aliases are typically useful only with binary relationships.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability. The scope with which the dictionary is open is the owner of the relationship.

Example

The following example creates a new relationship ORDERS contains CUSTOMER of relationship type IMAGE-DATABASE contains IMAGE-DATASET. It is linked to the relationship SALES contains SHIP-TO (of the same type) in the common domain.

```
>CREATE RELATIONSHIP orders, customer;

>>RELATIONSHIP-TYPE = image-database,image-dataset;

>>RELATIONSHIP-CLASS = contains;

>>COMMON = sales, ship-to.

>
```

CREATE RELATIONSHIP-CLASS

Creates a new relationship class.

Syntax

```
C[REATE] R[ELATIONSHIP-]C[LASS] relation-class-name

     [;INT[ERNAL] = internal-name ]

     .
```

Parameters

relation-class-name External name of the relationship class to be created.

internal-name Internal name of the relationship class to be created. If not specified, the internal name is the same as the *relation-class-name*.

Description

To avoid possible name conflicts with any future extensions to the core set, do not create any relationship classes with a name starting with the characters "HP" .

Open Mode Customization

Scope: DA scope or any scope with extend capability. The scope with which the dictionary is open is the owner of the relationship class.

Example

The following example creates the new relationship class generates.

```
>CREATE RELATIONSHIP-CLASS generates.
>
```

CREATE RELATIONSHIP-TYPE

Creates a new relationship type.

Syntax

```
C[REATE] R[ELATIONSHIP-]T[YPE] entity-type-name1
                        ,entity-type-name2
                       [,entity-type-name3 ]
                       [,entity-type-name4 ]
                       [,entity-type-name5 ]
                       [,entity-type-name6 ]
    ;R[ELATIONSHIP-]C[LASS] = relation-class-name
    .
```

Parameters

entity-type-name(N) Name of the entity type that establishes the relationship type.

relation-class-name Name of the relationship class.

Description

Every relationship type has the following attributes associated with it after it is created:

scope-owner scope-changed

date-created sensitivity

date-changed relationship-position

These attributes, called **special attributes**, are associated by default and can never be deleted from the relationship type's attribute list.

A minimum of two entity types and a maximum of six entity types can be related in a relationship type. The order of the entity types is important when creating a new relationship type. When only two entity types are involved, the first entity type is the subject (parent) and the second is the object (child). For example, if the first entity type is FILE, the second is ELEMENT, and the relationship class is contains, then the relationship type is interpreted as FILE contains ELEMENT.

Open Mode: Customization

Scope: DA scope or any scope with extend capability. The scope with which the dictionary is open is the owner of the relationship type.

Example

The following example creates a new relationship type project contains program, with contains being the relationship class of the relationship type.

```
>CREATE RELATIONSHIP-TYPE project, program;
>>RELATIONSHIP-CLASS = contains.
>
```

CREATE SCOPE

Creates a new scope.

Syntax

```
C[REATE] S[COPE] scope-name
     [;INT[ERNAL] = internal-name ]
      ;S[COPE-]R[IGHTS] = scope-rights1
                  [,scope-rights2 ]
                       .
                       .
                       .
                  [,scope-rightsN ]
      ;P[ASSWORD] = [password-parameter ]
      .
```

Parameters

scope-name External name of the scope to be created.

internal-name Internal name of the scope to be created. If not specified, the internal name is the same as the *scope-name*.

scope-rights(N) The capability to be given to the scope. The following capabilities are available:

create The scope can create, modify, and delete entities and relationships, depending on security restrictions. Read capability is automatically assigned.

domain The scope can create, modify, and delete domains. Version capability is automatically assigned.

extend The scope can create, modify, and delete entity types, relationship types, relationship classes, and attributes to extend, or customize, the dictionary structure.

read	The scope can read entities and relationships accessible to the scope.
secure	The scope can create, modify, and delete other scopes.
version	The scope can create, modify, and delete versions.

password-parameter Password of the scope to be created. If you do not specify a password, a blank password is assigned for the scope. Any characters are allowed in a password. If a character is desired in the password that is not valid in other System Dictionary names, you must enter the password inside a pair of quotes to allow recognition of the restricted characters. (See "Scope Password" in Chapter 3.)

Remember that the case of the entered password also counts when the password is checked.

Description

The scope being created cannot be assigned any capabilities (scope rights) that the current open scope does not have.

To avoid possible name conflicts with any future extensions to the core set, do not create any scopes with a name starting with the characters "HP" .

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with secure capability. The scope with which the dictionary is open is the owner of the scope.

Example

The following example creates a new scope tester with read capability and the password %Main.

```
>CREATE SCOPE tester;

>>SCOPE-RIGHTS = Read;

>>PASSWORD = %Main.

>
```

CREATE SYNONYM

Creates a new synonym for an entity.

Syntax

C[REATE] SYN[ONYM] *synonym-name*

[;INT[ERNAL] = *internal-name*]

;E[NTITY-]T[YPE] = *entity-type-name*

;E[NTITY] = *entity-name*

.

Parameters

synonym-name Name of the synonym to be created.

internal-name Internal name of the synonym to be created. If not specified, the internal name is the same as the *synonym-name*.

entity-type-name Name of the type of the entity for which to create a synonym.

entity-name Name of the entity for which to create a synonym.

Description

The new synonym refers to the same attribute list as the entity primary name and applies to all versions of the entity.

You can use synonyms only to directly reference an entity (a relationship) in the dictionary. You cannot use them to do qualified retrievals using the reporting commands. Accordingly, the name reported for the entity in a qualified retrieval is always the primary name.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or the entity's owner scope. The scope with which the dictionary is open is the owner of the synonym.

Example

The following example creates a new synonym purch-date for the element order-date.

```
>CREATE SYNONYM purch-date;
>>ENTITY-TYPE = element;
>>ENTITY = order-date.
>
```

CREATE VERSION

Creates a new version.

Syntax

C[REATE] V[ERSION] *version-name*

[;INT[ERNAL] = *internal-name*]

[;C[OMMON] = *common-version-name*]

.

Parameters

version-name External name of the version to be created.

internal-name Internal name of version to be created. If not specified, the internal name is the same as the *version-name*.

common-version-name Name of the common domain version to which this local domain version is linked. This keyword is not allowed if the dictionary is open in the common domain. If omitted when in a local domain, no link is established.

Description

The version status of the created version is automatically set to test. Refer to the SETVERSION command for information about changing the status of a version.

Note that if no link to a common version is established, no entity or relationship can be linked to a common entity/relationship unless the version is modified to establish a common link.

Open Mode:	Shared-update or exclusive-update
Scope:	DA scope or any scope with version capability. The scope with which the dictionary is open is the owner of the version.

Example

The following example creates a new version A_00_09 and links it to the version A_00_01 in the common domain.

```
>CREATE VERSION A_00_09;

>>COMMON = A_00_01.

>
```

DEFINE

Opens the dictionary and defines the dictionary environment.

Syntax

```
DEF[INE] [;DICT[IONARY] = dictionary-filename ]

     [;S[COPE] = scope-name ]

     [;P[ASSWORD] = [scope-password ]]

     [;O[PEN-]M[ODE] = open-mode ]

     [;N[AME-]M[ODE] = name-mode ]

     [;D[OMAIN] = [domain-name ]]

     [;V[ERSION] = version-name  ]

     [;STAT[US] = version-status ]

      .
```

Parameters

dictionary-filename File name of the dictionary to open. If you do not specify a value when opening the dictionary, the default for this parameter is SYSDIC. You can access a remote dictionary using the DEFINE command and the DICTIONARY parameter. This method is described in the discussion part of this command.

scope-name Name of the scope that definitions are assigned to. Required for the initial opening of the dictionary.

scope-password Gives access to the scope. Any characters are allowed in a password. If a character is desired in the password that is not valid in other System Dictionary names, you must enter the password inside a pair of quotes to allow recognition of the restricted characters. (See "Scope Password" in Chapter 3.) If you specify the keyword with no value, a blank password is used.

Remember that the case of the password you enter also counts when the password is checked. If you specify the SCOPE parameter and not a PASSWORD parameter, you are prompted for the password. For security reasons, the echo is turned off and you are given three chances to enter the correct password. If you are in session mode and you do not enter the correct password after three tries, the DEFINE command terminates with no change to the dictionary environment. If you are in batch mode and you do not enter the correct password after three tries, the program terminates. The password is always read from $STDINX.

open-mode Specifies the mode with which to open the dictionary. Valid choices are:

- read-allow-read Allows you to read System Dictionary data definitions, but does not allow the creation, deletion, or modification of these data definitions. Others can access the dictionary only for reading.
- read-only Allows you to read System Dictionary data definitions, but does not allow the creation, deletion, or modification of these data definitions. Others can also access the dictionary.
- shared-update Allows you to read, create, modify, or delete System Dictionary data definitions. Others can also access the dictionary.
- exclusive-update Allows you to read, create, modify, or delete System Dictionary data definitions. No one else can access the dictionary.
- customization Allows you to interact with the dictionary structure by using structure commands. No one else can access the dictionary.

The default open mode for a dictionary that is not open is read-only.

name-mode Specifies which group of names you wish to use. Valid choices are:

internal

external

The default for the first open is external.

domain-name Name of the domain used for creating or retrieving definitions. If you do not specify the keyword, the default is the common domain. To return to the common domain from another domain, specify the keyword with no value. This parameter is applicable only if the dictionary is open in read-allow-read, read-only, shared-update, or exclusive-update mode.

version-name Name of the version used for creating or retrieving definitions. If you do not specify this parameter, the version status parameter is used to determine the default version. This parameter is applicable only if the dictionary is opened in read-allow-read, read-only, shared-update, or exclusive-update mode.

The default version for the first open is determined by the setting of the *version-status* parameter.

version-status Version status indicating to choose the latest version of the status as the version to be used for creating or retrieving definitions. Valid choices are:

test

production

archival

The default status for the first open is production. The *version-status* parameter is used only if you do not specify *version-name*.

Description

Once the dictionary is open you may redefine the environment while in a session. For example, you may open a new dictionary, or the same dictionary with a different scope or open mode. It is not necessary to respecify any parameters that remain the same after redefining the environment. For example, if you open a new dictionary, but the scope, open mode, etc., are to be the same as in the previous DEFINE command, the previous values carry over from the first DEFINE command.

If the dictionary was initially opened in customization mode and the new DEFINE calls for a change of the dictionary or open mode, a RESTRUCTURE of the dictionary is automatically done. The restructure operation may take some time if many dictionary occurrences are affected by the structure changes.

Open Mode: Any

Scope: Any

Remote Dictionary Access

System Dictionary includes a feature that allows you to access dictionaries on remote systems, a remote system being any other system than the one to which you are currently logged on. The uses for this feature are explained in the *HP System Dictionary/XL General Reference Manual, Volume 1* (32256-90004). The following information explains how to use this feature using the DEFINE command.

A remote dictionary is accessed by first establishing a session on a remote system, as shown in the example below. The following names are used in the example:

Name	Represents
REMOTESYS	The name of the remote system
CORPINFO	The account on the remote system that contains the specified that contains the specified dictionary
MGR	The user in account CORPINFO
COM.MFGCO	The communications system being used
:DSLINE REMOTESYS	Establishes communications to the REMOTESYS.COM.MFGCO
ENVIRONMENT 1:	(Returned) name of the established line)
:REMOTE HELLO MGR.CORPINFO	Remote logon
ENTER ACCOUNT PASSWORD:	Password is "blind" for security reasons
:	

For more information on system to system communications, refer to the *NS3000/XL User/Programmer Reference Manual* or the *NS3000/XL Network Manager Reference manual*.

RUNNING WITH NS3000

After the remote session has been established, the remote dictionary can be accessed by including the remote system name in the dictionary name. For example, the DEFINE command could be specified in the following form:

```
DEFINE DICTIONARY=SYSDIC.PUB.SYS:NODEB;
```

It is also possible to use a file equation to completely identify the remote dictionary, as shown in the following example.

```
FILE RDICT1=SYSDIC.PUB.CORPINFO:REMOTESYS
```

where RDICT1 would be passed to the DICTIONARY parameter when using the DEFINE command, as shown in the following example.

```
DEFINE DICTIONARY=RDICT1
```

RUNNING WITH DS3000

When either the local or remote system (or both) is a DS3000 node the above methods of addressing the remote dictionary do not work. Instead, the device number of the remote node must be used, as shown in the following example.

```
FILE RDICT1=SYSDIC.PUB.CORPINFO;DEV=17#
```

where 17 is the device number of the remote system, and must be known to the user. The DEFINE command can then be used as follows:

```
DEFINE DICTIONARY=RDICT1
```

NOTE The dictionary on the remote system must be compatible with the System Dictionary software on the local system, that is, it must be the same version. If it is not, the DEFINE command will fail.

Note also, that as with any subsystem or program being accessed on a remote system, the performance of System Dictionary is likely to be slower than when run on a local system.

Example

The first of the following examples opens the dictionary sysdic for the scope manager in customization open mode. The password Mgr* allows access to the scope.

The second of the following examples changes the current scope leaving the dictionary open in the same state otherwise. Note that the password was prompted for (with echo off) since the SCOPE keyword clause is specified without a PASSWORD clause.

```
>DEFINE DICTIONARY = sysdic;

>>SCOPE = manager;

>>PASSWORD = Mgr*;

>>OPEN-MODE = customization.

>

>DEFINE SCOPE = da.

Scope Password>>

>
```

DELETE ATTRIBUTE

Deletes an attribute.

Syntax

```
DEL[ETE] A[TTRIBUTE] attribute-name
     [;PURGE]
     .
```

Parameters

attribute-name Name of the attribute to be deleted.

PURGE If you specify this keyword and any entity types or relationship types that are associated with the specified attribute exist, then the association between the attribute and the entity type or relationship type is also purged. If you do not specify PURGE, and the attribute is associated with an entity type or relationship type, the operation fails.

Description

When an attribute is deleted, the attribute values are deleted from all entities and relationships that previously had the attributes associated to them.

Open Mode: Customization

Scope: DA scope or the attribute's owner scope

WARNING When a version is in archival status, it is considered frozen, protecting its data from modification. However, archival status does not freeze the dictionary structure. Therefore, if the dictionary structure is changed, the structure of archival versions is also changed. In the case of additions to the dictionary structure, no serious problems will result. However, deletions from the dictionary structure may be detrimental to archival versions. Therefore, this should be considered when making deletions that affect dictionary structure.

Example

The following example deletes the attribute time-created. Because the keyword PURGE is not specified, the operation fails if time-created is associated with any entity types or relationship types.

```
>DELETE ATTRIBUTE time-created.

>
```

DELETE DOMAIN

Deletes a domain, including all entities and relationships that belong to that domain.

Syntax

```
DEL[ETE] D[OMAIN] domain-name .
```

Parameters

domain-name Name of the domain to be deleted.

Description

Neither the common domain nor the current domain can ever be deleted. When a domain is deleted, all versions of that domain and all entities and relationships in those versions are deleted.

Open Mode:	Exclusive-update
Scope:	DA scope or the domain's owner scope

Example

The following example deletes the domain accounts-payable.

```
>DELETE DOMAIN accounts-payable.
>
```

DELETE ENTITY

Deletes an entity.

Syntax

```
DEL[ETE] E[NTITY] entity-name
     ;E[NTITY-]T[YPE] = entity-type-name
     .
```

Parameters

entity-name	Name of the entity to be deleted.
entity-type-name	Name of the type of the entity to be deleted.

Description

If no other versions of the entity exist, then the entity name and all synonyms are deleted. All relationships involving the entity are also deleted.

The entity cannot be deleted if it is in the common domain and is linked to a local domain entity.

Open Mode:	Shared-update or exclusive-update
Scope:	DA scope or the entity's owner scope

Example

The following example deletes the file f01.

```
>DELETE ENTITY f01; ENTITY-TYPE = file.
>
```

DELETE ENTITY-TYPE

Deletes an entity type.

Syntax

```
DEL[ETE] E[NTITY-]T[YPE] entity-type-name
```

```
    [;PURGE]
    .
```

Parameters

entity-type-name Name of the entity type to be deleted.

PURGE If you specify this keyword and any relationship types that involve the specified entity type exist, then the relationship types are also deleted. If you do not specify PURGE, and the entity type is involved in a relationship type, the operation fails.

Description

When an entity type is deleted, all entities of that type are deleted. Also, all relationship-types involving the deleted entity-types and all relationships of these types are deleted.

Open Mode: Customization

Scope: DA scope or the entity type's owner scope

WARNING When a version is in archival status, it is considered frozen, protecting its data from modification. However, archival status does not freeze the dictionary structure. Therefore, if the dictionary structure is changed, the structure of archival versions is also changed. In the case of additions to the dictionary structure, no serious problems will result. However, deletions from the dictionary structure may be detrimental archival versions. Therefore, this should be considered when making deletions that affect dictionary structure.

Example

The following example deletes the entity-type terminal. Because the keyword PURGE is specified, if any relationship types that involve the entity type terminal exist, the relationship types are also deleted.

```
>DELETE ENTITY-TYPE terminal;
>>PURGE.
>
```

DELETE RELATIONSHIP

Deletes a relationship.

Syntax

```
DEL[ETE] R[ELATIONSHIP]  entity-name1
               ,entity-name2
              [,entity-name3 ]
              [,entity-name4 ]
              [,entity-name5 ]
              [,entity-name6 ]
    ;R[ELATIONSHIP-]T[YPE] = entity-type-name1
                  ,entity-type-name2
```

[,*entity-type-name3*]

[,*entity-type-name4*]

[,*entity-type-name5*]

[,*entity-type-name6*]

[;R[ELATIONSHIP-]C[LASS] = *relation-class-name*]

.

Parameters

entity-name(N) Name of the entity involved in the relationship to be deleted.

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

Description

The relationship cannot be deleted if it is in the common domain and is

linked to a local domain relationship.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or the relationship's owner scope

Example

The following example deletes the five-way relationship of the image database company, image master dataset division, image detail dataset employee, with the search item loc-code and sort item hire-date.

```
>DELETE RELATIONSHIP
>>employee,loc-code,hire-date,division,company;
>>RELATIONSHIP-TYPE = image-dataset,element,element,
>>image-dataset,image-database;
>>RELATIONSHIP-CLASS = chains.
>
```

DELETE RELATIONSHIP-CLASS

Deletes a relationship class.

Syntax

DEL[ETE] R[ELATIONSHIP-]C[LASS] *relation-class-name*

[;PURGE]

.

Parameters

relation-class-name Name of the relationship class to be deleted.

PURGE — If you specify this keyword and any relationship types that involve the specified relationship class exist, then the relationship type is also purged. If you do not specify PURGE, and the relationship class is involved in a relationship type, the operation fails.

Description

When both relationship class and relationship type are deleted, all relationships of that type are also deleted.

Open Mode: Customization

Scope: DA scope or the relationship class's owner scope

WARNING When a version is in archival status, it is considered frozen, protecting its data from modification. However, archival status does not freeze the dictionary structure. Therefore, if the dictionary structure is changed, the structure of archival versions is also changed. In the case of additions to the dictionary structure, no serious problems will result. However, deletions from the dictionary structure may be detrimental to archival versions. Therefore, this should be considered when making deletions that affect dictionary structure.

Example

The following example deletes the relationship class generates. Because the keyword PURGE is not specified, the operation will fail if generates is involved in any relationship types.

```
>DELETE RELATIONSHIP-CLASS generates.

>
```

DELETE RELATIONSHIP-TYPE

Deletes a relationship type.

Syntax

```
DEL[ETE] R[ELATIONSHIP-]T[YPE]  entity-type-name1
                    ,entity-type-name2
                    [,entity-type-name3 ]
                    [,entity-type-name4 ]
                    [,entity-type-name5 ]
                    [,entity-type-name6 ]
   [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
    .
```

Parameters

entity-type-name(N) — Name of the entity type involved in the relationship type to be deleted.

relation-class-name — Name of the relationship class.

Description

When a relationship type is deleted, all relationships of that type are also deleted.

Open Mode: Customization

Scope: DA scope or the relationship type's owner scope

WARNING When a version is in archival status, it is considered frozen, protecting its data from modification. However, archival status does not freeze the dictionary structure. Therefore, if the dictionary structure is changed, the structure of archival versions is also changed. In the case of additions to the dictionary structure, no serious problems will result. However, deletions from the dictionary structure may be detrimental to archival versions. Therefore, this should be considered when making deletions that affect dictionary structure.

Example

The following example deletes the relationship type project contains program.

```
>DELETE RELATIONSHIP-TYPE project,program;

>>RELATIONSHIP-CLASS = contains.

>
```

DELETE REPORT

Deletes a report.

Syntax

```
DEL[ETE] REP[ORT] report-name .
```

Parameters

report-name Name of the report to be deleted.

Description

Open Mode: Shared-update or exclusive-update

Scope: DA scope or the report's owner scope

Example

The following example deletes the report image-report.

```
>DELETE REPORT image-report.

>
```

DELETE SCOPE

Deletes a scope.

Syntax

DEL[ETE] S[COPE] *scope-name*

[;N[EW-]S[COPE] = *new-scope-name*]

```
        [;P[ASSWORD] = [password-parameter ]]

        .
```

Parameters

scope-name Name of the scope to be deleted.

new-scope-name Name of the scope that is the owner of all dictionary objects previously owned by the deleted scope. The current scope is the default new owner scope if you do not specify one.

password-parameter The password that gives access to the new owner scope. This password is the same password that was assigned to the new scope when it was created. If you omit the password or specify the keyword without a value, a blank password is assumed.

Description

The scope rights of the new scope must be at least as great as the deleted scope.

Open Mode: Exclusive-update

Scope: DA scope or the scope's owner scope

Example

The following example deletes the scope tester and assigns everything it owns to the new owner scope systester which has the password system*.

```
>DELETE SCOPE tester;

>>NEW-SCOPE = systester;

>>PASSWORD = system*.

>
```

DELETE SYNONYM

Deletes a synonym from an entity.

Syntax

```
DEL[ETE] SYN[ONYM] synonym-name

    ;E[NTITY-]T[YPE]= entity-type-name

    .
```

Parameters

synonym-name Name of the synonym to be deleted.

entity-type-name Name of the type of the entity from which to delete a synonym.

Description

Since the synonym name must be unique in the entity name set for the type, the entity name is not needed to delete a synonym.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or the owner scope of the entity to which the synonym is assigned

Example

The following example deletes the element synonym purch-date.

```
>DELETE SYNONYM purch-date;
>>ENTITY-TYPE = element.
>
```

DELETE VERSION

Deletes a version, including all entities and relationships that belong to this version.

Syntax

DEL[ETE] V[ERSION] *version-name* .

Parameters

version-name Name of the version to be deleted.

Description

The current version cannot be deleted. The version cannot be deleted if it is in the common domain and a local domain version is linked to it. All entities and relationships belonging to the version are also deleted.

Open Mode: Exclusive-update

Scope: DA scope or the version's owner scope

Example

The following example deletes the version B_00_08. All entities and relationships belonging to this version are also deleted.

```
>DELETE VERSION B_00_08.
>
```

DISPLAY ATTRIBUTE

Displays one or more attributes.

Syntax

DIS[PLAY] A[TTRIBUTE] [*attribute-name*]

[;T[YPE] = *attribute-data-type*]

[;LEN[GTH] = *attribute-length*]

[;S[COPE-]O[WNER] = *scope-owner-name*]

[;NAME[-ONLY]]

[;NO[SORT]]

.

Parameters

attribute-name Name of the attribute to be displayed.

attribute-data-type Type of the attribute to be displayed. Valid datatypes are:

alias

boolean

character

floating

integer

variable

attribute-length Length of the attribute to be displayed.

scope-owner-name Owner of the attribute to be displayed. If the current scope does not have secure capability, any qualification other than the current scope causes no attributes to be retrieved.

NAME-ONLY If specified, only the name of the qualifying attribute is displayed. If not specified, the name, owner, type, length, and edit values are displayed.

NOSORT Reports on the attributes in the unsorted order in which they are retrieved from the dictionary. Otherwise, the attributes are sorted alphabetically by the attribute names.

Description

When sorting is not required, the use of the NOSORT keyword speeds processing time. If the scope does not have secure capability, and the current scope is not the owner, the scope owner is not displayed.

Open Mode: Any

Scope: Any

Example

The following example displays all attributes with data type character.

```
>DISPLAY ATTRIBUTE;
>>TYPE = CHARACTER.
ACCESS
  Scope-Owner : CORESET
  Type : CHARACTER
  Length : 2
  Edit-Values :
CHAR-TYPE
  Scope-Owner : CORESET
  Type: CHARACTER
  Length : 10
```

```
    Edit-Values : ASCII, EBCDIC
       .             .
       .             .
       .             .
  UNITS
    Scope-Owner : CORESET
    Type : CHARACTER
    Length : 10
    Edit-Values :
    10 Attribute(s) Retrieved
  >
```

DISPLAY DOMAIN

Displays one or more domains.

Syntax

DIS[PLAY] D[OMAIN] [*domain-name*]

[;S[COPE-]O[WNER] = *scope-owner-name*]

[;SEN[SITIVITY] = *sensitivity*]

[;NAME[-ONLY]]

[;NO[SORT]]

.

Parameters

domain-name Name of the domain to be displayed.

scope-owner-name Owner of the domain to be displayed. If the current scope does not have secure capability, any qualification other than the current scope causes no domains to be retrieved.

sensitivity Domains with the indicated sensitivity are displayed. Valid sensitivities are:

private

public

NAME-ONLY If specified, only the name of the qualifying domain displayed. If not specified, the name, owner, and sensitivity are displayed.

NOSORT Reports on the domains in the unsorted order in which they are retrieved from the dictionary. Otherwise, the domains are sorted alphabetically by the domain names.

Description

When sorting is not required, the use of the NOSORT keyword speeds processing time. If the scope does not have secure capability, and the current scope is not the owner, then the scope owner is not displayed.

Open Mode: Read-only, shared-update, or exclusive-update

Scope: DA scope or a scope with domain capability

Example

The following example displays the domain jones which has the sensitivity private.

```
>DISPLAY DOMAIN jones.
JONES
  Sensitivity : PRIVATE
  Scope-Owner : PERSONNEL
  1 Domain(s) Retrieved
>
```

DISPLAY ENTITY-TYPE

Displays one or more entity types.

Syntax

DIS[PLAY] E[NTITY-]T[YPE] [*entity-type-name*]

[;S[COPE-]O[WNER] = *scope-owner-name*]

[;NAME[-ONLY]]

[;NO[SORT]]

.

Parameters

entity-type-name Name of the entity type to be displayed.

scope-owner-name Owner of the entity type to be displayed. If the current scope does not have secure capability, any qualification other than the current scope causes no entity types to be retrieved.

NAME-ONLY If specified, only the name of the qualifying entity type is displayed. If not specified, the name, owner, and attribute list are displayed.

NOSORT Reports on the entity types in the unsorted order in which they are retrieved from the dictionary. Otherwise, the entity types are sorted alphabetically by the entity type name.

Description

When sorting is not required, the use of the NOSORT keyword speeds processing time. If the scope does not have secure capability, and the current scope is not the owner, then the scope owner is not displayed.

Open Mode: Any

Scope: Any

Example

The following example displays the entity type element with its scope owner and attributes.

```
>DISPLAY ENTITY-TYPE element.
ELEMENT
  Scope-Owner : CORESET
  Attributes : DATE-CREATED, DATE-CHANGED, SCOPE-OWNER,
    SCOPE-CHANGED, SENSITIVITY, ID-NUMBER, ELEMENT-TYPE,
    DISPLAY-LENGTH, DECIMAL, BYTE-LENGTH, COUNT, UNITS,
    SIGN, BLANK, JUSTIFY, SYNCHRONIZE
  1 Entity-Type(s) Retrieved
>
```

DISPLAY ENTITY-TYPE-ATTRIBUTE

Displays one or more entity type attribute pairs.

Syntax

```
DIS[PLAY] E[NTITY-]T[YPE-]A[TTRIBUTE] [entity-type-name ]
       [;A[TTRIBUTE] = attribute-name ]
       [;S[COPE-]O[WNER] = scope-owner-name ]
       [;NAME[-ONLY]]
       [;NO[SORT]]
       .
```

Parameters

entity-type-name Name of the entity type of the entity type attribute pair to be displayed.

attribute-name Name of the attribute of the entity type attribute pair to be displayed.

scope-owner-name Owner of the entity type attribute pair to be displayed. If the current scope does not have secure capability, any qualification other than the current scope causes no entity type attribute pairs to be retrieved.

NAME-ONLY If specified, only the name of the qualifying entity type attribute pair is displayed. If not specified, the name and owner are displayed.

NOSORT Reports on the entity types in the unsorted order in which they are retrieved from the dictionary. Otherwise, the entity types are sorted alphabetically by the entity type name.

Description

When sorting is not required, the use of the NOSORT keyword speeds processing time. If the scope does not have secure capability, and the current scope is not the owner, then the scope owner is not displayed.

Open Mode: Any

Scope: Any

Example

The following example displays the attributes linked to the entity type cabinet along with the owners of the links.

```
>DISPLAY ENTITY-TYPE-ATTRIBUTE cabinet.

CABINET

  SCOPE-OWNER

    Scope-Owner : CORESET

  SCOPE-CHANGED

    Scope-Owner : CORESET

  DATE-CREATED

    Scope-Owner : CORESET

  DATE-CHANGED

    Scope-Owner : CORESET

  SENSITIVITY

    Scope-Owner : CORESET

  ID-NUMBER

    Scope-Owner : CORESET

  COUNT

    Scope-Owner : ACCOUNTING

  CABINET-SIZE

    Scope-Owner : FACILITIES

  8 Entity-Type/Attribute Pair(s) Retrieved

>
```

DISPLAY RELATIONSHIP-CLASS

Displays one or more relationship classes.

Syntax

DIS[PLAY] R[ELATIONSHIP-]C[LASS] [*relation-class-name*]

[;S[COPE-]O[WNER] = *scope-owner-name*]

[;NAME[-ONLY]]

[;NO[SORT]]

.

Parameters

relation-class-name Name of the relationship class to be displayed.

scope-owner-name Owner of the relationship class to be displayed. If the current scope does not have secure capability, any qualification other than the current scope causes no relationship classes to be retrieved.

NAME-ONLY If specified, only the name of the qualifying relationship class is displayed. If not specified, the name and owner are displayed.

NOSORT Reports on the relationship classes in the unsorted order in which they are retrieved from the dictionary. Otherwise, the relationship classes are sorted alphabetically by the relationship class name.

Description

When sorting is not required, the use of the NOSORT keyword speeds processing time. If the scope does not have secure capability, and the current scope is not the owner, then the scope owner is not displayed.

Open Mode: Any

Scope: Any

Example

The following example displays the relationship class contains with its scope owner.

```
>DISPLAY RELATIONSHIP-CLASS contains.

CONTAINS

  Scope-Owner : CORESET

  1 Relationship-Class(es) Retrieved

>
```

DISPLAY RELATIONSHIP-TYPE

Displays one or more relationship types.

Syntax

```
DIS[PLAY] R[ELATIONSHIP-]T[YPE] [ entity-type-name1 ]
                    [,entity-type-name2 ]
                    [,entity-type-name3 ]
                    [,entity-type-name4 ]
                    [,entity-type-name5 ]
                    [,entity-type-name6 ]
     [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
     [;S[COPE-]O[WNER] = scope-owner-name ]
     [;NAME[-ONLY]]
```

```
[;NO[SORT]]
.
```

Parameters

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

scope-owner-name Owner of the relationship type to be displayed. If the current scope does not have secure capability, any qualification other than the current scope causes no relationship types to be retrieved.

NAME-ONLY If specified, only the name and class of the qualifying relationship type are displayed. If not specified, the name, class, owner, and attribute list are displayed.

NOSORT Reports on the relationship types by class in the unsorted order in which the classes are retrieved from the dictionary. Otherwise, the relationship classes are sorted alphabetically by the relationship class name.

Description

You can use either nothing between commas (,,) or a ^ between commas (,^,) as a placeholder to match any entity type name in the middle of a list. When sorting is not required, the use of the NOSORT keyword speeds processing time. If the scope does not have secure capability, and the current scope is not the owner, then the scope owner is not displayed.

Open Mode: Any

Scope: Any

Example

The following example displays all relationship types of class contains with image-dataset as the first entity type and any entity types beginning with rec as the second entity type. The owner scopes and a list of all associated attributes are printed for each.

```
>DISPLAY RELATIONSHIP-TYPE image-dataset, rec^;
>>      RELATIONSHIP-CLASS = contains.
IMAGE-DATASET  RECORD
  Rel-Class : CONTAINS
  Scope-Owner : CORESET
  Attributes : DATE-CREATED, DATE-CHANGED, SCOPE-OWNER,
    SCOPE-CHANGED, SENSITIVITY, RELATIONSHIP-POSITION,
    PRIMARY-RECORD
  1 Relationship-Type(s) Retrieved
>
```

DISPLAY RELATIONSHIP-TYPE-ATTRIBUTE

Displays one or more relationship type attribute pairs.

Syntax

DIS[PLAY] R[ELATIONSHIP-]T[YPE-]A[TTRIBUTE] [*entity-type-name1*]
[,*entity-type-name2*]
[,*entity-type-name3*]
[,*entity-type-name4*]
[,*entity-type-name5*]
[,*entity-type-name6*]
[;R[ELATIONSHIP-]C[LASS] = *relationship-class-name*]
[;A[TTRIBUTE] = *attribute-name*]
[;S[COPE-]O[WNER] = *scope-owner-name*]
[;NAME[-ONLY]]
[;NO[SORT]]
.

Parameters

entity-type-name(N) Name of the entity type involved in the relationship type of the relationship type attribute pair to be displayed.

relation-class-name Name of the relationship class.

attribute-name Name of the attribute of the relationship type attribute pair to be displayed.

scope-owner-name Owner of the relationship type attribute pair to be displayed. If the current scope does not have secure capability, any qualification other than the current scope causes no relationship type attribute pairs to be retrieved.

NAME-ONLY If specified, only the names of the qualifying relationship type attribute pair are displayed (remember that a relationship type name includes its relationship class). If not specified, the names and owner are displayed.

NOSORT Reports on the relationship classes in the unsorted order in which they are retrieved from the dictionary. Otherwise, the relationship classes are sorted alphabetically by the relationship class name.

Description

When sorting is not required, the use of the NOSORT keyword speeds processing time. If the scope does not have secure capability, and the current scope is not the scope owner, then the scope owner is not displayed.

Open Mode: Any

Scope: Any

Example

The following example displays the attributes linked to the relationship type cabinet contains elements and the owners of the links.

```
>DISPLAY RELATIONSHIP-TYPE-ATTRIBUTE cabinet, element;
>>RELATIONSHIP-CLASS = contains;
>>SCOPE-OWNER = coreset.
CABINET ELEMENT
  Relationship-Class : CONTAINS
  SCOPE-OWNER
    Scope-Owner : CORESET
  SCOPE-CHANGED
    Scope-Owner : CORESET
  DATE-CREATED
    Scope-Owner : CORESET
  DATE-CHANGED
    Scope-Owner : CORESET
  SENSITIVITY
    Scope-Owner : CORESET
  RELATIONSHIP-POSITION
    Scope-Owner : CORESET
  6 Relationship-Type/Attribute Pair(s) Retrieved
>
```

DISPLAY REPORT

Displays one or more reports.

Syntax

```
DIS[PLAY] REP[ORT] [report-name ]
     [;S[COPE-]O[WNER] = scope-owner-name ]
     [;NAME[-ONLY]]
     [;NO[SORT]]
     .
```

Parameters

report-name Name of the report to be displayed.

scope-owner-name Owner of the report to be displayed.

NAME-ONLY If specified, only the name of the qualifying report is displayed. If not specified, the name, owner, and description of the report are displayed.

NOSORT Reports on the reports in the unsorted order in which they are retrieved from the dictionary. Otherwise, the reports are sorted alphabetically by the report name.

Description

When sorting is not required, the use of the NOSORT keyword speeds processing time.

Open Mode: Read-only, shared-update, or exclusive-update

Scope: Any scope with read capability

Example

The following example displays the report image-database.

```
>DISPLAY REPORT image-database.
IMAGE-DATABASE
  Scope-Owner : DA
  Description : Detailed report on IMAGE databases including data
    about its location, classes, datasets, records, and elements.
  1 Report(s) Retrieved
>
```

DISPLAY SCOPE

Displays one or more scopes.

Syntax

```
DIS[PLAY] S[COPE] [scope-name ]
     [;S[COPE-]R[IGHTS] = scope-right ]
     [;S[COPE-]O[WNER] = scope-owner-name ]
     [;NAME[-ONLY]]
     [;NO[SORT]]
     .
```

Parameters

scope-name Name of the scope to be displayed.

scope-right Right assigned to the scope to be displayed. Valid rights are:

create

domain

extend

read

secure

version

scope-owner-name Owner of the scope to be displayed.

NAME-ONLY If specified, only the name of the qualifying scope is displayed. If not specified, the name, right, password, and owner are displayed.

NOSORT — Reports on the scopes in the unsorted order in which they are retrieved from the dictionary. Otherwise, the scopes are sorted alphabetically by the scope name.

Description

Note that any scope having the specified scope right(s) is selected. For example, if the rights selected are domain and version, any scope that has at least the domain and version capabilities is displayed. Therefore, if scopes with only domain and version capabilities, and no other capabilities, are desired, then the *scope-right* specification must be:

```
SCOPE-RIGHTS = D and V and

S and

E and

C and

R

                  or

SCOPE-RIGHTS = D, V,

S,

E,

C,

R.
```

When sorting is not required, the use of the NOSORT keyword speeds processing time. The scope must have secure capability in order to report on any scopes other than the current scope.

Open Mode: Any

Scope: DA scope, the scope's owner scope, or the current scope

Example

The following example displays all scopes with at least domain and version capabilities. The only one found and displayed is the DA scope which has all scope rights.

```
>DISPLAY SCOPE;

>>SCOPE-RIGHTS = D and V.

DA

  Scope-Rights : S E C R D V

  Password : data-admin

  Scope-Owner : CORESET

  1 Scope(s) Retrieved

>
```

DISPLAY VERSION

Displays one or more versions.

Syntax

```
DIS[PLAY] V[ERSION] [version-name ]
    [;STAT[US] = version-status ]
    [;C[OMMON]] = [common-version-name ]]
    [;LAST[-ONLY]]
    [;S[COPE-]O[WNER] = scope-owner-name ]
    [;NAME[-ONLY]]
    [;NO[SORT]]
    .
```

Parameters

version-name Name of the version to be displayed.

version-status Status of the version to be displayed. Valid statuses are:

test

production

archival

common-version-name Common version linked to the version to be displayed. You should only use this clause in a local domain, as the common link is always blank in the common domain. Specifying COMMON without a value requests a match to all versions that do not have a common link.

LAST-ONLY This flag displays the version last set to each status (or a specific status if indicated). If specified, only the last version set to the specified status is listed. If not specified, all versions set to the specified status are listed.

scope-owner-name Owner of the version to be displayed. If the current scope does not have secure capability, any qualification other than the current scope causes no versions to be retrieved.

NAME-ONLY If specified, only the name of the qualifying version is displayed. If not specified, the name, status, owner, common version, and history of the version are displayed.

NOSORT Reports on the versions in the unsorted order in which they are retrieved from the dictionary. Otherwise, the versions are sorted alphabetically by the version name.

Description

When sorting is not required, the use of the NOSORT keyword speeds processing time. If the scope does not have secure capability, the scope owner is not displayed unless the current scope is the owner.

Open Mode: Read-only, shared-update, or exclusive-update

Scope: Any

Example

The following example displays all versions having a status of either archival or production.

```
>DISPLAY VERSION;
>>STATUS = Archival or Production.
SALES1
  Version-Status : ARCHIVAL
  Scope-Owner : MARKETING
  Common : COMMON1
  History : 02/07/84 10:04 AM  ARCHIVAL
         02/02/84  3:45 PM  PRODUCTION
         02/02/84  3:44 PM  TEST
SALES2
  Version-Status : ARCHIVAL
  Scope-Owner : MARKETING
  Common : COMMON2
  History : 03/01/84  9:56 PM  ARCHIVAL
         02/07/84 11:03 AM  PRODUCTION
         02/04/84 10:53 PM  TEST
SALES3
  Version-Status : ARCHIVAL
  Scope-Owner : MARKETING
  Common : COMMON3
  History : 03/04/84  3:16 AM  ARCHIVAL
         03/01/84  7:23 PM  PRODUCTION
         02/20/84  9:34 PM  TEST
SALES4
  Version-Status : PRODUCTION
  Scope-Owner : MARKETING
  Common : COMMON4
  History : 03/04/84 12:04 AM  PRODUCTION
         03/01/84 12:53 PM  TEST
  4 Version(s) Retrieved
>
```

EDIT ENTITY

Edits an entity's variable length attribute text.

Syntax

ED[IT] E[NTITY] *entity-name*

;E[NTITY-]T[YPE] = *entity-type-name*

;A[TTRIBUTE] = *attribute-name*

.

Parameters

entity-name Name of the entity whose variable length attribute text is to be edited.

entity-type-name Name of the type of the entity.

attribute-name Name of the variable length attribute whose text is to be edited.

Description

You cannot use the EDIT ENTITY command if input is from a file or you are executing in batch mode. The EDIT command calls the EDIT/V subsystem, which performs the specified editing. (Refer to the *EDIT/V Reference Manual* for details regarding this subsystem.) EDTXT*xxx* is the name of the temporary text file used as the interface between the System Dictionary and EDIT/V. To perform the editing tasks, you may use all but the following two EDIT/V subsystem commands: TEXT and KEEP. You cannot use the TEXT command because EDTXT*xxx* is automatically used as the text file when entering the EDIT command. You may not use the KEEP command for the same reason. When exiting the EDIT/V subsystem, you are prompted with "PURGE OLD?". If you respond "Yes" or "Y", an automatic KEEP is issued. Otherwise, the original text remains unchanged and the new text is not kept. If you delete all lines, the variable length attribute is deleted from the entity.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability and modify access to the entity

Example

The following example adds the line "The records in the file are fixed length 80-byte records with 3 fields: a customer ID, a part number, and a quantity ordered." to the description text of entity orders.

```
>EDIT ENTITY orders;
>>ENTITY-TYPE = file;
>>ATTRIBUTE = description.
HP32201A.7.16 EDIT/3000  WED, NOVEMBER 13, 1985, 4:34 PM
(C) HEWLETT-PACKARD CO. 1984
/L ALL
    1    This file contains all of the information
    2    available for an order.
/ADD
```

```
    3    The records in the file are fixed length
    4    80-byte records with 3 fields:  a customer ID,
    5    a part number, and a quantity ordered.
    6    //
/L ALL
    1    This file contains all of the information
    2    available for an order.
    3    The records in the file are fixed length
    4    80-byte records with 3 fields:  a customer ID,
    5    a part number, and a quantity ordered.
/E"
EDTXT0 ALREADY EXISTS - RESPOND YES TO PURGE OLD AND KEEP NEW
PURGE OLD? Y
>
```

EDIT RELATIONSHIP

Edits a relationship's variable length attribute text.

Syntax

```
ED[IT] R[ELATIONSHIP]  entity-name1
                        ,entity-name2
                       [,entity-name3 ]
                       [,entity-name4 ]
                       [,entity-name5 ]
                       [,entity-name6 ]
    ;R[ELATIONSHIP-]T[YPE] = entity-type-name1
                              ,entity-type-name2
                             [,entity-type-name3 ]
                             [,entity-type-name4 ]
                             [,entity-type-name5 ]
                             [,entity-type-name6 ]
   [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
    ;A[TTRIBUTE] = attribute-name
    .
```

Parameters

entity-name(N) Name of the entity involved in the relationship whose variable length attribute text is to be edited.

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

attribute-name Name of the variable length attribute whose text is to be edited.

Description

You cannot use the EDIT RELATIONSHIP command if input is from a file or you are executing in batch mode. The EDIT command calls the EDIT/V subsystem, which performs the specified editing. (Refer to the *EDIT/V Reference Manual* for details regarding this subsystem.) EDTXT*xxx* is the name of the temporary text file used as the interface between the System Dictionary and EDIT/V. To perform the editing tasks, you can use all but the following two EDIT/V subsystem commands: TEXT and KEEP. You cannot use the TEXT command because EDTXT*xxx* is automatically used as the text file when entering the EDIT command. You may not use the KEEP command for the same reason. When exiting the EDIT/V subsystem, you are prompted with "PURGE OLD?”. If you respond "Yes" or "Y", an automatic KEEP is issued. Otherwise, the original text remains unchanged and the new text is not kept. If you delete all lines, the variable length attribute is deleted from the relationship.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability and modify access to the relationship

Example

The following example adds the line "These orders are only from the Midwestern Region." to the description text for the relationship, Midwestern contains order.

```
>EDIT RELATIONSHIP Midwestern, order;

>>RELATIONSHIP-TYPE = region, file;

>>RELATIONSHIP-CLASS = contains;

>>ATTRIBUTE = description.

HP32201A.7.16 EDIT/3000  WED, NOVEMBER 13, 1984, 4:34 PM

(C) HEWLETT-PACKARD CO. 1984

/L ALL

    1    This file contains all of the information

    2    available for an order.

/ADD

    3    These orders are only from the Midwestern

    4    Region.

    5    //

/L ALL
```

```
      1    This file contains all of the information
      2    available for an order.
      3    These orders are only from the Midwestern
      4    Region.
   /E
   EDTXT0 ALREADY EXISTS - RESPOND YES TO PURGE OLD AND KEEP NEW
   PURGE OLD? Y
   >
```

EDIT REPORT

Edits a stored report or its description.

Syntax

```
   ED[IT] REP[ORT] report-name
      [;DESC[RIPTION]]
        .
```

Parameters

report-name Name of the stored report to be edited.

DESCRIPTION If specified, the description of the report is edited. If not specified, the report definition is edited.

Description

You cannot use the EDIT REPORT command if input is from a file or you are executing in batch mode. The EDIT command calls the EDIT/V subsystem, which performs the specified editing. (Refer to the *Edit/V Reference Manual* for details regarding this subsystem.) EDTXT*xxx* is the name of the temporary text file used as the interface between the System Dictionary and EDIT/V. To perform the editing tasks, you can use all but the following two EDIT/V subsystem commands: TEXT and KEEP. You cannot use the TEXT command because EDTXT is automatically used as the text file when entering the EDIT command. You may not use the KEEP command for the same reason. When exiting the EDIT/V subsystem, you are prompted with "PURGE OLD?". If you respond "Yes" or "Y" , an automatic KEEP is issued. Otherwise, the original text remains unchanged and the new text is not kept. When a keep is being processed, if the definition of the report was being changed, it is parsed and any error found reported. If you delete all lines of the description, the description is deleted from the report. If you delete all lines of the definition, the stored report itself is deleted from the dictionary.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or the report's owner scope

Example

The following example modifies the output spacing of the report named image-database-report from page spacing between items to double-spacing between items.

```
   >EDIT REPORT image-database-report.
```

```
HP32201A.7.16 EDIT/3000 WED, NOVEMBER 13, 1985, 4:34 PM
(C) HEWLETT-PACKARD CO. 1984
/L ALL
   1    CONFIGURE PAGE.
   2
   3    REPORT ENTITY;
   4    ENTITY-TYPE = element;
   5    SORT.
/MODIFY 1
   1    CONFIGURE PAGE.
      :       ddddiDOUBLE-SPACE
   1    CONFIGURE DOUBLE-SPACE.
      :
/L ALL
   1    CONFIGURE DOUBLE-SPACE;
   2
   3    REPORT ENTITY;
   4    ENTITY-TYPE = element;
   5    SORT.
/E
EDTXT0 ALREADY EXISTS - RESPOND YES TO PURGE OLD AND KEEP NEW
PURGE OLD? Y
>
```

EXECUTE

Executes a REPORT command that has been previously created and stored by using a START/SAVE pair.

Syntax

```
EX[ECUTE] report-name
      [;E[NTITY] = entity-name          ]
      [;R[ELATIONSHIP] = [entity-name1 ]  ]
      [            [,entity-name2  ]]
      [            [,entity-name3  ]]
      [            [,entity-name4  ]]
      [            [,entity-name5  ]]
```

```
    [             [,entity-name6  ]]
    .
```

Parameters

report-name Name of the stored report to be executed. No selection criteria are allowed.

entity-name Name of the entity to be reported in the executed report. Passes this entity name to the stored REPORT ENTITY command.

entity-name(N) Name of the entity involved in the relationship to be reported in the executed report. Passes the entity name to the stored REPORT RELATIONSHIP command.

Description

Note that you can use either nothing between commas (,,) or a ^ between commas (,^,) as a placeholder to match any *entity-name(N)* in the middle of a list.

Open Mode: Read-only, shared-update, or exclusive-update

Scope: DA scope or any scope that has read access to the data being reported

Example

The following example executes the stored report named image-database-report for the entity named accounting. The report reports the values of all the attributes for the specified entity.

```
>EXECUTE image-database-report;

>>ENTITY = accounting.

ACCOUNTING

  DATE-CREATED : 11/11/84  3:30 AM

  DATE-CHANGED : 11/11/84  3:30 AM

  SCOPE-OWNER : DA

  SCOPE-CHANGED : DA

  SENSITIVITY : PRIVATE

  ID-NUMBER : 0

  IMAGE-DATABASE-TYPE : IMAGE

  1 Entity(ies) Retrieved

>
```

Nested EXECUTE

Executes a secondary REPORT ENTITY command that has been previously created and stored by using a START/SAVE pair. You can only call this command format from the SUB-REPORT keyword of the REPORT ENTITY command. Note that no selection criteria is allowed for this reporting command.

Syntax

EX[ECUTE] *report-name*

```
;R[ELATIONSHIP-]T[YPE] = entity-type-name1
                         ,entity-type-name2
                        [,entity-type-name3 ]
                        [,entity-type-name4 ]
                        [,entity-type-name5 ]
                        [,entity-type-name6 ]
[;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
[;S[OURCE-]P[OSITION] = source-position ]
[;R[EPORT-]P[OSITION] = report-position ]
 .
```

Parameters

report-name Name of the stored report to be executed.

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

source-position An integer (from 1 to 6) that specifies the position in the RELATIONSHIP-TYPE list of the entity type being reported in the primary report. The entity type indicated must match the value specified for ENTITY-TYPE in the primary REPORT command.

The value defaults to the matching entity type (specified in the REPORT ENTITY command) if the type appears only one time in the list. If the entity type appears more than once in the list, you must enter a value to indicate the one desired.

report-position An integer (from 1 to 6) that specifies the position in the RELATIONSHIP-TYPE list of the entity type to be reported in the secondary report. The entity type indicated must match the value specified for ENTITY-TYPE in the secondary (stored) REPORT command. If the relationship type is binary (2-way), the default is to the entity type that is not chosen as the match for the source entity type. However, for all relationships that are not binary relationships, you must supply a value to indicate which entity type is the desired report entity type.

Description

This command allows you to retrieve data about an entity and then, by following relationships in the dictionary, report on related entities.

Open Mode: Read-only, shared-update, or exclusive-update

Scope: DA scope or any scope that has read access to the data being reported

Example

The following report consists of the attribute values of the image-database sales and all the image-datasets that are related to it. The image-database contains image-dataset relationship type is specified as the path to be used to find the image-dataset that is associated to the image-database sales. Note that neither SOURCE-POSITION nor REPORT-POSITION are needed as the source entity type is unambiguous and the specified relationship type is a binary one, respectively.

```
>REPORT ENTITY sales;
>>ENTITY-TYPE = image-database;
>>SUB-REPORT = (EXECUTE image-dataset-report;
>>                  RELATIONSHIP-TYPE = image-database,
>>                                image-dataset;
>>                  RELATIONSHIP-CLASS = contains.).
SALES
  DATE-CREATED : 11/11/84  4:23 AM
  DATE-CHANGED : 11/24/84  5:02 PM
  SCOPE-OWNER : DA
  SCOPE-CHANGED : DA
  SENSITIVITY : PRIVATE
  ID-NUMBER : 0
  IMAGE-DATABASE-TYPE : IMAGE
2) CUSTOMERS
    Entity-Type : IMAGE-DATASET
    DATE-CREATED : 11/11/84  4:23 AM
    DATE-CHANGED : 11/11/84  4:23 AM
    SCOPE-OWNER : DA
    SCOPE-CHANGED : DA
    SENSITIVITY : PRIVATE
    ID-NUMBER : 0
    IMAGE-DATASET-TYPE : MANUAL
2) SUPPLIERS
    Entity-Type : IMAGE-DATASET
    DATE-CREATED : 11/11/84  4:23 AM
    DATE-CHANGED : 11/11/84  4:23 AM
    SCOPE-OWNER : DA
    SCOPE-CHANGED : DA
    SENSITIVITY : PRIVATE
    ID-NUMBER : 0
    IMAGE-DATASET-TYPE : MANUAL
2) ORDERS
    Entity-Type : IMAGE-DATASET
```

```
     DATE-CREATED : 11/11/84  4:23 AM
     DATE-CHANGED : 11/11/84  4:23 AM
     SCOPE-OWNER : DA
     SCOPE-CHANGED : DA
     SENSITIVITY : PRIVATE
     ID-NUMBER : 0
     IMAGE-DATASET-TYPE : DETAIL
2) STOCK
     Entity-Type : IMAGE-DATASET
     DATE-CREATED : 11/11/84  4:23 AM
     DATE-CHANGED : 11/11/84  4:23 AM
     SCOPE-OWNER : DA
     SCOPE-CHANGED : DA
     SENSITIVITY : PRIVATE
     ID-NUMBER : 0
     IMAGE-DATASET-TYPE : DETAIL
   1 Entity(ies) Retrieved
>
```

EXIT

Terminates the SDMAIN program, closing the dictionary if needed.

Syntax

```
E[XIT] [.]
```

Description

If the dictionary is open in customization mode, a restructuring of the dictionary structure occurs before the program terminates. The restructure operation may take some time if many dictionary occurrences are affected by the structure changes.

Open Mode: Any

Scope: Any

Example

The following example exits the program (closing the dictionary and any files used by the system) and returns control to the operating system.

```
>EXIT
END OF PROGRAM
: (Returns to MPE )
```

FORMAT

Defines titles and/or page headers for dictionary reports.

Syntax

F[ORMAT]

[;ALIGN[MENT] = *alignment-option*]

[;F[ORMAT]-V[ARIABLE] = *format-flag*]

[;P[AGE-]H[EADER] = ["*page-header-text* "]]

[;TI[TLE] = ["*title-text* "]]

.

Parameters

alignment-option A value that specifies which alignment option to use with the output data from the dictionary. Valid choices are:

aligned Aligns all values with respect to the longest value header.

fixed-aligned Aligns all values assuming a fixed length header of 15 characters. If any value header (attribute name) is larger than 15 characters, it is truncated at 15 characters and a caret (^) is placed in the 16th position to indicate that the name is truncated.

unaligned Aligns each value independently, immediately after its respective value header. The default is to print the values unaligned.

format-flag A value indicating how the text of variable length attributes is formatted. Valid choices are:

on Formats the variable length attribute text before it is printed.

off Does not format the variable length attribute text before it is printed. The text is printed in the format in which it is stored.

The default is for the text to be formatted. If the text is not formatted and a line will not fit within the page or screen boundaries, the line will wrap-around to the next line.

page-header-text A quoted string that is printed at the top of each page of a report for all output that is sent to a destination other than the terminal. It uses the following format:

string1; string2; string3

where

string1 is set flush to the left margin *string2* is centered on the page *string3* is set flush to the right margin. You can omit any of the strings by leaving them blank. Two special characters are recognized:
^ substitutes the current date and time
! substitutes an ascending page number

title-text A quoted string that is printed on the first page as the title of the report. It is centered on the page with three blank lines before and two lines after.

Description

To delete a value from the PAGE-HEADER or TITLE keyword, specify the keyword with no value. To reset all values to their default, issue the FORMAT command with no keywords. If you redefine the dictionary environment, (that is, switch scopes, open a new dictionary, etc.), you do not need to reissue the FORMAT command if this information remains the same for the new environment.

Open Mode: Any

Scope: Any

Example

The following example sets up a page header including the current date and time at the left margin, the string "January Month-End Report" in the middle, and ascending page numbers at the right margin. The title, alignment, and format-variable options remain unchanged.

```
>FORMAT PAGE-HEADER = "^;January Month-End Report;!".

>
```

HELP

Provides a quick reference to SDMAIN commands by listing the valid commands in the current mode, the valid subcommands for a command, the syntax of a specific command, or the definition of a macro.

Syntax

H[ELP] [*command* [*subcommand*]] [.]

Parameters

command Name of the command or user-defined macro.

subcommand Name of the subcommand.

Description

What you see when you issue the HELP command depends upon three things:

- Whether or not the dictionary is open
- If the dictionary is open, what access mode it is open in
- If the dictionary is open, whether or not the START command has been issued

See Chapter 3 of this manual for a detailed description of the HELP command in relation to the various open modes and the START command. You can also use the HELP command to look at definitions of user-defined macros. The definition of the macro is printed for you so you can check the parameters needed or see what a given macro does (COMMENT commands needed in the macro with a description could be very helpful). You can use the SHOWMACRO command to get a list of all the currently defined macros if you do not remember the macro name.

Open Mode: Any

Scope: Any

Example

The first of the following examples asks for help with the HELP command. The syntax of the HELP is displayed. The second of the following examples asks for the definition of the user-defined macro OPEN.

```
>HELP HELP.

HELP - Returns a list of valid subcommands for a command or a detailed

    description of a specific command

SYNTAX:   HELP<H>   [command [subcommand]]  [.]

>

>HELP OPEN

MACRO Definition:

open mode = shared-update.

define scope = da; open-mode = !mode; status = test.

>
```

INCLUDE

Causes SDMAIN to temporarily stop reading its input from the current input device or file and start reading input from the specified file.

Syntax

IN[CLUDE] *include-file-name* .

Parameters

include-file-name   Name of the file from which to read input.

Description

When the end of file is reached, SDMAIN goes back to the previous input device or file for input. Within an include file, you can have other INCLUDE commands and macro calls. After a macro call is expanded, SDMAIN executes the expanded macro as if it is an include file. The maximum number of nested levels of input at any one time using INCLUDE commands or macro calls is 20. When executing a file specified by the INCLUDE command, the EDIT command and the REDO command are not allowed. Also, attribute prompting is not performed at this time. Commands are echoed to $STDLIST as they are executed from within the include file. The INCLUDE command itself is not written to the log file, but each command read from the include file for processing is written to the log file as if it were being entered interactively. If you enter INCLUDE $STDIN or INCLUDE $STDINX, you can return to the previous level by issuing the MPE command :EOD.

Open Mode: Any

Scope: Any

Example

The following example specifies the file file1 to be the input file. The next command processed by the system is read from file1.

```
>INCLUDE file1.
```

>

MODIFY ATTRIBUTE

Modifies an attribute.

Syntax

```
M[ODIFY] A[TTRIBUTE] attribute-name
     [;S[COPE-]O[WNER] = scope-owner-name ]
     [;LEN[GTH] = attribute-length ]
     [;E[DIT-]V[ALUE] = [ attr-edit-value1
                 [,attr-edit-value2 ]
                     .
                     .
                     .
                 [,attr-edit-valueN ]]]
     .
```

Parameters

attribute-name Name of the attribute to be modified

scope-owner-name Name of the new owner scope

attribute-length Length of the attribute. You cannot modify the length if the data type is alias, boolean, or variable. For the remaining data types, character, floating, and integer, the following length ranges are permissible:

Data Type	Length
character	1 to 255
floating	4 or 8
integer	2 or 4

attr-edit-value(N) The new edit values for the attribute. The first value you specify is the default value used whenever you specify an occurrence using this attribute and you do not provide a value. If there is only one value, it is only a default and no edit check is performed. However, if there is more than one value, the values specified during CREATE ENTITY, CREATE RELATIONSHIP, MODIFY ENTITY, and MODIFY RELATIONSHIP commands must match one of the values specified here. If the value entered through one of these commands is not one of the edit values, an error is returned. Edit values are not allowed for data types alias and variable. The edit values are a list of values separated by commas. If the type is boolean, only the values true and false are allowed. If the type is character, any character string less than or equal to the length is allowed. (If you use any character not allowed in a user-defined name, you must enclose the value in quotes.) If the type is either floating or integer, any single number or a number range of the form "number : number" is allowed (if you specify a range as the first entry in the list, the first number of the range is the default).

To remove all edit values from an attribute, specify the EDIT-VALUE keyword with no edit values. In order to leave attribute edit values unchanged, issue the MODIFY ATTRIBUTE command without the EDIT-VALUE clause.

Description

If you change the edit value of the attribute, there is no check to make sure that any existing values of that attribute meet the new criteria. Accordingly, if you add or change an edit value to be more restrictive, it is your responsibility to check to make sure that all existing values of the attribute are updated as needed. Changing the length of the attribute requires modification of all attribute values associated with that attribute. This operation (done at restructuring time) may take some time if there are many values for this attribute. If lengths are expanded, the current value is fit into the larger space; if lengths are reduced, the following occurs:

- Character strings are truncated to the new length
- Integers between - 32768 and 32767 remain the same. Positive integers that are larger than 32767 are set 32767. Negative integers that are less than -32768 are set to -32768.
- Floating point numbers will have the number of significant digits reduced from 16 to 6.

Open Mode: Customization

Scope: DA scope or the attribute's owner scope

Example

In the following example, the attribute length for attribute terminal-type is changed to 26 and the edit-value is changed to be a default value of TTY with no editing (the owner scope remains unchanged).

```
>MODIFY ATTRIBUTE terminal-type;

>>LENGTH = 26;

>>EDIT-VALUE = TTY.

>
```

MODIFY DOMAIN

Modifies a domain.

Syntax

```
M[ODIFY] D[OMAIN] domain-name

    [;S[COPE-]O[WNER] = scope-owner-name ]

    [;SEN[SITIVITY] = sensitivity ]

     .
```

Parameters

domain-name Name of the domain to be modified.

scope-owner-name Name of the new owner scope.

sensitivity The new sensitivity of the domain. Valid values are:

private

public

Description

Open Mode: Exclusive-update

Scope: DA scope or the domain's owner scope

Example

The following example changes the owner scope of the domain accounts-payable to the scope manager.

```
>MODIFY DOMAIN accounts-payable;
>>SCOPE-OWNER = manager.
>
```

MODIFY ENTITY

Modifies an entity.

Syntax

```
M[ODIFY] E[NTITY] entity-name
    ;E[NTITY-]T[YPE] = entity-type-name
   [;A[TTRIBUTE-]L[IST]=([ attribute-name1=[ attribute-value1]]
                [, attribute-name2=[attribute-value2 ]]
                        .
                        .
                        .
                [,attribute-nameN =[attribute-valueN ]])]
   [;C[OMMON] = [common-entity-name ]]
   .
```

Parameters

entity-name Name of the entity whose attributes are to be modified.

entity-type-name Name of the type of the entity whose attributes are to be modified.

attribute-name(N) Name of the attribute to be modified.

attribute-value(N) New value for the attribute being modified.

common-entity-name Name of the entity in the common domain whose attributes are to be shared by the local domain entity being modified. You cannot specify this keyword if the dictionary is open in the common domain. If you specify this parameter, then the attribute list can only contain the following attributes:

scope-owner Name of the new owner scope

sensitivity Specifies the access rights to the entity

id-number A user-specified identification number which is never checked for uniqueness by System Dictionary

If you specify attributes other than these in the attribute list, then you must not use this parameter. If you specify COMMON and do not assign a value to it, the existing link to the entity in the common domain is deleted.

Description

System Dictionary automatically assigns values for the following attributes. You cannot modify the values of these attributes.

date-created

date-changed

scope-changed

If the attribute is of type boolean, character, floating, or integer and you specify the name, but do not assign a value, then the value from the attribute's edit values is used. If the attribute does not have any edit values, then the System Dictionary default value for the attribute's data type is used.

boolean : false

character : ASCII blanks

floating : Floating point zero

integer : Binary zero

If the attribute is of type alias and you specify the name, but do not assign a value, the alias value is deleted. If the attribute is of type variable and you specify the name, but do not assign a value, the variable length attribute's text is deleted. If the attribute is of type boolean the only values allowed are true and false (may be abbreviated T and F, respectively).

To modify or add a variable length attribute, you must specify the text within quotes. If the attribute is already associated to the entity, the original text is replaced with this new text. To replace only part of the text, refer to the EDIT ENTITY command.

To create an alias attribute value, specify the alias attribute name and assign it an alias value.

To modify an attribute of type alias or character, you must enclose the value within quotes if it includes any invalid characters (See

"User-Defined Names" in Chapter 3) or is greater than 32 characters in length.

If sensitivity is the attribute to be modified, the valid values are:

private Only the owner scope is allowed access to the entity, unless it assigns access to other scopes by associating the entity to the scope by means of the ADD SCOPE-ENTITY command. This is the default.

read Any scope with read capability may read the entity. The owner scope may, in addition, assign modify access to other scopes by associating the entity to the scope by means of the ADD SCOPE-ENTITY command.

modify Any scope with read capability may read the entity. Any scope with create capability may read and modify the entity.

If you modify the sensitivity from read or modify to private, all scopes that previously had access to this entity will no longer have access, unless the entity is explicitly associated with the scope. If the entity is linked to an entity in the common domain, the sensitivity for this entity cannot be changed to a sensitivity greater than the sensitivity of the entity in the common domain. To link the entity to an entity in the common domain, the current version must be linked to a version in the common domain.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability and modify access to the entity

Example

The following example deletes the alias attribute value for cobol-alias and changes the type of file and minimum and maximum record sizes for the file base01.

```
>MODIFY ENTITY base01;
>>ENTITY-TYPE = file;
>>ATTRIBUTE-LIST = (cobol-alias = ,
>>file-type = sequential,
>>min-record-size = 128,
>>max-record-size = 256).
>
```

MODIFY ENTITY-TYPE

Modifies an entity type.

Syntax

```
M[ODIFY] E[NTITY-]T[YPE] entity-type-name
    [;S[COPE-]O[WNER] = scope-owner-name ]
    .
```

Parameters

entity-type-name Name of the entity type to be modified.

scope-owner-name Name of the new owner scope.

Description

Open Mode: Customization

Scope: DA scope or the entity type's owner scope

Example

The following example changes the owner scope of entity-type terminal to the scope manager.

```
>MODIFY ENTITY-TYPE terminal;
```

```
>>SCOPE-OWNER = manager.
>
```

MODIFY ENTITY-TYPE-ATTRIBUTE

Modifies entity type attribute pairs.

Syntax

```
M[ODIFY] E[NTITY-]T[YPE-]A[TTRIBUTE] entity-type-name
        ;A[TTRIBUTE] = attribute-name1
                  [,attribute-name2 ]
                         .
                         .
                         .
                  [,attribute-nameN ]
       [;S[COPE-]O[WNER] = scope-owner-name ]
        .
```

Parameters

entity-type-name Name of the entity type involved in the pair to be modified.

attribute-name(N) Name of the attribute involved in the pairs to be modified. You can specify any number of attributes. No attribute may appear more than once.

scope-owner-name Name of the new owner scope.

Description

If you specify more than one attribute, the operation continues until the end of the attribute list, even if an error occurs with any previous attribute.

Open Mode: Customization

Scope: DA scope or the entity type attribute pair's owner scope

Example

The following example changes the scope owner of the entity type attribute pairs terminal / terminal-type and terminal / manufacturer to the scope fred.

```
>MODIFY ENTITY-TYPE-ATTRIBUTE terminal;
>>ATTRIBUTE = terminal-type, manufacturer;
>>SCOPE-OWNER = fred.
>
```

MODIFY RELATIONSHIP

Modifies a relationship.

Syntax

```
M[ODIFY] R[ELATIONSHIP] entity-name1
                         ,entity-name2
                        [,entity-name3 ]
                        [,entity-name4 ]
                        [,entity-name5 ]
                        [,entity-name6 ]
     ;R[ELATIONSHIP-]T[YPE] = entity-type-name1
                              ,entity-type-name2
                             [,entity-type-name3 ]
                             [,entity-type-name4 ]
                             [,entity-type-name5 ]
                             [,entity-type-name6 ]
    [;R[ELATIONSHIP-]C[LASS] = relation-class-name]
    [;A[TTRIBUTE-]L[IST]=([ attribute-name1 =[attribute-value1 ]]
                   [,attribute-name2 =[attribute-value2 ]]
                              .
                              .
                              .
                   [,attribute-nameN =[attribute-valueN ]])]
    [;C[OMMON] = [ common-entity-name1
             ,common-entity-name2
            [,common-entity-name3 ]
            [,common-entity-name4 ]
            [,common-entity-name5 ]
            [,common-entity-name6 ]]]
    .
```

Parameters

entity-name(N) Name of the entity involved in the relationship whose attributes are to be modified.

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

attribute-name(N) Name of the attribute to be modified.

attribute-value(N) New value for the attribute being modified.

common-entity- Name of the entity involved in the relationship in *name(N)* the common domain whose attributes are to be shared by the local domain relationship being modified. You cannot specify this keyword if the dictionary is open in the common domain. If you specify this parameter, then the attribute list can only contain the following attributes:

scope-owner Name of the new owner scope

sensitivity Access rights to the relationship

relationship- Logical order of a child entity

position (the second entity in the relationship) relative to all other child entities for the same parent entity of the same relationship type

If you specify attributes other than these in the attribute list, then you must not use this parameter. If you specify common and do not assign a value to it, the existing link to the relationship in the common domain is deleted.

Description

System Dictionary automatically assigns values for the following attributes and you cannot modify their values:

date-created

date-changed

scope-changed

If the attribute is of type boolean, character, floating, or integer and you specify the name, but do not assign a value, the default value from the attribute's edit values is used. If the attribute does not have any edit values, then the System Dictionary default value for the attribute's data type is used.

boolean : false

character : ASCII blanks

floating : Floating point zero

integer : Binary zero

If the attribute is of type alias and you specify the name, but do notassign a value, the alias value is deleted. If the attribute is of type variable and you specify the name, but do not assign a value, the variable length attribute's text is deleted. If the attribute is of type boolean the only values allowed are true and false (may be abbreviated T and F, respectively). To modify or add a variable length attribute, you must enclose the new text within quotes. If the attribute is already associated to the relationship, the original text is replaced with this new text. To replace only part of the text, refer to the EDIT RELATIONSHIP command. To create an alias attribute value, specify the alias attribute name and assign it an alias value.

To modify an attribute of type alias or character, you must enclose the value in quotes if it includes any invalid characters (See "User-Defined Names" in Chapter 3) or is greater than 32 characters in length. If the relationship is assigned an alias, it is associated with the second entity in the relationship. Only the second entity in the relationship may have an alias, regardless of how many entities form the relationship. Therefore, relationship aliases are typically useful only with binary relationships. If sensitivity is the attribute to be modified, the valid values are:

private Only the owner scope is allowed access to the relationship, unless it assigns access to other scopes by associating the relationship to the scope by means of the ADD SCOPE-RELATIONSHIP command. The default is private.

read — Any scope with read capability may read the relationship. The owner scope may, in addition, assign modify access to other scopes by associating the relationship to the scope by means of the ADD SCOPE-RELATIONSHIP command.

modify — Any scope with read capability may read the relationship. Any scope with create capability may read and modify the relationship.

If the sensitivity is modified from read or modify to private, all scopes that previously had access to this relationship will no longer have access, unless the relationship is explicitly associated with the scope. The relationship-position attribute is treated specially by the system. This attribute is used as a sequencing number to order relationships with the same parent entity (i.e. to represent the order of elements in a record). Accordingly, two relationships of a type with the same entity in the first position cannot have the same value for relationship-position. If specified without a value, the system assigns a relationship-position value to place the relationship at the end of the current list (assigns a value of greater than the largest currently assigned to another relationship with the same parent entity). Note that this is different from other default values as the default depends on the values assigned to other relationships and is different for each relationship. As defaulting the relationship-position value can cause the reordering of a relationship list, you should consider the effects before assigning a default value for this attribute. If the relationship is linked to a relationship in the common domain, the sensitivity for this relationship cannot be changed to a sensitivity greater than the sensitivity of the relationship in the common domain. To link the relationship to a relationship in the common domain, the current version must be linked to a version in the common domain.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability and modify access to the relationship

Example

The following example modifies the inventory-control contains item-master relationship of the relationship type image-database contains image-dataset. The blocking-factor was changed to 256 and the capacity was changed to 501.

```
>MODIFY RELATIONSHIP inventory-control, item-master;

>>RELATIONSHIP-TYPE = image-database, image-dataset;

>>RELATIONSHIP-CLASS = contains;

>>ATTRIBUTE-LIST = (blocking-factor = 256, capacity = 501).

>
```

MODIFY RELATIONSHIP-CLASS

Modifies a relationship class.

Syntax

M[ODIFY] R[ELATIONSHIP-]C[LASS] *relation-class-name*

[;S[COPE-]O[WNER] = *scope-owner-name*]

.

Parameters

relation-class-name — Name of the relationship class to be modified.

scope-owner-name Name of the new owner scope.

Description

Open Mode: Customization

Scope: DA scope or the relationship class's owner scope

Example

The following example changes the scope owner of the relationship class includes to the scope manager.

```
>MODIFY RELATIONSHIP-CLASS includes;
>>SCOPE-OWNER = manager.
>
```

MODIFY RELATIONSHIP-TYPE

Modifies a relationship type.

Syntax

```
M[ODIFY] R[ELATIONSHIP-]T[YPE]  entity-type-name1
                              ,entity-type-name2
                             [,entity-type-name3 ]
                             [,entity-type-name4 ]
                             [,entity-type-name5 ]
                             [,entity-type-name6 ]
      [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
      [;S[COPE-]O[WNER] = scope-owner-name ]
       .
```

Parameters

entity-type-name(N) Name of the entity type involved in the relationship type to be modified.

relation-class-name Name of the relationship class.

scope-owner-name Name of the new owner scope.

Description

Open Mode: Customization

Scope: DA scope or the relationship type's owner scope

Example

The following example changes the owner scope of the relationship type project contains program to the scope manager.

```
>MODIFY RELATIONSHIP-TYPE project, program;
```

```
>>RELATIONSHIP-CLASS = contains;
>>SCOPE-OWNER = manager.
>
```

MODIFY RELATIONSHIP-TYPE-ATTRIBUTE

Modifies relationship type attribute pairs.

Syntax

```
M[ODIFY] R[ELATIONSHIP-]T[YPE]-A[TTRIBUTE] entity-type-name1
                                ,entity-type-name2
                               [,entity-type-name3 ]
                               [,entity-type-name4 ]
                               [,entity-type-name5 ]
                               [,entity-type-name6 ]
    [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
     ;A[TTRIBUTE] = attribute-name1
             [,attribute-name2 ]
                   .
                   .
                   .
             [,attribute-nameN ]
    [;S[COPE-]O[WNER] = scope-owner-name ]
     .
```

Parameters

entity-type-name(N) Name of the entity type involved in the relationship type to be modified.

relation-class-name Name of the relationship class.

attribute-name(N) Name of the attribute involved in the pairs to be modified. You can specify any number of attributes. No attribute may appear more than once.

scope-owner-name Name of the new owner scope.

Description

If you specify more than one attribute, the operation continues until the end of the attribute list even if an error occurs with any previous attribute.

Open Mode: Customization

Scope: DA scope or the relationship type attribute pair's owner scope

Example

The following example changes the owner scope of the relationship type

attribute pair program contains module / size to the scope fred.

```
>MODIFY RELATIONSHIP-TYPE-ATTRIBUTE program, module;

>>RELATIONSHIP-CLASS = contains;

>>ATTRIBUTE = size;

>>SCOPE-OWNER = fred.

>
```

MODIFY REPORT

Modifies a report.

Syntax

```
M[ODIFY] REP[ORT]  report-name

      [;DESC[RIPTION] = ["description-text "]]

      [;S[COPE-]O[WNER] = scope-owner-name ]

       .
```

Parameters

report-name Name of the report whose description or owner scope

is to be modified.

description-text Description of the stored report. If you specify

text, you must enclose the text within quotes.

scope-owner-name Name of the new owner scope.

Description

If a description is already associated to the report, the original text

is replaced with this new text. If you do not specify any text, the

description is deleted. To replace only part of the description text,

refer to the EDIT REPORT command. In addition, to change the definition

of a report, use the EDIT REPORT command.

Open Mode: Exclusive-update or shared-update

Scope: DA scope or the report's owner scope

Example

The following example changes the report description for the report sales-report to "Monthly report on the dictionary used by the sales force of the company." .

```
>MODIFY REPORT sales-report;
>>DESCRIPTION = "Monthly report on the dictionary used by
>>             the sales force of the company.".
>
```

MODIFY SCOPE

Modifies a scope.

Syntax

```
M[ODIFY] S[COPE] scope-name
     [;P[ASSWORD] = [password-parameter ]]
     [;N[EW-]P[ASSWORD] = [new-password-parm ]]
     [;S[COPE-]O[WNER] = scope-owner-name ]
     [;S[COPE-]R[IGHTS] = scope-rights1
                   [,scope-rights2 ]
                          .
                          .
                          .
                   [,scope-rightsN ]]
     .
```

Parameters

scope-name Name of the scope to be modified.

password-parameter Current scope password. If you omit this keyword or specify it without a value, a blank password is assumed.

The Dictionary Administrator (DA) scope can change any field for a scope without providing the current password.

new-password-parm New scope password. If you specify the keyword without a value, a blank password is assigned. Any characters are allowed in a password, If a character is desired in the password that is not valid in other System Dictionary names, you must enter the password within quotes to allow recognition of the restricted characters. (See "Scope Password" in Chapter 3.) Remember that the case of the entered password also counts when the password is checked.

scope-owner-name Name of the new owner scope.

scope-rights(N) New capability of the scope. The new capabilities must be sufficient for all items owned by the scope.

`create` The scope can create, modify, and delete entities and relationships, depending on security restrictions. Read capability is automatically assigned.

`domain` The scope can create, modify, and delete domains. Version capability is automatically assigned.

`extend` The scope can create, modify, and delete entity types, relationship types, relationship classes, and attributes to extend, or customize, the dictionary structure.

`read` The scope can read entities and relationships accessible to the scope.

`secure` The scope can create, modify, and delete other scopes.

`version` The scope can create, modify, and delete versions.

The scope rights assigned are not additive. The set you specify in this field become the new scope rights set for the scope.

Description

A scope can change its own password but not its owner scope or scope rights. The current scope cannot modify a scope to have any scope right that it does not have itself (i.e. cannot give extend capability to a scope if the current scope does not have extend capability).

Open Mode: Exclusive-update

Scope: DA scope or the scope's owner scope (the current scope can change its own password)

Example

The following example changes the password of the scope named manager from mgr to perm and changes its scope rights to secure, version, and read.

```
>MODIFY SCOPE manager;

>>PASSWORD = mgr;

>>NEW-PASSWORD = perm;

>>SCOPE-RIGHTS = secure, version, read.

>
```

MODIFY SCOPE-ENTITY

Modifies the access a scope has to an entity.

Syntax

M[ODIFY] S[COPE-]E[NTITY] *scope-name*

;E[NTITY] = *entity-name*

;E[NTITY-]T[YPE] = *entity-type-name*

;S[COPE-]A[CCESS] = *scope-access*

.

Parameters

scope-name Name of the scope with which the entity is associated.

entity-name Name of the entity that is associated with the scope.

entity-type-name Name of the type of the entity that is associated with the scope.

scope-access The new access allowed to the entity. The two types of access that are allowed are:

read	The scope can only read the entity information.
modify	The scope may modify the entity; read access is automatically assigned.

The scope to be given access must have create capability to be allowed modify access.

Description

Open Mode: Shared-update or exclusive-update

Scope: DA scope or the entity's owner scope

Example

The following example changes the access allowed the scope personnel to the document IND-102 to modify access.

```
>MODIFY SCOPE-ENTITY personnel;
>>ENTITY = IND-102;
>>ENTITY-TYPE = document;
>>SCOPE-ACCESS = modify.
>
```

MODIFY SCOPE-RELATIONSHIP

Modifies the access a scope has to a relationship.

Syntax

```
M[ODIFY] S[COPE-]R[ELATIONSHIP] scope-name
     ;R[ELATIONSHIP] = entity-name1
                 ,entity-name2
                [,entity-name3 ]
                [,entity-name4 ]
                [,entity-name5 ]
                [,entity-name6 ]
     ;R[ELATIONSHIP-]T[YPE] = entity-type-name1
                      ,entity-type-name2
                     [,entity-type-name3 ]
                     [,entity-type-name4 ]
                     [,entity-type-name5 ]
                     [,entity-type-name6 ]
    [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
```

```
        ;S[COPE-]A[CCESS] = scope-access
        .
```

Parameters

scope-name Name of the scope with which the relationship is associated.

entity-name(N) Name of the entity involved in the relationship that is associated with the scope.

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

scope-access The new access allowed to the relationship. The two types of access that are allowed are:

read The scope can only read the relationship information.

modify The scope may modify the relationship; read access is automatically assigned.

Description

The scope to be given access must have create capability to be allowed modify access.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or the relationship's owner scope

Example

The following example changes the access allowed the scope candidate to the relationship name contains last-name to modify access.

```
>MODIFY SCOPE-RELATIONSHIP candidate;
>>RELATIONSHIP = name, last-name;
>>RELATIONSHIP-TYPE = element, element;
>>RELATIONSHIP-CLASS = contains;
>>SCOPE-ACCESS = modify.
>
```

MODIFY VERSION

Modifies a version.

Syntax

```
M[ODIFY] V[ERSION] version-name
        [;S[COPE-]O[WNER] = scope-owner-name ]
        [;C[OMMON] = [common-version-name ]]
        .
```

Parameters

version-name Name of the version to be modified.

scope-owner-name Name of the new owner scope.

common-version-name Name of the common domain version to which this local domain version is linked. If specified without a value, the link that currently exists to a version in the common domain will be deleted.

Description

The common link cannot be deleted if any occurrences are linked. Remember that a common link can be deleted only if one currently exists.

Open Mode: Exclusive-update

Scope: DA scope or the version's owner scope

Example

The following example changes the owner scope of the version B_00_08 to the scope sales, and links it to version A_00_01 in the common domain.

```
>MODIFY VERSION B_00_08;

>>SCOPE-OWNER = SALES;

>>COMMON = A_00_01.

>
```

OPTIONS

Sets the prompting, logging and macro options.

Syntax

OP[TIONS]

[;PROM[PT] = *prompt-flag*]

[;C[OMMAND-]LOG = *command-log-file*]

[;I[NTRINSIC-]LOG = *intrinsic-log-file*]

[;LOG = [*log-file-name*]]

[;MAC[RO] = [*macro-file-name*]]

.

Parameters

prompt-flag The attribute prompting option. If prompting is on, you are prompted for attribute values whenever you issue a CREATE, MODIFY, or REPORT command is without the ATTRIBUTE-LIST keyword. Valid choices are:

on

off

The default is on when the system is running interactively. If not running interactively, prompting is off and cannot be turned on using this command. (If you attempt to turn prompting on in batch mode, a warning message is issued.) See "Dictionary Attribute Prompting Facility" in Chapter 3 for a complete discussion of attribute prompting.

command-log-file The command logging option. Commands are logged if command logging is on. Valid choices are:

on

off

The default is on if there is a valid log file when the system is started (and you do not respond with no to the overwrite prompt if running interactively). The command that disables command logging is not logged. However, the command that enables command logging is logged.

intrinsic-log-file The System Dictionary transaction logging option. This logging is in conjunction with IMAGE logging. You must be logged on under the DA scope in exclusive-update mode in order to change logging from its current state. Valid choices are:

on

off

For more information on intrinsic logging, see the SDSetControl intrinsic under Code 1 (intrinsic logging) in the *HP System Dictionary/XL Intrinsics Reference Manual*. Logging is initially disabled by the initialization process (SDINIT). Like IMAGE logging, System Dictionary logging remains disabled until enabled and once enabled, remains enabled until disabled again.

See the *IMAGE Data Base Management System Reference Manual* for more information on IMAGE transaction logging.

log-file-name Name of the new log file. If you enter the LOG keyword with no value, the logging facility is turned off. If you enter a file name, but the file cannot be opened, the previous log file (if any) is retained. If the new file is successfully opened, command logging is activated even if it was turned off before the command, (unless this command disables logging).

macro-file-name Name of the new macro file. If you enter the MACRO keyword with no value, the macro facility is turned off. If you enter a file name, but the file cannot be opened, the previous macro file (if there is one) is retained.

Description

Only the values of the keywords you specify in the command are changed. All other keyword values remain as they were before you issued the command.

If not needed, you can turn command logging off to improve performance.

Open Mode: Any mode for all keywords except INTRINSIC-LOG, which requires exclusive-update

Scope: Any scope for all keywords except INTRINSIC-LOG, which requires the DA scope

Example

The following example sets attribute prompting OFF and specifies the file mymacro to be used as the macro file.

```
>OPTIONS PROMPT = off;
>>MACRO = mymacro.
>
```

REDO

Allows error correction or changes to any of the last 20 commands issued.

Syntax

REDO [*qualifier*] [.]

Parameters

qualifiers

You choose the command to edit using the REDO command with an optional qualifier. The possible qualifiers include:

No qualifier

No qualifier means to edit the previous command. Example: REDO

Absolute number

An absolute number is a positive number between 1 and 20, inclusive, that indicates the command on the redo stack corresponding to the number entered that is to be selected for processing. Example: REDO 4

Relative number

A relative number is a negative number between -20 and -1, inclusive, that indicates the command on the redo stack at the specified offset from the current REDO command that is to be selected for processing. Note that a relative offset of -1 is equivalent to no qualifier, as it specifies the previous command. Example: REDO -5

Character string

A character string is a string of characters that can be accepted as a valid System Dictionary name. The string is used as a pattern (no selection criteria can be used) to search back into the redo history stack to find the first command starting with characters that match the specified string. Note that the case of the letters is important since an exact match is required. The corresponding command is then selected for processing. Example: REDO CREATE.

Quoted string

A quoted string is a string of characters surrounded by quotes. A quoted string works the same as a character string except that the quotes allow the inclusion of characters, including blanks, that are not allowed for character strings. Example: REDO "CREATE ENTITY"

Subcommands

A

Appends one or more characters following the subcommand to the end of the current line, regardless of the position of the subcommand.

B Breaks the line into two lines, moving the character above the B and all following characters to the next line. The second line becomes the current edit line.

D Deletes the character above the D. You may also use a D below both the first and last character to be deleted with either spaces or Ds between thus deleting all characters between the first and last D inclusive.

E Exits the REDO editing mode without executing the edited command. All REDO subcommands issued between the REDO command and the EXIT command are ignored.

H Lists all available editing subcommands in REDO mode. Your current edit line is then redisplayed.

I Inserts one or more characters immediately preceding the character that is above the I. You can join a delete and insert by using D's followed by an I and the characters to be inserted.

L Lists the complete command as it is currently edited and then redisplays the line you are currently editing.

R Replaces the characters above with the new characters you enter. The first character replaced is the one above the R.

X Executes the current command as it has been edited.

+ Places the cursor on the next line of the command you are editing. You can enter + followed by an number n, for the number of lines you want to skip forward. If you do not enter a number, the default is to move forward one line. If the number of lines advanced moves the cursor beyond the end of the command, the cursor is moved to the last line of the command.

Places the cursor on the previous line of the command you are editing. You can enter - followed by n, for the number of lines you want to skip backward. If you do not enter a number, the default is to move backward one line. If the number of lines moved places the cursor before the first line of the command, the cursor is moved to the first line of the command.

[[RETURN]] Places the cursor on the next line of the command you are editing, if the cursor is not currently on the last line of the command. If the cursor is on the last line, the command executes as it has been edited.

Description

You can use the REDO command to edit any of the previous 20 commands you issued. When you issue the REDO command you will enter an editing mode and the first line of the command is displayed for modification. To modify the command, use the previous subcommands. If you enter any character other than a valid subcommand, it and all following characters are interpreted as replacement characters and they replace the characters above them in the command. For example, if you type "TYPE" below a set of characters, the word "TYPE" replaces the four characters in the command line above.

The REDO command is not allowed when executing in a macro, when input is from a file, or when you are executing in batch mode. Because dictionary commands can span more than one line, REDO allows you to edit the command, line by line. You therefore do not have to reenter the entire command when an error is made. Once you issue the REDO command, you are placed in edit mode and the first line of the command is displayed for modification. If you press [[CONTROL]] Y during processing, the REDO command terminates as if you specified the E subcommand.

Open Mode: Any

Scope: Any

Example

The following example corrects the typographical error for the word "entity" (which is misspelled "entiy") and then executes the corrected command.

```
>CREATE ENTITY FIRST-NAME;
>>ENTIY-TYPE = ELEMENT;
>>ATTRIBUTE-LIST = (ELEMENT-TYPE = X, BYTE-LENGTH = 20).
CREATE ENTITY FIRST-NAME;
ENTIY-TYPE = ELEMENT;
     ^
Invalid keyword (SDERR 1242)
            ^
Text skipped from last error to here (SDWARN 1210)
Required keyword ENTITY-TYPE not specified (SDERR 1249)
>REDO
CREATE ENTITY FIRST-NAME;
+
ENTIY-TYPE = ELEMENT;
  IT
ENTITY-TYPE = ELEMENT;
X
>
```

REMOVE ENTITY-TYPE-ATTRIBUTE

Removes attributes from an entity type's attribute list.

Syntax

REM[OVE] E[NTITY-]T[YPE-]A[TTRIBUTE] *entity-type-name*

;A[TTRIBUTE] = *attribute-name1*

[,*attribute-name2*]

.

.

.

[,*attribute-nameN*]

.

Parameters

entity-type-name Name of the entity type from which an attribute is removed.

attribute-name(N) Name of the attribute to be removed.

Description

You can specify any number of attributes. No attribute may appear more than once. If you specify more than one attribute, the operation continues until the end of the attribute list, even if an error occurs with any previous attribute.

You cannot remove any entity type attribute pairs owned by the core set. Remember that the six "automatic" pairs for each entity type (links to attributes *scope-owner, date-created, date-changed, scope-changed, sensitivity,* and *id-number*) are always owned by the core set.

Open Mode: Customization

Scope: DA scope or the entity type attribute pair's owner scope

Example

The following example removes the attributes terminal-type, manufacturer, and location from the entity type terminal.

```
>REMOVE ENTITY-TYPE-ATTRIBUTE terminal;

>>ATTRIBUTE = terminal-type, manufacturer, location.

>
```

REMOVE RELATIONSHIP-TYPE-ATTRIBUTE

Removes attributes from a relationship type's attribute list.

Syntax

```
REM[OVE] R[ELATIONSHIP-]T[YPE-]A[TTRIBUTE]  entity-type-name1

                                      ,entity-type-name2

                                     [,entity-type-name3 ]

                                     [,entity-type-name4 ]

                                     [,entity-type-name5 ]

                                     [,entity-type-name6 ]

     [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]

      ;A[TTRIBUTE] = attribute-name1

               [,attribute-name2 ]

                     .

                     .

                     .
```

```
                [,attribute-nameN ]
        .
```

Parameters

entity-type-name(N) Name of the entity type involved in the relationship type from which attributes are removed.

relation-class-name Name of the relationship class.

attribute-name(N) Name of the attribute to be removed.

Description

You can specify any number of attributes. No attribute may appear more than once. If you specify more than one attribute , the operation continues until the end of the attribute list, even if an error occurs with any previous attribute.

You cannot remove any relationship type attribute pairs owned by the core set. Remember that the six "automatic" pairs for each relationship type (links to attributes *scope-owner*, *date-created*, *date-changed*, *scope-changed*, *sensitivity*, and *relationship-position*) are always owned by the core set.

Open Mode: Customization

Scope: DA scope or the relationship type attribute pair's owner scope

Example

The following example removes the attribute size from the relationship type program contains module.

```
>REMOVE RELATIONSHIP-TYPE-ATTRIBUTE program, module;
>>RELATIONSHIP-CLASS = contains;
>>ATTRIBUTE = size.
>
```

REMOVE SCOPE-DOMAIN

Removes a scope's access to a domain.

Syntax

```
REM[OVE] S[COPE-]D[OMAIN] scope-name
    ;D[OMAIN] = domain-name
    .
```

Parameters

scope-name Name of the scope from which to remove access to the domain.

domain-name Name of the domain from which access by the scope is being removed.

Description

You cannot remove access to a domain from its owner scope.

Open Mode: Shared-update or exclusive-update

Scope: DA scope, the domain's owner scope, or the scope from which access is to be removed

Example

The following example removes access to the domain extra-domain from the scope candidate.

```
>REMOVE SCOPE-DOMAIN candidate;
>>DOMAIN = extra-domain.
>
```

REMOVE SCOPE-ENTITY

Removes a scope's access to an entity.

Syntax

```
REM[OVE] S[COPE-]E[NTITY] scope-name
     ;E[NTITY] = entity-name
     ;E[NTITY-]T[YPE] = entity-type-name
     .
```

Parameters

scope-name Name of the scope from which to remove access to the entity.

entity-name Name of the entity from which access by the scope is being removed.

entity-type-name Name of the type of the entity.

Description

You cannot remove access to an entity from its owner scope.

Open Mode: Shared-update or exclusive-update

Scope: DA scope, the entity's owner scope, or the scope from which access is to be removed

Example

The following example removes access to the file manuals from the scope training.

```
>REMOVE SCOPE-ENTITY training;
>>ENTITY = manuals;
>>ENTITY-TYPE = file.
>
```

REMOVE SCOPE-RELATIONSHIP

Removes a scope's access to a relationship.

Syntax

```
REM[OVE] S[COPE-]R[ELATIONSHIP]  scope-name
        ;R[ELATIONSHIP] = entity-name1
                   ,entity-name2
                  [,entity-name3 ]
                  [,entity-name4 ]
                  [,entity-name5 ]
                  [,entity-name6 ]
        ;R[ELATIONSHIP-]T[YPE] = entity-type-name1
                      ,entity-type-name2
                     [,entity-type-name3 ]
                     [,entity-type-name4 ]
                     [,entity-type-name5 ]
                     [,entity-type-name6 ]
       [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
        .
```

Parameters

scope-name Name of the scope from which to remove access to the relationship.

entity-name(N) Name of the entity involved in the relationship to which access by the scope is being removed.

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

Description

You cannot remove access to a relationship from its owner scope.

Open Mode: Shared-update or exclusive-update

Scope: DA scope, the relationship's owner scope, or the scope from which access is to be removed

Example

The following example removes access to the relationship partno contains 1 of relationship type image-dataset contains image-class from the scope manufacturing.

```
>REMOVE SCOPE-RELATIONSHIP manufacturing;
>>RELATIONSHIP = partno, 1;
>>RELATIONSHIP-TYPE = image-dataset, image-class;
>>RELATIONSHIP-CLASS = contains.
```

```
>
```

RENAME ATTRIBUTE

Renames an attribute.

Syntax

```
REN[AME] A[TTRIBUTE]  old-attribute-name

                 ,new-attribute-name

       .
```

Parameters

old-attribute-name Name of the attribute to be renamed.

new-attribute-name New name of the attribute.

Description

To avoid possible name conflicts with any future extensions to the core set, do not rename any attributes to a name starting with the characters "HP" .

Open Mode: Customization

Scope: DA scope or the attribute's owner scope

Example

The following example renames the attribute length to the new name size.

```
>RENAME ATTRIBUTE length, size.
>
```

RENAME DOMAIN

Renames a domain.

Syntax

```
REN[AME] D[OMAIN]  old-domain-name

              ,new-domain-name

       .
```

Parameters

old-domain-name Name of the domain to be renamed.

new-domain-name New name of the domain.

Description

To avoid possible name conflicts with any future extensions to the core set, do not rename any domains to a name starting with the characters "HP" .

Open Mode: Exclusive-update

Scope: DA scope or the domain's owner scope

Example

The following example renames the domain dicta to the new name dictb.

```
>RENAME DOMAIN dicta, dictb.
>
```

RENAME ENTITY

Renames an entity.

Syntax

```
REN[AME] E[NTITY]  old-entity-name
               ,new-entity-name
      ;E[NTITY-]T[YPE] = entity-type-name
      .
```

Parameters

old-entity-name Name of the entity to be renamed.

new-entity-name New name of the entity.

entity-type-name Name of the type of the entity.

Description

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability and modify access to the entity

Example

The following example renames the element ship-date to the new name purchase-date.

```
>RENAME ENTITY ship-date, purchase-date;
>>ENTITY-TYPE = element.
>
```

RENAME ENTITY-TYPE

Renames an entity type.

Syntax

```
REN[AME] E[NTITY-]T[YPE] old-entity-type-name
                   ,new-entity-type-name
        .
```

Parameters

old-entity-type-name Name of the entity type to be renamed.

*new-entity-type-name*New name of the entity type.

Description

To avoid possible name conflicts with any future extensions to the core set, do not rename any entity types to a name starting with the characters "HP" .

Open Mode: Customization

Scope: DA scope or the entity type's owner scope

Example

The following example renames the entity type equipment to the new name furniture.

```
>RENAME ENTITY-TYPE equipment, furniture.

>
```

RENAME RELATIONSHIP-CLASS

Renames a relationship class.

Syntax

```
REN[AME] R[ELATIONSHIP-]C[LASS]  old-rel-class-name

                               ,new-rel-class-name

           .
```

Parameters

old-rel-class-name Name of the relationship class to be renamed.

new-rel-class-name New name of the relationship class.

Description

To avoid possible name conflicts with any future extensions to the core set, do not rename any relationship class to a name starting with the characters "HP" .

Open Mode: Customization

Scope: DA scope or the relationship class's owner scope

Example

The following example renames the relationship class generates to the new name creates.

```
>RENAME RELATIONSHIP-CLASS generates, creates.

>
```

RENAME REPORT

Renames a report.

Syntax

```
REN[AME] REP[ORT]  old-report-name
                ,new-report-name
        .
```

Parameters

old-report-name Name of the report to be renamed.

new-report-name New name of the report.

Description

Open Mode: Shared-update or exclusive-update

Scope: DA scope or the report's owner scope

Example

The following example renames the report accounting-report to the new name actg-report.

```
>RENAME REPORT accounting-report, actg-report.
>
```

RENAME SCOPE

Renames a scope.

Syntax

```
REN[AME] S[COPE]  old-scope-name
               ,new-scope-name
     [;P[ASSWORD] = [password-parameter ]]
       .
```

Parameters

old-scope-name Name of the scope to be renamed.

new-scope-name New name of the scope.

password-parameter Password of the scope being renamed. The dictionary administrator (DA) scope can rename a scope without providing its password.

Description

To avoid possible name conflicts with any future extensions to the core set, do not rename any scopes to a name starting with the characters "HP".

Open Mode: Exclusive-update

Scope: DA scope or the scope's owner scope

Example

The following example renames the scope marketing to the new name support.

```
>RENAME SCOPE marketing, support;
>>PASSWORD = mgr.
>
```

RENAME VERSION

Renames a version.

Syntax

```
REN[AME] V[ERSION]  old-version-name
                 ,new-version-name
          .
```

Parameters

old-version-name	Name of the version to be renamed.
new-version-name	New name of the version.

Description

Open Mode:	Exclusive-update
Scope:	DA scope or the version's owner scope

Example

The following example renames the version test1 to the new name A_03_01.

```
>RENAME VERSION test1, A_03_01.
>
```

RENUMBER

Renumbers the relationships of a relationship type.

Syntax

```
RENU[*ER] parent-entity-name
     ;R[ELATIONSHIP-]T[YPE] = entity-type-name1
                    ,entity-type-name2
                   [,entity-type-name3 ]
                   [,entity-type-name4 ]
                   [,entity-type-name5 ]
                   [,entity-type-name6 ]
    [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
```

[;S[TART-]POS[ITION] = *start-rel-position*]

[;INC[REMENT] = *position-increment*]

.

Parameters

parent-entity-name Name of the parent entity of the relationships to be renumbered. For example, if the type and class of the relationship to be renumbered is FILE contains RECORD, the *parent-entity-name* is the name of the FILE.

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

start-rel-position The *relationship-position* to be assigned to the first relationship in the list. The default is 1000.

position-increment The amount that the relationship-position attribute is incremented for each subsequent relationship. The default is 1000.

Description

Renumber allows you to assign values with equal intervals between them to the relationship-position attribute of relationships. You can do your own renumbering by modifying the relationship-position attribute value through the MODIFY RELATIONSHIP command. You need to use the RENUMBER command when you want to create a relationship and have it ordered between two existing relationships whose relationship-position attribute values are consecutive.

For example, a COBOL program has the following record layout:

```
01  SALES-DATA.
    05  ACCOUNT          PIC S9(9) COMP.
    05  QUANTITY         PIC S9(4) COMP.
    05  PURCH-DATE         PIC X(6).
```

To define the record layout, the following relationships of the relationship type RECORD contains ELEMENT exist in the dictionary: sales-data contains account with byte-offset of 1 and relationship-position of 1, sales-data contains quantity with byte-offset of 5 and relationship-position of 2, and sales-data contains purch-date with byte-offset of 7 and relationship-position of 3.

The program changes to have the following record layout:

```
01  SALES-DATA.
    05  ACCOUNT          PIC S9(9) COMP.
    05  QUANTITY         PIC S9(4) COMP.
    05  PRICE          PIC S9(9) COMP.
    05  PURCH-DATE         PIC X(6).
```

To redefine the record layout, you need to modify the relationship sales-data contains purch-date to have a byte-offset of 11. You also want to create the relationship sales-data contains price and have it ordered between relationships sales-data contains quantity and sales-data contains purch-date. The relationships would have to be renumbered since sales-data contains price needs its relationship-position value to be between 2 and 3 which is impossible. You can renumber the relationships by specifying a START-POSITION of 10 and an INCREMENT of 10. This assigns relationship-position values of 10, 20, and 30 to the relationships sales-data contains account, sales-data contains quantity, and sales-data contains purch-date, respectively. Now you can create the new relationship sales-data contains price and assign its byte-offset to 7 and its relationship-position to a value between 20 and 30. The relationships now define the new record layout.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability and modify access to all the relationships with the specified parent entity

Example

The following example renumbers all the relationships of relationship type record contains element whose parent entity is sales-data starting with 10 and increasing by increments of 10.

```
>RENUMBER sales-data;

>>RELATIONSHIP-TYPE = record, element;

>>RELATIONSHIP-CLASS = contains;

>>START-POSITION = 10;

>>INCREMENT = 10.

>
```

REPORT ENTITY

Reports on one or more entities.

Syntax

```
REP[ORT] E[NTITY] [entity-name ]

    ;E[NTITY-]T[YPE] = entity-type-name

    [;A[TTRIBUTE-]L[IST]=([ attribute-name1 =[attribute-value1 ]]

                [,attribute-name2 =[attribute-value2 ]]

                          .

                          .

                          .

                [,attribute-nameN=[attribute-valueN ]])]

    [;C[OMMON]= [common-entity-name ]]

    [;ALL            ]

    [;NAME[-ONLY]        ]

    [;L[IST] = information-list]

    [;NO[SORT]]

    [;SUB[-REPORT] = (nested-execute1

          [ [,] nested-execute2 ]
```

.

.

.

[[,] *nested-executeN*])]

Parameters

entity-name Name of the entity to be reported.

entity-type-name Name of the type of the entity to be reported. No selection criteria are allowed.

attribute-name(N) Attribute of the specified entity type. No selection criteria are allowed. For any attribute whose name is not included here, there is no qualification on the value of the attribute. You cannot quality variable length attributes, therefore, you cannot specify them here.

attribute-value(N) Value for the attribute of the entity to be reported. If you omit *attribute-value(N)*, there is no qualification on the value for the corresponding attribute (as if the attribute-name was not specified at all).

common-entity-name If in a local domain, it is the name of the common domain entity linked to the local domain entity to be reported. If in the common domain, it is the name of the local domain entity linked to the entity in the common domain to be reported. Specifying COMMON without a keyword value requests a match to all entities that do not have a common link.

ALL Reports all information about the entity that the scope capabilities and sensitivity allow.

NAME-ONLY Reports only the name of the entities.

information-list A list of the information about the entity that is desired. You can specify each piece of information available in the ALL option individually so you can tailor the information returned. You can specify one or more of the following items:

name Reports the name of the entity (the name is always printed even if this choice is omitted)

attribute(1) Reports the value of the specified attribute that is linked to the entity type. The names are listed if a subset of the attributes is desired.

attribute(N)

attributes Reports the value of all attributes linked to the entity type. If this option is chosen, any specific attribute name that has been listed is ignored.

scope-access Reports the access the current scope has to the entity.

primary Reports the primary name of the entity.

common If in a local domain, common reports the name of the entity in the common domain to which this entity is linked. If in a common domain, common reports all entities in a local domain linked to this entity in the common domain.

synonyms Lists all of the synonyms for the entity.

versions Lists all of the versions having a copy of the entity.

scopes Lists all of the scopes having access to the entity.

relationdata Reports the name and attribute values of the relationship followed for a nested report. This option is chosen in the nested report, and is ignored if specified in the top level of a report.

relationships List all the relationships involving the entity.

The above information is listed if the scope capabilities and the sensitivity of the entity allow the necessary access. If the capabilities or sensitivity do not allow access to some piece of information, then that information is skipped and processing continues with the next piece. This clause with LIST = name, attributes is the default if you do not specify any of the three listing options (ALL, LIST, or NAME-ONLY).

NOSORT Reports on the entities in the unsorted order in which they are retrieved from the dictionary. Otherwise, the entities are sorted alphabetically by the entity name.

nested-execute(N) This parameter allows a nested EXECUTE command that produces a sub-report to the report being generated by this command. (See the syntax for the Nested EXECUTE in this chapter for a more detailed description of this capability.)

Description

Do not specify an *entity-name* if this command is being saved (i.e., is inside a START/SAVE pair). If specified within a START/SAVE pair, the *entity-name* will be ignored.

Remember, you can use selection criteria (see "Selection Criteria" in Chapter 3) in most fields to qualify the entities that are retrieved.

Open Mode: Read-only, shared-update, or exclusive-update

Scope: DA scope or any scope with read access to the entity

Example

The following example produces a report that lists information on the image-database named info-network-division. The information reported was limited to a specific subset including the values of several attributes, the access the current scope has to the entity, and the names of other scopes that have access.

```
>REPORT ENTITY info-network-division;

>>ENTITY-TYPE = image-database;

>>LIST = name, scope-access, sensitivity, id-number, image-database-type

>>scopes, date-created, scope-owner.

INFO-NETWORK-DIVISION

  Scope-Access : OWNER

  DATE-CREATED : 11/11/84  6:25  PM

  SCOPE-OWNER : DA

  SENSITIVITY : PRIVATE

  ID-NUMBER : 0
  IMAGE-DATABASE-TYPE : IMAGE
  Scopes : PERSONNEL, SALES
  1 Entity(ies) Retrieved
>
```

Example

The following example lists the names of all entities of the entity type element that have the name p-no as the image alias.

```
>REPORT ENTITY;
>>ENTITY-TYPE=element;
>>ATTRIBUTE-LIST=(image-alias=p-no);
>>NAME-ONLY.
0 Entity(ies) Retrieved
>
```

REPORT RELATIONSHIP

Reports on one or more relationships.

Syntax

```
REP[ORT] R[ELATIONSHIP] [entity-name1 ]
             [,entity-name2 ]
             [,entity-name3 ]
             [,entity-name4 ]
             [,entity-name5 ]
             [,entity-name6 ]
   ;R[ELATIONSHIP-]T[YPE] = entity-type-name1
                ,entity-type-name2
               [,entity-type-name3 ]
               [,entity-type-name4 ]
               [,entity-type-name5 ]
               [,entity-type-name6 ]
   [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
   [;A[TTRIBUTE-]L[IST]=([ attribute-name1 =[attribute-value1 ]]
               [,attribute-name2 =[attribute-value2 ]]
                      .
                      .
```

```
                    .
              [,attribute-nameN=[attribute-valueN ]])]
      [;C[OMMON] = [ common-entity-name1 ]
                [,common-entity-name2 ]
                [,common-entity-name3 ]
                [,common-entity-name4 ]
                [,common-entity-name5 ]
                [,common-entity-name6 ]]
      [;ALL                 ]
      [;NAME[-ONLY]         ]
      [;L[IST] = information-list]
      .
```

Parameters

entity-name(N) Name of the entity involved in the relationship to be reported.

entity-type-name(N) Name of the entity type involved in the relationship type of the reported relationship. No selection criteria are allowed.

relation-class-name Name of the relationship class. No selection criteria are allowed.

attribute-name(N) Attribute of the specified relationship type. No selection criteria are allowed. For any attribute whose name is not included here, there is no qualification on the value of the attribute. You cannot specify variable length attributes.

attribute-value(N) Value for the attribute of the relationship to be reported. If you omit *attribute-value(N)*, there is no qualification on the value for the corresponding attribute (as if the attribute-name was not specified at all).

common-entity-name(N) If in a local domain, it is the name of the entity involved in the common domain relationship linked to the local domain relationship to be reported. If in the common domain, it is the name of the entity involved in the local domain relationship linked to the common domain relationship to be reported. Specifying COMMON without a value requests a match to all relationships that do not have a common link.

ALL Reports all information about the relationship that the scope capabilities allow.

NAME-ONLY Reports only the name of the relationships.

information-list A list of the information about the relationship that is desired. You can specify each piece of information available in the ALL option individually so you can tailor the information returned. You can specify one or more of the following items:

name Reports the name of the relationship (the name is always printed even if you omit this choice).

attribute(1) Reports the value of the specified attribute that is linked to the relationship type. The names are listed if a subset of the attributes is desired.

attribute(N)

attributes — Reports the values of all attributes linked to the relationship type. If you choose this option, any specific attribute names that have been listed are ignored.

scope-access — Reports the access the current scope has to the relationship.

common — If in a local domain, specifying common reports the name of the relationship in the common domain to which this relationship is linked. If in the common domain, specifying common reports all relationships in a local domain linked to this relationship in the common domain.

versions — Lists all of the versions having a copy of the relationship.

scopes — Lists all of the scopes having access to the relationship.

The above information is listed if the scope capabilities allow the necessary access. If the capabilities do not allow access to some piece of information, then that information is skipped and processing continues with the next piece.

This clause with LIST = name, attributes is the default if you do not specify any of the three listing options (ALL, LIST, or NAME-ONLY).

Description

Do not specify *entity-name(N)* if this command is being saved (i.e., is inside a START/SAVE pair). If specified within a START/SAVE pair, it is ignored. Note that you can use either nothing between commas (,,) or a ^ between commas (,^,) as a placeholder to match any entity name in the middle of a list.

Remember, you can use selection criteria (see "Selection Criteria" in Chapter 3) in most fields to qualify the relationships retrieved.

Open Mode: Read-only, shared-update, or exclusive-update

Scope: DA scope or any scope with read access to the relationship

Example

The following example produces a report on all relationships of type image-database contains image-dataset with the image-database accounts. All information about the relationship is reported: its common link (it does not have one), the access the current scope has to the relationship, the values of all the attributes for the relationship, the scopes having access to the relationship, and the versions containing a copy of the relationship.

```
>REPORT RELATIONSHIP accounts;

>>RELATIONSHIP-TYPE = image-database,image-dataset;

>>RELATIONSHIP-CLASS = contains;

>>ALL.

ACCOUNTS CUSTOMER

  Common :

  Scope-Access : OWNER

  DATE-CREATED : 11/11/84  4:44 AM
```

```
    DATE-CHANGED : 11/11/84  4:44 AM
    SCOPE-OWNER : DA
    SCOPE-CHANGED : DA
    SENSITIVITY : PRIVATE
    RELATIONSHIP-POSITION : 1000
    BLOCKING-FACTOR : 100
    CAPACITY : 10000
    Scopes : SALES, MARKETING
    Versions : V1, V2
    1 Relationship(s) Retrieved
  >
```

RESEQUENCE

Resequences the relationships of a relationship type.

Syntax

```
RESE[QUENCE] entity-name1
         ,entity-name2
        [,entity-name3 ]
        [,entity-name4 ]
        [,entity-name5 ]
        [,entity-name6 ]
         ;R[ELATIONSHIP-]T[YPE] = entity-type-name1
                                 ,entity-type-name2
                                [,entity-type-name3 ]
                                [,entity-type-name4 ]
                                [,entity-type-name5 ]
                                [,entity-type-name6 ]
        [;R[ELATIONSHIP-]C[LASS] = relation-class-name ]
        [;B[EFORE-]E[NTITY] = before-entity-name2
                             [,before-entity-name3 ]
                             [,before-entity-name4 ]
                             [,before-entity-name5 ]
                             [,before-entity-name6 ]]
         .
```

Parameters

entity-name(N) Name of the entity involved in the relationship to be resequenced.

entity-type-name(N) Name of the entity type involved in the relationship type.

relation-class-name Name of the relationship class.

before-entity-name(N) Name of the child entity in the relationship that the relationship to resequence is to be positioned before. The parent entity in the relationships is *entity-name1*. The entity you specify here must be the child entity associated with the parent entity. Therefore, the number of entities you specify in the list must be one less than the number you specified in the entity-name list in the object clause. The numbers correspond to the 2nd, 3rd, ..6the positions in the relationship. If you do not specify this parameter, the relationship to resequence is positioned as the last relationship of the relationship type.

Description

The sequence in which the relationships are numbered, through the relationship-position attribute, is the sequence the relationships are reported when using the reporting commands if one or more entity positions are specified without qualification. When a relationship is resequenced, the relationship-position attribute is modified to have a value between the BEFORE-ENTITY relationship and the relationship preceding the BEFORE-ENTITY relationship. For example, a Pascal program has the following data structure:

```
Employee = RECORD
        Last-Name   : PACKED ARRAY [1..20] OF CHAR;
        First-Name  : PACKED ARRAY [1..20] OF CHAR;
        Empl-Num    : INTEGER;
      END;
```

To define the data structure, the following relationships of relationship type RECORD contains ELEMENT exist in the dictionary: employee contains last-name, employee contains first-name, and employee contains empl-num with relationship-position values of 10, 20, and 30 respectively. The program changes to have the following data structure:

```
Employee = RECORD
        Empl-Num    : INTEGER;
        Last-Name   : PACKED ARRAY [1..20] OF CHAR;
        First-Name  : PACKED ARRAY [1..20] OF CHAR;
      END;
```

To redefine the data structure in the dictionary, you can resequence the relationships by specifying employee contains empl-num as the relationship to resequence and last-name as the BEFORE-ENTITY. This assigns 5 as the relationship-position value of relationship employee contains empl-num. The relationships now define the new structure. If the new relationship-position coincides with an existing one, a collision error occurs and you will either have to renumber the relationships through the RENUMBER command, or modify the relationship-position attribute value through the MODIFY RELATIONSHIP command/subcommand pair.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or a scope with create capability and modify access to all relationships with the specified *entity-name1* as the parent entity

Example

The following example positions the relationship employee contains empl-num of relationship type record contains element before the relationship employee contains last-name.

```
>RESEQUENCE employee, empl-num;

>>RELATIONSHIP-TYPE = record, element;

>>RELATIONSHIP-CLASS = contains;

>>BEFORE-ENTITY = last-name.

>
```

RESTRUCTURE

Restructures the dictionary.

Syntax

```
REST[RUCTURE] [.]
```

Description

RESTRUCTURE allows you to explicitly restructure the dictionary while it is open in customization mode. Restructuring occurs automatically when you open the dictionary in customization mode and any one of the following operations occur:

- An EXIT is issued
- A new dictionary is opened
- The same dictionary is redefined using a different open mode or name mode

The restructure operation may take some time if many dictionary occurrences are affected by the structure changes.

Open Mode: Customization

Scope: Any (however, since scopes without extend capability cannot issue any of the commands to change the structure, the command will have no action for them)

Example

The following example restructures the dictionary that is currently opened in customization mode.

```
>RESTRUCTURE

>
```

SAVE

Saves a report that has just been defined.

Syntax

```
SA[VE] report-name
    [;INT[ERNAL] = internal-name ]
    [;DESC[RIPTION] = "description-text "]
     .
```

Parameters

report-name Name of the report to be saved.

internal-name Internal name of the report being saved. If not specified, the internal name is the same as the *report-name*.

description-text Description of the report being saved. If used, you must specify text within quotes. If omitted, no description is associated with the report.

Description

The report definition must have immediately preceded this command, while the definition must have immediately followed the START command. A report definition consists of an optional CONFIGURE and/ or FORMAT command followed by a required REPORT command.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability

Example

The following example saves the report element-report that produces a report of the names and attributes on entities of type element. In addition, the report on each element begins on a new page.

```
>START
>CONFIGURE PAGE.
>REPORT ENTITY;
>>    ENTITY-TYPE = element.
>SAVE element-report;
>>   DESCRIPTION = "Detailed report to list occurrences" of
>>                 entity-type ELEMENT in sorted order.".
>
```

SETVERSION

Sets a version of entities and relationships to a version status.

Syntax

```
SET[VERSION] version-name
      ;STAT[US] = version-status
      .
```

Parameters

version-name Name of the version whose status is to be set.

version-status The status to which the version is set. Valid statuses are:

test

production

archival

Description

If you set the version that you are currently working in (specified through the DEFINE command), you will then be operating in a version of the new status.

Whenever you create a version, the version is automatically set to test status. You do not need to issue the SETVERSION command after the version is created, if you want the status to remain in test. It is only necessary to issue this command when you want to change the status to either production or archival.

Remember, only one version in a domain can be in production status at any given time.

Open Mode: Exclusive-update

Scope: DA scope or any scope with version capability

Example

The following example sets the version status of version B_000_* to production.

```
>SETVERSION B_000_*;
>>STATUS = production.
>
```

SHOW

Lists all current environment information, including the dictionary environment defined by the DEFINE command, the SDMAIN environment defined by the OPTIONS command, and the output environment defined by the CONFIGURE and FORMAT commands.

Syntax

```
SH[OW] [.]
```

Description

The information that you see when you issue the SHOW command is the information for the dictionary environment, the SDMAIN environment, and the output environment. The dictionary environment includes the name of the dictionary that is open, the scope, the open mode and name mode, and the domain, version, and version status that are used for creating or retrieving definitions. The SDMAIN environment provides the current state of the command log and prompt flags, the log file name, and macro file name. The output environment includes the length and width of the terminal and paper that is being used, the size of the four margins of the paper, the item spacing information, the output destination, the alignment and format-variable specifications, the current page header, and title line.

Open Mode: Any

Scope: Any

Example

The following example shows all current environment information.

```
>SHOW
Dictionary     : SYSDIC.PUB.SYS
Scope          : MANAGER
Open-Mode      : READ-ONLY
Name-Mode      : EXTERNAL
Domain         : DOMAIN1
Version        : VERSION1
Version-Status : PRODUCTION
Log-File       : SDLOG.PUB.SYS
Macro-File     : SDMACRO.PUB.SYS
Command-Log    : OFF          Prompt-Mode    : ON
Screen-Length  : 24           Screen-Width   : 80
Page-Length    : 66           Page-Width     : 85
Left-Margin    : 10           Right-Margin   : 10
Top-Margin     :  6           Bottom-Margin  :  6
Spacing        : DOUBLE-SPACE
Output-File    : PRINTER
Alignment      : UNALIGNED
Format-Variable: OFF
Page-Header    : ANNUAL REPORT
Title          : PURCHASING DEPARTMENT
>
```

SHOWMACRO

Lists the name of all macros currently active in the system.

Syntax

```
S[HOW]M[ACRO] [.]
```

Description

Only the names of valid macros that are available to you are displayed. You can use the SHOWMACRO command to find the name of a macro. You can then use the HELP command list the macro to determine its actions.

Open Mode: Any

Scope: Any

Example

The following example lists all the currently active macros.

```
>SHOWMACRO
OPEN
CREATE-FILE
CREATE-RECORD
CREATE-ELEMENT
CREATE-FILE-RT
CREATE-RECORD-RT
REPORT-FILE
REPORT-ELEMENT
REPORT-RECORD
BYE
>
```

SHOWREDO

Displays a list of all the commands that are currently on the redo history stack.

Syntax

```
S[HOW]R[EDO] [.]
```

Description

The list that SDMAIN displays lists the text of the commands with each marked with a number. You can then use either the numbers to produce a relative or absolute reference to a previous command or the command itself to issue a character string qualification to a previous command.

Example

The following example lists all of the commands that are currently on the redo history stack.

```
>SHOWREDO
1. FORMAT
   TITLE = "Monthly Report".
2. OPTIONS ALIGNMENT = ALIGNED.
3. REPORT ENTITY; ENTITY-TYPE = ELEMENT;
   LIST = ATTRIBUTES, COMMON, SCOPES, VERSIONS.
4. HELP SHOWREDO
5. SHOWREDO
```

```
>
```

START

Marks the start of a report definition.

Syntax

```
ST[ART] [.]
```

Description

The report definition must immediately follow this command and it, in turn, must be immediately followed by a SAVE command. A report definition consists of an optional CONFIGURE and/or FORMAT command followed by a required REPORT command.

No commands, other than the CONFIGURE, FORMAT, and REPORT commands are logged after the START command until you issue the SAVE command. Once you issue the SAVE command, all commands are executed and logged. A [[Control]] Y entered while a START is active terminates the START command, thereby eliminating the need for a SAVE command. When this occurs, you are returned to command processing mode. In addition, as is normal in the processing of [[Control]] Y, all levels of redirected input are exited and input is reset to $STDINX.

Open Mode: Shared-update or exclusive-update

Scope: DA scope or any scope with create capability

Example

The following example saves the report element-report that produces a report on entities of type element. In addition, the report on each element begins on a new page.

```
>START

>CONFIGURE PAGE.

>REPORT ENTITY;

>>     ENTITY-TYPE = element.

>SAVE element-report;

>>    DESCRIPTION = "Detailed report to list occurrences of

>>                 entity-type ELEMENT in sorted order.".

>
```

A Appendix A SDMAIN Error Messages

The following is a complete list of System Dictionary SDMAIN errors listed in order by error number. The list includes at least one possible cause of the error and a recommended action for each cause.

The list is divided into the following groups:

Message Number	ErrorType
1001-1099	File System
1100-1149	Initialization
1150-1174	Driver
1175-1199	Validator
1200-1299	Scanner/Parser/Prompt
1300-1324	Configure
1325-1349	Define
1350-1374	Edit
1375-1399	Format
1400-1424	Help
1425-1449	Redo
1450-1474	Renumber/Resequence
1475-1499	Macro Processor
1500-1599	Create/Copy/Modify
1600-1699	Display/Report
1700-1724	Start/Save
1725-1749	Stored Report
1750-1799	Execute
1800-1899	Miscellaneous

NOTE A number of the error messages list an action of "Refer to the associated error". If this is the suggested action, the system has encountered an error in an MPE subsystem or with an System Dictionary intrinsic call. The subsystem or intrinsic returns a message that is printed for you and helps to explain the error encountered.

Another set of error messages list an action of "Contact the DA". If this is the suggested action, the system has encountered an error with the file system or an MPE subsystem that is unexpected. You should report this problem to the Dictionary Administrator, who should try to determine its cause and fix it, if possible. If assistance is needed, the System Manager should contact an HP Response Center.

Some of the messages, as shown in this list, include an exclamation point (!). This character stands for a parameter (such as a file name), which is displayed as part of the actual message in place of the !.

File System Messages (1001-1099)

1001 MESSAGE **Unable to open the dictionary (SDERR 1001)**

CAUSE The dictionary cannot be opened with the specified DEFINE command.

ACTION Refer to the associated error and correct the indicated problem.

1002 MESSAGE **Cannot reopen dictionary after restructuring (SDERR 1002)**

CAUSE The dictionary cannot be reopened after the completion of the RESTRUCTURE operation.

ACTION Refer to the associated error and correct the indicated problem.

1003 MESSAGE **Dictionary close failed (SDERR 1003)**

CAUSE The dictionary cannot be closed.

ACTION Refer to the associated error and correct the indicated problem.

1005 MESSAGE **Unable to open $STDLIST file (SDERR 1005)**

CAUSE Cannot open the $STDLIST file due to a file error.

ACTION Refer to the associated file error and correct the indicated problem.

1006 MESSAGE **Unable to retrieve information about $STDLIST file (SDERR 1006)**

CAUSE Unable to retrieve information about the $STDLIST file through the FGetInfo intrinsic.

ACTION Contact the DA. (Remember to get the associated error, too.)

1007 MESSAGE **Error while writing the banner message (SDERR 1007)**

CAUSE Error while writing the banner message to $STDLIST.

ACTION Refer to the associated file error and correct the indicated problem.

1008 MESSAGE **Error while writing the command prompt (SDERR 1008)**

CAUSE Error while writing the command prompt to $STDLIST.

ACTION Refer to the associated file error and correct the indicated problem.

1009 MESSAGE **Error while writing a prompt message (SDERR 1009)**

CAUSE Error while writing a prompt message to $STDLIST.

ACTION Refer to the associated file error and correct the indicated problem.

1010 MESSAGE **Error while writing output line (SDERR 1010)**

CAUSE Error while writing an output line to $STDLIST.

ACTION Refer to the associated file error and correct the indicated problem.

1015 MESSAGE **Unable to open $STDINX file (SDERR 1015)**

CAUSE Unable to open the $STDINX file.

ACTION Refer to the associated file error and correct the indicated problem.

1016 MESSAGE **Unable to read from $STDINX file (SDERR 1016)**

CAUSE Unable to read from the $STDINX file.

ACTION Refer to the associated file error and correct the indicated problem.

1017 MESSAGE **Unable to retrieve information about the $STDINX file (SDERR 1017)**

CAUSE Unable to retrieve information about the $STDINX file through the FGetInfo intrinsic.

ACTION Contact the DA. (Remember to get the associated error, too.)

1018 MESSAGE **Unable to close $STDINX file (SDERR 1018)**

CAUSE The $STDINX file cannot be closed.

ACTION Contact the DA. (Remember to get the associated error, too.)

1020 MESSAGE **Unable to open the input file (SDERR 1020)**

CAUSE Cannot open the specified input file.

ACTION Refer to the associated file error and correct the indicated problem.

1021 MESSAGE **Unable to close the input file (SDERR 1021)**

CAUSE Cannot close the input file while exiting the system.

ACTION Refer to the associated file error and correct the indicated problem.

1022 MESSAGE **Error while reading an input line (SDERR 1022)**

CAUSE Bad read when reading the next line from the input file.

ACTION Refer to the associated file error and correct the indicated problem.

1023 MESSAGE **Unable to retrieve information about the input file (SDERR 1023)**

CAUSE Unable to retrieve information about the input file through the FGetInfo intrinsic.

ACTION Contact the DA. (Remember to get the associated error, too.)

1024 MESSAGE **Input file recsize too long. All lines truncated to 80 chars (SDWARN 1024)**

CAUSE The user-specified input file has a record size greater than 80 characters. All input lines are a maximum of 80 characters so all lines are truncated to 80 characters.

ACTION If the commands in the file do not take up more than 80 characters per line, no action is needed. If the commands do take more than 80 characters per line, all characters after 80 are ignored so the expected action will not occur. In this case, the commands must be placed on multiple lines so that no one line contains more than 80 characters (including embedded blanks).

1025 MESSAGE **Input has been reset to your terminal (SDWARN 1025)**

CAUSE A Control-Y was entered while input was initially redirected by using SDIN. All nested input levels are exited and input is reset to $STDINX.

ACTION No action is needed.

1030 MESSAGE **The log file is full (SDWARN 1030)**

CAUSE The log file for the system is full. The system will attempt to open an alternate log file if one is available. If no alternate is available, logging is disabled. In either case, processing can continue.

ACTION No action is needed.

1031 MESSAGE **Unable to close the log file (SDERR 1031)**

CAUSE Cannot close the log file while exiting the system.

ACTION Refer to the associated file error and correct the indicated problem.

1032 MESSAGE **Command logging has been disabled for this session (SDERR 1032)**

CAUSE SDLOG is equated to $NULL or the user responded that the system cannot overwrite the existing log file on entrance to the system.

ACTION To start command logging, use the OPTIONS command to specify a log file.

CAUSE The SDLOG file is filled up or there is a write error to the SDLOG file so it is no longer available for use.

ACTION To continue command logging, use the OPTIONS command to specify a new log file.

1033 MESSAGE **Close failed on old log file (SDWARN 1033)**

CAUSE Cannot close the old log file when specifying a new log file. The new file will be the log file.

ACTION Refer to the associated error and correct the indicated problem.

1035 MESSAGE **Unable to open macro file SDMACRO (SDERR 1035)**

CAUSE The SDMACRO file cannot be opened.

ACTION No action is needed if you do not wish to use the macro facility. If you wish to use the facility, specify a valid macro file name using the OPTIONS command.

1036 MESSAGE **Error while reading from SDMACRO file (SDERR 1036)**

CAUSE An error was encountered while reading from the SDMACRO file.

ACTION Refer to the associated file error and correct the indicated problem.

1037 MESSAGE **Close failed on old Macro file (SDWARN 1037)**

CAUSE Could not close the old macro file when specifying a new macro file. The new file will be the macro file.

ACTION Refer to the associated file error and correct the indicated problem.

1038 MESSAGE **Macro file recsize too long. All lines truncated to 80 chars (SDWARN 1038)**

CAUSE The user specified macro file has a record size of larger than 80 characters. All macro lines are a maximum of 80 characters so all lines are truncated to 80 characters.

ACTION If the commands in the file do not take up more than 80 characters per line, no action is needed. If the commands do take more than 80 characters per line, all characters after 80 are ignored so the expected action will not occur. In this case, the commands must be placed on multiple lines so that no one line contains more than 80 characters (including embedded blanks).

1040 MESSAGE **Default the output file to $STDLIST (SDWARN 1040)**

CAUSE Cannot open the specified output file. The system defaulted to opening $STDLIST as the output file so processing can continue.

ACTION Refer to the associated file error, correct the indicated problem, and use the CONFIGURE command to redirect to the output file. Or, do nothing and continue processing with the output directed to $STDLIST.

1041 MESSAGE **Output file is full. Must reassign before additional output (SDERR 1041)**

CAUSE The output file is full so no more output can be directed into it. If more reporting output is desired, the file must be redirected before continuing.

ACTION Use the CONFIGURE command to redirect the output file to another file or to $STDLIST as desired.

1042 MESSAGE **Unable to close the output file (SDERR 1042)**

CAUSE Cannot close the output file while exiting the system.

ACTION Refer to the associated file error and correct the indicated problem.

1043 MESSAGE **Old Output file close failed. Opening new Output file (SDERR 1043)**

CAUSE Cannot close the old output file when redirecting output to a new file. The new file will be the output file.

ACTION Refer to the associated file error and correct the indicated problem.

1044 MESSAGE **Unable to open printer file. Default Output to $STDLIST (SDERR 1044)**

CAUSE Cannot open the printer as requested. Output is directed to $STDLIST.

ACTION Refer to the associated file error, correct the indicated problem, and use the CONFIGURE command again to direct output to the printer. Or, do nothing and leave the output directed to $STDLIST.

1050 MESSAGE **Cannot reset EDTXT file equate. Reset manually if necessary (SDERR 1050)**

CAUSE Cannot reset the EDTXT file equate while exiting the system.

ACTION Issue the following MPE command: RESET EDTXT.

1051 MESSAGE **EDTXT file is full (SDERR 1051)**

CAUSE The text of the specified variable attribute does not fit in the edit file.

ACTION Contact the DA. (Remember to get the associated error, too.)

1052 MESSAGE **Error while writing to file ! (SDERR 1052)**

CAUSE Error encountered while writing to the edit file.

ACTION Contact the DA. (Remember to get the associated error, too.)

1053 MESSAGE **Unable to reopen EDTXT file (SDERR 1053)**

CAUSE The edit file cannot be opened by the system after the return from the editor. All the edited text is lost but the original text is preserved.

ACTION Contact the DA. (Remember to get the associated error, too.)

1054 MESSAGE **Cannot write eof to ! file (SDERR 1054)**

CAUSE Error encountered while trying to write an end-of-file mark to the edit file.

ACTION Contact the DA. (Remember to get the associated error, too.)

1055 MESSAGE **! close failed so no edit was made (SDERR 1055)**

CAUSE Error encountered while closing the edit file in preparation for passing it to the editor.

ACTION Contact the DA. (Remember to get the associated error, too.)

1056 MESSAGE **! close failed (SDERR 1056)**

CAUSE Cannot close the edit file while exiting the system.

ACTION Contact the DA. (Remember to get the associated error, too.)

1057 MESSAGE **Edited text is too long. Truncated to maximum allowed (SDWARN 1057)**

CAUSE The edited text is greater than the maximum length allowed by SDMAIN for variable attributes. The maximum length is allowed and any extra is truncated.

ACTION No action is needed. However, the text should be modified to compact the text so that the portion truncated can again be added onto the end.

1060 MESSAGE **Unable to open the include file (SDERR 1060)**

CAUSE Cannot open the specified include file.

ACTION Refer to the associated file error and correct the indicated problem.

1061 MESSAGE **Unable to retrieve information about the include file (SDERR 1061)**

CAUSE Unable to retrieve information about the include file through the FGetInfo intrinsic.

ACTION Contact the DA. (Remember to get the associated error, too.)

1062 MESSAGE **Include file recsize too long. All lines truncated to 80 chars (SDWARN 1062)**

CAUSE The user specified input file has a record size of larger than 80 characters. All input lines are a maximum of 80 characters so all lines are truncated to 80 characters.

ACTION If the commands in the file do not take up more than 80 characters per line, no action is needed. If the commands do take more than 80 characters per line, all characters after 80 are ignored so the expected action will not occur. In this case, the commands must be placed on multiple lines so than no one line contains more than 80 characters (including embedded blanks).

1090 MESSAGE **Unable to open temporary file ! (SDERR 1090)**

CAUSE Cannot open a temporary file for the system to use.

ACTION Contact the DA. (Remember to get the associated error, too.)

1091 MESSAGE **Error while rewinding temporary file ! (SDERR 1091)**

CAUSE Error while resetting the file pointer of a temporary file to the beginning of the file.

ACTION Contact the DA. (Remember to get the associated error, too.)

1092 MESSAGE **Error while reading from temporary file ! (SDERR 1092)**

CAUSE Error while reading from the temporary file.

ACTION Contact the DA. (Remember to get the associated error, too.)

1093 MESSAGE **Error while retrieving information on temporary file ! (SDERR 1093)**

CAUSE Unable to retrieve information about the temporary file through the FGetInfo intrinsic.

ACTION Contact the DA. (Remember to get the associated error, too.)

1094 MESSAGE **Temporary file is full ! (SDERR 1094)**

CAUSE A temporary file used by the system is full.

ACTION The number of items requested by the command is too large (> 32767). Use Selection Criteria to reduce the number of items retrieved or to split the retrieval so that half are retrieved in one command and the other half in another command.

1095 MESSAGE **Error while writing to temporary file ! (SDERR 1095)**

CAUSE Error while writing to a temporary file.

ACTION Contact the DA. (Remember to get the associated error, too.)

1096 MESSAGE **Error while writing End-Of-File to temporary file ! (SDERR 1096)**

CAUSE Error while writing an end-of-file marker to a temporary file.

ACTION Contact the DA. (Remember to get the associated error, too.)

1097 MESSAGE **Purge failed on temporary file ! (SDERR 1097)**

CAUSE Error while trying to delete a temporary file that is no longer needed.

ACTION Contact the DA. (Remember to get the associated error, too.)

Initialization Messages (1100-1149)

1100 MESSAGE **Unable to find the Banner message in the catalog (SDERR 1100)**

CAUSE Cannot retrieve the banner message from the message catalog.

ACTION Contact the DA to get a new copy of the catalog with the banner message in it.

1101 MESSAGE **Word overflow, too many characters in message number ! (SDERR 1101)**

CAUSE The number at the specified location in the message catalog is too large.

ACTION Contact the DA to get a new copy of the catalog with a legal number in the specified location.

1102 MESSAGE **An illegal character was encountered in message number ! (SDERR 1102)**

CAUSE The number at the specified location in the message catalog has an illegal character in it.

ACTION Contact the DA to get a new copy of the catalog with a legal number in the specified location.

1103 MESSAGE **Illegal negative number found in message number ! (SDERR 1103)**

CAUSE The number at the specified location in the message catalog is negative.

ACTION Contact the DA to get a new copy of the catalog with a legal number in the specified location.

1104 MESSAGE **Invalid language number specified in the catalog (SDERR 1104)**

CAUSE The language number specified in the catalog is invalid.

ACTION Contact the DA to get a new copy of the catalog with a valid language number.

1105 MESSAGE **Error while converting a number in message set ! (SDERR 1105)**

CAUSE Error encountered while converting a number in the specified set in the message catalog.

ACTION Refer to the associated error message for the message number of the bad number and the type of error encountered. Then, contact the DA to get a new copy of the catalog with the error corrected.

1106 MESSAGE **Abbreviation for command word ! is missing (SDERR 1106)**

CAUSE The abbreviation for the specified command word in the command word set of the message catalog is missing.

ACTION Contact the DA to get a new copy of the catalog with the missing abbreviation present.

1107 MESSAGE **Command word with abbreviation ! is missing (SDERR 1107)**

CAUSE The command word with the specified abbreviation in the command word set of the message catalog is missing.

ACTION Contact the DA to get a new copy of the catalog with the missing command word present.

1108 MESSAGE **Wild card character is missing (SDERR 1108)**

CAUSE Wild card character is missing from the message catalog.

ACTION Contact the DA to get a new copy of the catalog with the missing wild card character present.

1109 MESSAGE **SDMAIN and Dictionary Intrinsics versions are not compatible (SDERR 1109)**

CAUSE The version of the SDMAIN program and the System Dictionary Intrinsics is not compatible.

ACTION Contact the DA and get a version of the SDMAIN program that is compatible with the version of the intrinsics that are on the system.

Driver Messages (1150-1174)

1150 MESSAGE **Exceeded maximum number of allowed errors (SDWARN 1150)**

CAUSE The system encountered the number of errors specified as the maximum in the PARM option.

ACTION Correct the errors found and rerun/finish running the job.

1151 MESSAGE **Logging has been disabled (SDWARN 1151)**

CAUSE Logging has been disabled because the user does not want the system to overwrite an existing log file.

ACTION Do nothing and no logging will take place or use the OPTIONS command to specify a new log file.

CAUSE Logging has been disabled because the current log file has filled up and an alternate log file was not available.

ACTION Do nothing and no logging will take place or use the OPTIONS command to specify a new log file.

1152 MESSAGE **Logging proceeding on temporary file SDLOG (SDWARN 1152)**

CAUSE Logging is being redirected to an alternate temporary file (SDLOG) after the permanent file (SDLOG) filled up.

ACTION Do nothing as logging will continue in a temporary file with the same name as the permanent file being logged to previously.

1153 MESSAGE **SDLOG is a temporary file. Save if needed (SDWARN 1153)**

CAUSE The log file was saved as a temporary file.

ACTION If the log file is not needed, do nothing and it will remain as a temporary file. If the log file is needed, save it (and rename it if its name is SDLOG).

1154 MESSAGE **Expected End-Of-File NOT found in input file (SDERR 1154)**

CAUSE Unexpected input lines found in the input file after an EXIT command.

ACTION Edit the input file to remove the extra lines.

1155 MESSAGE **Hit End-Of-File of input file before executed an EXIT (SDERR 1155)**

CAUSE The end of the input file was found without finding an EXIT command.

ACTION Edit the input file to place an EXIT command as the last command in the command stream.

1156 MESSAGE **Logging will continue to the previous log file (SDWARN 1156)**

CAUSE There was an error while trying to open the new log file. Logging will continue to the same file as before this command was issued.

ACTION Refer to the associated file error and correct the indicated problem.

Validator Messages (1175-1199)

1175 MESSAGE **Invalid command (SDERR 1175)**

CAUSE The command entered is not recognized by the system.

ACTION Issue a command that is recognized by the system (use HELP for help).

1176 MESSAGE **Invalid command within a START/SAVE pair (SDERR 1176)**

CAUSE The command entered is not allowed while within a START/SAVE pair.

ACTION Issue a command that is allowed within the pair (use HELP for help) or wait until the pair is completed before entering the command.

1177 MESSAGE **Invalid subcommand (SDERR 1177)**

CAUSE The subcommand entered is not recognized by the system.

ACTION Issue a subcommand that is recognized by the system (use HELP command for help).

1178 MESSAGE **Invalid command/subcommand pair (SDERR 1178)**

CAUSE The command and subcommand entered are not legal together.

ACTION Issue a command/subcommand pair that is allowed by the system (use HELP command for help).

1179 MESSAGE **Command/subcommand pair is NOT valid in this open mode (SDERR 1179)**

CAUSE The command/subcommand pair entered is not allowed in the current open mode.

ACTION Issue a command/subcommand pair that is allowed in the current open mode or switch to the appropriate open mode for the specified command/subcommand pair.

1180 MESSAGE **Command is NOT valid in this open mode (SDERR 1180)**

CAUSE The command entered is not allowed in the current open mode.

ACTION Issue a command that is allowed in the current open mode or switch to the appropriate open mode for the specified command.

1181 MESSAGE **Command is NOT allowed when processing a compiled dictionary (SDERR 1181)**

CAUSE The command entered is not allowed when the currently opened dictionary is a compiled dictionary. The command is requesting a modification and a compiled dictionary cannot be modified.

ACTION Issue a command that is allowed in a compiled dictionary (any READ command is allowed) or switch to an appropriate master dictionary to make the desired modification.

Scanner/Parser/Prompt Messages (1200-1299)

1200 MESSAGE **Missing closing quote on a character string (SDERR 1200)**

CAUSE There is no closing quote on the character string specified.

ACTION Insert a closing quote in the appropriate location.

1201 MESSAGE **Exceeded maximum number of temporary files (SDERR 1201)**

CAUSE The command involves too many temporary files (probably in the form of too many variable length attributes specified in a single command).

ACTION Simplify the command so that it does less work (divide the operation into a two-step operation).

1202 MESSAGE **Invalid character found (SDERR 1202)**

CAUSE A character not allowed in the local language was found.

ACTION Remove the indicated character from the command string.

1203 MESSAGE **Name is too long (SDERR 1203)**

CAUSE The indicated name is greater than 32 characters.

ACTION Change the name so that it contains no more than 32 characters.

CAUSE The indicated character value is more than 32 characters long.

ACTION Enclose the character string in quotes.

1204 MESSAGE **Invalid character for an MPE file name (SDERR 1204)**

CAUSE The indicated character is not allowed in an MPE filename.

ACTION Remove the invalid character from the filename.

1210 MESSAGE **Text skipped from last error to here (SDWARN 1210)**

CAUSE Note indicating that some of the input string was not processed in order to recover from the preceding error.

ACTION No action is needed.

1211 MESSAGE **Unexpected token for a Value (SDERR 1211)**

CAUSE An attribute or keyword value was expected in the indicated location.

ACTION Insert the missing value or change the existing unexpected token into a valid value.

1212 MESSAGE **Operators are NOT allowed here (SDERR 1212)**

CAUSE A relational operator is not allowed for this value (it cannot be qualified).

ACTION Remove the operator from the command.

1213 MESSAGE **Wild card characters are NOT allowed here (SDERR 1213)**

CAUSE Wild card characters are not allowed for this value (it cannot be qualified).

ACTION Remove the wild card character from the value.

1214 MESSAGE **Unexpected token: probably missing semicolon, comma, or quotes (SDERR 1214)**

CAUSE The token indicated is not allowed in this position.

ACTION Review the command syntax and correct the entered command accordingly.

CAUSE A semicolon is expected before the indicated token to separate the object and a keyword clause or to separate two keyword clauses.

ACTION Insert a semicolon before the indicated token.

CAUSE A comma is expected before the indicated token to separate the attribute value clause from the next attribute value clause.

ACTION Insert a comma before the indicated token.

CAUSE The indicated character string contains characters not allowed in a System Dictionary name and so must be placed within a pair of quotes.

ACTION Place a pair of quotes around the indicated character string.

CAUSE There is a token missing at the indicated position.

ACTION Review the command syntax and insert the part of the command expected.

1215 MESSAGE **Expected an Identifier (SDERR 1215)**

CAUSE An identifier was expected at this location.

ACTION Insert an identifier or change the value in this position to an identifier.

1216 MESSAGE **Too few values in the list (SDERR 1216)**

CAUSE A relationship or relationship type must consist of at least 2 entities or entity types, respectively.

ACTION Include a list of at least 2 in the indicated position.

1217 MESSAGE **Too many values in the list (SDERR 1217)**

CAUSE A relationship or relationship type consists of a maximum of 6 entities or entity types, respectively.

ACTION Reduce the number of items in the list to no more than 6.

1218 MESSAGE **Object clause expected (SDERR 1218)**

CAUSE An object clause indicating the item on which to operate was expected here.

ACTION Include an object on which to operate.

1219 MESSAGE **Characters found after the end of the command (SDERR 1219)**

CAUSE Characters were found after the period ending the command.

ACTION Remove the extra characters that follow the period from the command.

1220 MESSAGE **A period was expected here (SDERR 1220)**

CAUSE No period was found to end the command.

ACTION Place a period at the end of the command.

1221 MESSAGE **Invalid nested command (SDERR 1221)**

CAUSE The nested command is not a valid command.

ACTION Enter a valid command as the nested command (EXECUTE).

CAUSE The nested command is not allowed as a nested command.

ACTION Enter a valid command as the nested command (EXECUTE).

1222 MESSAGE **Invalid attribute name (SDERR 1222)**

CAUSE The indicated string is not a valid attribute name.

ACTION Specify a valid attribute name.

1223 MESSAGE **Equal expected (SDERR 1223)**

CAUSE An equal sign is expected at the indicated location.

ACTION Place an equal sign in the command string as indicated.

1224 MESSAGE **Missing close parenthesis (SDERR 1224)**

CAUSE The closing parenthesis on the complex value is not present.

ACTION Place a closing parenthesis where specified.

CAUSE The closing parenthesis in the qualifying value is missing.

ACTION Place a closing parenthesis where specified.

1225 MESSAGE **Bad character(s) after close parenthesis (SDERR 1225)**

CAUSE The characters following the closing parenthesis are not any of the expected characters.

ACTION Check the command syntax and remove the unexpected characters or insert the missing expected characters.

1226 MESSAGE **Expected a quoted character string (SDERR 1226)**

CAUSE A quoted character string was expected in this position.

ACTION Supply a string within quotes.

CAUSE The value specified here is not quoted.

ACTION Place the string in quotes.

1227 MESSAGE **Expected a filename (SDERR 1227)**

CAUSE An MPE filename was expected here.

ACTION Modify the value in the indicated location to produce a valid filename.

1228 MESSAGE **Invalid domain access (SDERR 1228)**

CAUSE The specified domain access is not one of the valid domain accesses.

ACTION Check the valid domain accesses in the documentation and enter a valid one.

1229 MESSAGE **Invalid list item (SDERR 1229)**

CAUSE The item in the LIST list is not legal.

ACTION Remove the illegal item or change it to a legal list item.

1230 MESSAGE **Invalid option flag (SDERR 1230)**

CAUSE The specified option flag is not one of the valid option flags.

ACTION Check the valid option flags in the documentation and enter a valid one.

1231 MESSAGE **Invalid name mode (SDERR 1231)**

CAUSE The specified name mode is not one of the valid name modes.

ACTION Check the valid name modes in the documentation and enter a valid one.

1232 MESSAGE **Invalid open mode (SDERR 1232)**

CAUSE The specified open mode is not one of the valid open modes.

ACTION Check the valid open modes in the documentation and enter a valid one.

1233 MESSAGE **Invalid output destination (SDERR 1233)**

CAUSE The specified output destination is not one of the valid destinations.

ACTION Check the valid output destinations in the documentation and enter a valid one.

1234 MESSAGE **Invalid scope rights (SDERR 1234)**

CAUSE The specified scope right is not one of the valid scope rights.

ACTION Check the valid scope rights in the documentation and enter a valid one.

1235 MESSAGE **Invalid scope access (SDERR 1235)**

CAUSE The specified scope access is not one of the valid scope accesses.

ACTION Check the valid scope accesses in the documentation and enter a valid one.

1236 MESSAGE **Invalid version status (SDERR 1236)**

CAUSE The specified version status is not one of the valid version statuses.

ACTION Check the valid version statuses in the documentation and enter a valid one.

1237 MESSAGE **Invalid attribute data type (SDERR 1237)**

CAUSE The specified attribute data type is not one of the valid data types.

ACTION Check the valid data types in the documentation and enter a valid one.

1238 MESSAGE **The number is out of range (SDERR 1238)**

CAUSE The specified number is not in the legal range for the field.

ACTION Check the legal range for the field and specify a number in that range.

1239 MESSAGE **The number is too large (maximum is 32767) (SDERR 1239)**

CAUSE The specified number is too large to be represented internally. The largest number allowed in this position is 32767.

ACTION Enter a number that can be represented by the system.

1240 MESSAGE **A number is required (SDERR 1240)**

CAUSE A number value is required in the indicated location.

ACTION Change whatever value is in this position to an appropriate number.

1241 MESSAGE **Quoted string not allowed here (SDERR 1241)**

CAUSE A quoted character string is not allowed in this location.

ACTION Specify a value for the indicating position that is not a quoted string.

1242 MESSAGE **Invalid keyword (SDERR 1242)**

CAUSE The specified keyword is not one of the recognized keywords in the system.

ACTION Change the keyword to one recognized by the system (use HELP command subcommand for help).

1243 MESSAGE **This keyword is NOT valid for this command/subcommand pair (SDERR 1243)**

CAUSE The specified keyword is not allowed with the current command(/subcommand).

ACTION Change the keyword to one that is valid with the current command/subcommand (use HELP command subcommand for help).

1244 MESSAGE **This keyword had already been specified in the command (SDERR 1244)**

CAUSE The indicated keyword has already been specified in the command.

ACTION Combine the two clauses into one clause for the command or eliminate the extra clause.

1245 MESSAGE **There are illegal characters after a keyword flag (SDERR 1245)**

CAUSE The specified keyword is a flag and has unexpected characters following it.

ACTION Remove the unexpected characters after the flag.

1246 MESSAGE **A value is expected (SDERR 1246)**

CAUSE A value is expected in the indicated location.

ACTION Enter an appropriate value where indicated.

1247 MESSAGE **Unexpected left parenthesis (SDERR 1247)**

CAUSE Found an unexpected left parenthesis where Selection Criteria is not allowed.

ACTION Remove the parenthesis and any Selection Criteria as they are not allowed in this location.

CAUSE The indicated keyword clause does not have a complex value and so cannot be within parenthesis.

ACTION Remove the parenthesis and issue an appropriate value clause for the keyword.

1248 MESSAGE **Expected a left parenthesis (SDERR 1248)**

CAUSE Expected an opening left parenthesis for the complex value to be entered.

ACTION Insert parenthesis around the value for this keyword clause.

1249 MESSAGE **Required keyword ! not specified (SDERR 1249)**

CAUSE The supplied keyword is required for this command but was not included.

ACTION Insert the required keyword clause.

1250 MESSAGE **The number specified here must be positive (SDERR 1250)**

CAUSE The number in the specified location must be positive.

ACTION Change the indicated number from a negative to an appropriate positive number.

1252 MESSAGE **No keywords are allowed for this command/subcommand pair (SDERR 1252)**

CAUSE The specified command/subcommand does not have any keywords.

ACTION Remove the keyword clause from the command and execute it.

1253 MESSAGE **= and <> are the only operators allowed (SDERR 1253)**

CAUSE Only = and <> operators are allowed when dealing with scope rights (other relational operators are not meaningful as the rights are not hierarchical).

ACTION Change the Selection Criteria so that only = and <> operations are used.

1254 MESSAGE **There is a duplicate scope right entry (SDERR 1254)**

CAUSE The indicated scope right is specified more than one time.

ACTION Remove the duplicate specification of the scope right.

1255 MESSAGE **Expected a Nested Execute command (SDERR 1255)**

CAUSE A nested EXECUTE command was expected in the SUB-REPORT clause.

ACTION Either specify a nested EXECUTE or remove the SUB-REPORT clause.

1256 MESSAGE **Expected an 'attribute = value' pair (SDERR 1256)**

CAUSE Expected an attribute to be assigned a value (value can be blank) here.

ACTION Check the syntax of the ATTRIBUTE-LIST clause. Probably need to remove an extra comma before the closing parenthesis of the clause.

1257 MESSAGE **Expected an attribute edit value (SDERR 1257)**

CAUSE Expected an edit value for the attribute.

ACTION If there is an extra comma with no value, remove it. Otherwise, make sure that the attribute edit values specified are legal for the attribute type (remember, quotes are required if a character string will not be accepted as a legal System Dictionary name).

1258 MESSAGE **Ranges are legal only for numeric edit values (SDERR 1258)**

CAUSE Ranges of values in the EDIT-VALUE list are allowed only for numeric data types (integer and floating).

ACTION If the attribute is of a numeric type, specify only numeric values and the range can remain. Otherwise, a range is not allowed for this type so the possibilities must be listed explicitly.

1259 MESSAGE **A number range can consist of only two values (SDERR 1259)**

CAUSE A range consists of just two numeric values (val1 : val2). There is a range specified with more than two values (i.e. val1 : val2 : val3).

ACTION Collapse the values into a range of two values or use several values and/or ranges separated by commas to build the edit value list for the attribute.

1260 MESSAGE **Invalid macro parameter name (SDERR 1260)**

CAUSE One or more wild card characters are specified in the parameter name. They are not allowed as they confuse the substitution process.

ACTION Change the name of the indicated parameter to remove all wild card characters.

CAUSE The indicated parameter name in the macro header is not a valid name for a parameter.

ACTION The indicated parameter name should be corrected to make it into a name acceptable to System Dictionary.

1261 MESSAGE **Expected a macro parameter (SDERR 1261)**

CAUSE Expected a parameter to be specified in the macro header at this point.

ACTION There is probably an extra comma in the macro header with no parameter specified.

1275 MESSAGE **Illegal menu option found at position ! (SDERR 1275)**

CAUSE There is an unexpected character at the indicated position.

ACTION Enter a valid option (a number).

1276 MESSAGE **! is not in the valid range for menu options (SDERR 1276)**

CAUSE The indicated number is not one of the possible numbers on the menu.

ACTION Look at the menu and choose a valid option from it.

1277 MESSAGE **Invalid range specified at position ! (SDERR 1277)**

CAUSE The end of the range is the same as the start of the range.

ACTION No range is needed as it is a single number. So, just enter the single number and do not specify a range.

CAUSE The end of the range is less than the start of the range.

ACTION Specify a higher ending point than starting point for the range.

1278 MESSAGE **Invalid use of colon or comma at position ! (SDERR 1278)**

CAUSE The indicated comma or colon is used incorrectly.

ACTION Do nothing as it is ignored. The next time the option string is entered, remove the extra comma/colon.

1279 MESSAGE **Prompting can be used only when input is from $STDINX (SDERR 1279)**

CAUSE Prompting for attributes makes sense only when input is interactive (from $STDINX).

ACTION Do nothing as no action is taken and prompting is not allowed.

1280 MESSAGE **Invalid alignment option (SDERR 1280)**

CAUSE The specified alignment option is not one of the valid alignment options.

ACTION Check the valid alignment options in the documentation and enter a valid one.

Configure Messages (1300-1324)

1300 MESSAGE **Too few columns are specified for the terminal (SDERR 1300)**

CAUSE The number specified for the SCREEN-WIDTH keyword is too small.

ACTION Specify a number for the SCREEN-WIDTH that is 53 or larger. This size is required because the width must be able to handle a 32-character name, a 15-character header and some spacing.

1301 MESSAGE **Too many columns are specified for the terminal (SDERR 1301)**

CAUSE The number specified for the SCREEN-WIDTH keyword is too large.

ACTION Specify a number for the SCREEN-WIDTH that is not larger than the maximum output size (132).

1302 MESSAGE **Too few lines are specified for the terminal (SDERR 1302)**

CAUSE The number specified for the SCREEN-LENGTH keyword is too small.

ACTION Specify a number for the SCREEN-LENGTH that is 1 or larger.

1303 MESSAGE **Too few columns are specified for a page of paper (SDERR 1303)**

CAUSE The number specified for the PAGE-WIDTH keyword is too small.

ACTION Specify a number for the PAGE-WIDTH that is 53 or larger. This size is required because the width must be able to handle a 32 character name, a 15-character header and some spacing.

1304 MESSAGE **Too many columns are specified for a page of paper (SDERR 1304)**

CAUSE The number specified for the PAGE-WIDTH keyword is too large.

ACTION Specify a number for the PAGE-WIDTH that is not larger than the maximum output size (132).

1305 MESSAGE **Too few lines specified for a page of paper (SDERR 1305)**

CAUSE The number specified for the PAGE-LENGTH keyword is too small.

ACTION Specify a number for the PAGE-LENGTH that is 1 or larger.

1306 MESSAGE **Configure not executed due to error (SDWARN 1306)**

CAUSE There was an error in one or more of the CONFIGURE values so none of the values were changed.

ACTION Refer to the associated error and correct the indicated problem.

1307 MESSAGE **SINGLE-SPACE used instead of PAGE (SDWARN 1307)**

CAUSE The SINGLE-SPACE keyword is recognized and the PAGE keyword ignored.

ACTION No action is needed. The next time the command is issued, specify only one spacing option.

1308 MESSAGE **SINGLE-SPACE used instead of DOUBLE-SPACE (SDWARN 1308)**

CAUSE The SINGLE-SPACE keyword is used and the DOUBLE-SPACE keyword ignored.

ACTION No action is needed. The next time the command is issued, specify only one spacing option.

1309 MESSAGE **DOUBLE-SPACE used instead of PAGE (SDWARN 1309)**

CAUSE The DOUBLE-SPACE keyword is used and the PAGE keyword ignored.

ACTION No action is needed. The next time the command is issued, specify only one spacing option.

1310 MESSAGE **Default margins and report page-length are incompatible (SDWARN 1310)**

CAUSE The output file and page-length fields cannot be reassigned while in a nested report (you do not want output sent to different files or have the page length changing within a file while in the same report). When the values for these fields are reset to their default values, they are not compatible with the page defined by the top level report.

ACTION No action is required as the margins and page-length of the top level report are used. The incompatibility should be checked to make sure the various levels of the report are all set up to write to the same size paper.

Define Messages (1325-1349)

1325 MESSAGE **The ! keyword is ignored in customization mode (SDWARN 1325)**

CAUSE The indicated keyword is not allowed if the open mode of the dictionary is going to be CUSTOMIZATION.

ACTION Do nothing as the clause is ignored. The next time the dictionary is open in CUSTOMIZATION mode, do not specify this keyword.

1326 MESSAGE **Both VERSION and STATUS were specified. STATUS is ignored (SDWARN 1326)**

CAUSE Both the VERSION and STATUS keyword cannot be specified in the same command. Either VERSION for a specific version or STATUS for the last version of the given status should be used.

ACTION Do nothing as the version will be used and the status ignored. The next time the dictionary is opened, specify either a specific version (VERSION) or a specific status (STATUS) but not both.

1327 MESSAGE **Quoted passwords cannot extend to multiple lines (SDERR 1327)**

CAUSE If the password is entered within quotes it must be entered on a single line.

ACTION Specify the password so that the opening and closing quotes are on the same line.

1328 MESSAGE **Invalid scope password. Remember case counts (SDWARN 1328)**

CAUSE An invalid password has been entered in response to the scope password prompt. Remember, System Dictionary distinguishes between uppercase and lowercase characters in a scope password.

ACTION Specify the correct password for the scope.

1329 MESSAGE **Exceeded maximum tries for a valid scope password (SDERR 1329)**

CAUSE Unable to open the dictionary because a valid scope password was not specified within three tries.

ACTION Reenter the DEFINE command and specify the correct password for the scope.

1330 MESSAGE **Unexpected End-Of-File on $STDINX (SDERR 1330)**

CAUSE Reached the end of the file on $STDINX when trying to read the scope password.

ACTION Refer to the associated file error and correct the indicated problem.

1331 MESSAGE **Unable to turn the echo facility off (SDERR 1331)**

CAUSE Unable to turn the echo facility off through the FControl Intrinsic when prompting for the scope password.

ACTION Contact the DA. (Remember to get the associated file error, too.)

1332 MESSAGE **Unable to turn the echo facility on (SDERR 1332)**

CAUSE Unable to turn the echo facility on through the FControl Intrinsic when prompting for the scope password.

ACTION Contact the DA. (Remember to get the associated file error, too.)

Edit Messages (1350-1374)

1350 MESSAGE **EDIT only allowed when input is from $STDINX (SDERR 1350)**

CAUSE The EDIT command is only allowed when input is interactive (or when running interactively and input is from a redirected STDIN).

ACTION Use the MODIFY command in the command file or batch job to 'edit' an existing variable attribute or report.

1351 MESSAGE **No EDTXT file is available so the edit fails (SDERR 1351)**

CAUSE No edit file could be opened so the EDIT command has been disabled.

ACTION The system tried to open files EDTXT0 through EDTXT999 and failed. Either there is insufficient disc space for this file or files exist with all of these names. Rerun the system with the above problem corrected. For now, the same results can be obtained by using the appropriate MODIFY command and entering the complete text.

1352 MESSAGE **Cannot delete the variable length attribute (SDERR 1352)**

CAUSE The old variable attribute could not be deleted.

ACTION Contact the DA. Internal Error.

1353 MESSAGE **Cannot create the variable length attribute (SDERR 1353)**

CAUSE The variable length attribute could not be created with the new text.

ACTION Refer to the associated error.

1354 MESSAGE **Cannot invoke EDIT/V (SDERR 1354)**

CAUSE The EDIT/V program could not be invoked from the program.

ACTION Make sure that the EDIT/V program is on the system. If it is, contact the DA. Internal Error.

1355 MESSAGE **The report definition does not exist (SDERR 1355)**

CAUSE There is no definition stored for the indicated report.

ACTION Contact the DA. Internal Error. The definition was deleted manually without using the SDMAIN commands that manipulate reports or the dictionary has been corrupted.

1356 MESSAGE **The report description does not exist (SDERR 1356)**

CAUSE There is no description stored for the specified report.

ACTION Use the MODIFY REPORT command to add the new text as the description for the report.

1357 MESSAGE **The entity was deleted while being edited (SDERR 1357)**

CAUSE The entity was deleted by another user while its attribute was being edited.

ACTION Do nothing if the entity is not needed. If it is needed, find out who deleted it and why. To avoid this problem, open the dictionary in EXCLUSIVE-UPDATE mode.

1358 MESSAGE **The relationship was deleted while being edited (SDERR 1358)**

CAUSE The relationship was deleted by another user while its attribute was being edited.

ACTION Do nothing if the relationship is not needed. If it is needed, find out who deleted it and why. To avoid this problem, open the dictionary in EXCLUSIVE-UPDATE mode.

1359 MESSAGE **The report was deleted while being edited (SDERR 1359)**

CAUSE The report was deleted by another user while its definition or description was being edited.

ACTION Do nothing if the report is not needed. If it is needed, find out who deleted it and why. To avoid this problem, open the dictionary in EXCLUSIVE-UPDATE mode.

1360 MESSAGE **Cannot delete report ! (SDERR 1360)**

CAUSE All lines of the report definition were deleted during editing but the report cannot be deleted from the system.

ACTION Contact the DA. Internal Error.

Format Messages (1375-1399)

1375 MESSAGE **Value for ! is too long (SDWARN 1375)**

CAUSE The value specified for the PAGE-HEADER or TITLE keyword is longer than the maximum output width (132 characters) of the system.

ACTION Reduce the length of the value to no more than the maximum output width. Probably want to reduce the length to less than or equal to the current output width or the line will be truncated when printed.

Help Messages (1400-1424)

1400 MESSAGE **Expected a period (SDERR 1400)**

CAUSE A period was expected in the indicated position if any character is present.

ACTION Remove the unexpected character and replace them by a period or blanks.

1401 MESSAGE **Expected a command or a period (SDERR 1401)**

CAUSE The characters after the HELP command must be either a command or a period.

ACTION Remove the unexpected characters and replace them by a command, a period, or blanks.

1402 MESSAGE **Command does not have a subcommand (SDERR 1402)**

CAUSE The specified command does not have a subcommand.

ACTION Remove the subcommand from the command line. May want to check to see if the desired command is specified correctly.

1403 MESSAGE **Expected a subcommand or a period (SDERR 1403)**

CAUSE The characters after the HELP command must be either a subcommand or a period.

ACTION Remove the unexpected characters and replace them by a subcommand, a period, or blanks.

1404 MESSAGE **Macros do not have subcommands (SDERR 1404)**

CAUSE Macros are identified by the name and cannot be qualified in the HELP command.

ACTION Reenter the command specifying only HELP and the macro name.

1405 MESSAGE **Help not available for this macro definition (SDWARN 1405)**

CAUSE The NOHELP option is specified in the definition for the requested macro so the HELP command cannot be used to display its definition.

ACTION No action is needed. If the command definition should be accessible, edit the macro file to remove the NOHELP option from the macro header.

Redo Messages (1425-1449)

1425 MESSAGE**Cannot REDO a REDO command (SDERR 1425)**

CAUSE REDO cannot be used to change a command into a REDO command.

ACTION Do not change a command being edited to a REDO command.

1426 MESSAGE **REDO and SHOWREDO only allowed when input is from $STDINX (SDERR 1426)**

CAUSE The REDO command and SHOWREDO command are only allowed when input is interactive (or when running interactively and input is from a redirected STDIN).

ACTION Enter the entire command in the command file or batch job instead of trying to use REDO or SHOWREDO.

1427 MESSAGE **No command has been entered to REDO (SDERR 1427)**

CAUSE No command has been entered into the SDMAIN from the current input file yet.

ACTION Enter a command other than REDO. Then use REDO to reissue or correct the command entered.

1428 MESSAGE **The number is too large (SDERR 1428)**

CAUSE The number entered causes an integer overflow.

ACTION Reissue the +/- command with a smaller number.

1429 MESSAGE **There are illegal characters in the number (SDERR 1429)**

CAUSE There are unexpected characters in the number.

ACTION Reissue the +/- command with all non-digit characters removed.

1430 MESSAGE **Must break the line before insert/append to prevent line overflow (SDERR 1430)**

CAUSE Together, the existing line and the string to be inserted or appended will not fit on a single input line.

ACTION Break the existing line so that the new line and the string to be inserted or appended will fit on a single input line.

1431 MESSAGE **Bad command after delete (ignored) (SDERR 1431)**

CAUSE Unexpected characters were on the line after the DELETE command.

ACTION Delete can consist of a continuous string of one or more D' s or two single D' s, with blanks between. The only other command allowed on the same line as a Delete is an Insert (I). It (I) is allowed after either of the two D strings.

1432 MESSAGE **The entire command has been deleted (SDWARN 1432)**

CAUSE The entire command in the REDO buffer has been deleted.

ACTION Since the entire command was deleted, the REDO system returns to the command mode. Edit the original command or issue a new command.

1433 MESSAGE **Unexpected characters after REDO command (SDERR 1433)**

CAUSE There are characters on the line after the REDO system command. Except for the D/I combination, only a single edit command can be entered on a line.

ACTION Reissue the edit command with only a single command on the line.

1434 MESSAGE **+ and - NOT allowed in a number (SDERR 1434)**

CAUSE The numbers used to move within REDO buffer cannot involve a + or -.

ACTION Remove the + or - from the number specifying how far to move and simply use the appropriate command to specify direction (+ to move forward and - to move backward).

1435 MESSAGE **The quoted string after REDO must contain at least one char (SDERR 1435)**

CAUSE The string specified after the REDO command contains no characters (or only blanks). A qualifying value must contain at least one non-blank character.

ACTION Reenter the REDO command specifying a qualifying string to be used to determine the appropriate command to select from the redo history stack.

1436 MESSAGE **No match found for the requested string in the redo stack (SDERR 1436)**

CAUSE No match was found for the specified string in the redo history stack.

ACTION Use the SHOWREDO command to obtain a list of the commands currently in the redo history stack. If the desired command is present, select it by number (either absolute or relative) or provide a qualifying string that matches the start of the command.

1437 MESSAGE **No command is currently stored in the referenced location (SDERR 1437)**

CAUSE No command is stored in the redo stack at the location referred to by the absolute or relative number offset specified.

ACTION Use the SHOWREDO command to obtain a list of the commands currently in the redo history stack. Specify an absolute or relative command number that will refer to one of the commands listed by the command or issue a command that performs the desired action.

Redo Messages (1425-1449)

1425 MESSAGE **Cannot REDO a REDO command (SDERR 1425)**

CAUSE REDO cannot be used to change a command into a REDO command.

ACTION Do not change a command being edited to a REDO command.

1426 MESSAGE **REDO and SHOWREDO only allowed when input is from $STDINX (SDERR 1426)**

CAUSE The REDO command and SHOWREDO command are only allowed when input is interactive (or when running interactively and input is from a redirected STDIN).

ACTION Enter the entire command in the command file or batch job instead of trying to use REDO or SHOWREDO.

1427 MESSAGE **No command has been entered to REDO (SDERR 1427)**

CAUSE No command has been entered into the SDMAIN from the current input file yet.

ACTION Enter a command other than REDO. Then use REDO to reissue or correct the command entered.

1428 MESSAGE **The number is too large (SDERR 1428)**

CAUSE The number entered causes an integer overflow.

ACTION Reissue the +/- command with a smaller number.

1429 MESSAGE **There are illegal characters in the number (SDERR 1429)**

CAUSE There are unexpected characters in the number.

ACTION Reissue the +/- command with all non-digit characters removed.

1430 MESSAGE **Must break the line before insert/append to prevent line overflow (SDERR 1430)**

CAUSE Together, the existing line and the string to be inserted or appended will not fit on a single input line.

ACTION Break the existing line so that the new line and the string to be inserted or appended will fit on a single input line.

1431 MESSAGE **Bad command after delete (ignored) (SDERR 1431)**

CAUSE Unexpected characters were on the line after the DELETE command.

ACTION Delete can consist of a continuous string of one or more D' s or two single D' s, with blanks between. The only other command allowed on the same line as a Delete is an Insert (I). It (I) is allowed after either of the two D strings.

1432 MESSAGE **The entire command has been deleted (SDWARN 1432)**

CAUSE The entire command in the REDO buffer has been deleted.

ACTION Since the entire command was deleted, the REDO system returns to the command mode. Edit the original command or issue a new command.

1433 MESSAGE **Unexpected characters after REDO command (SDERR 1433)**

CAUSE There are characters on the line after the REDO system command. Except for the D/I combination, only a single edit command can be entered on a line.

ACTION Reissue the edit command with only a single command on the line.

1434 MESSAGE **+ and - NOT allowed in a number (SDERR 1434)**

CAUSE The numbers used to move within REDO buffer cannot involve a + or -.

ACTION Remove the + or - from the number specifying how far to move and simply use the appropriate command to specify direction (+ to move forward and - to move backward).

1435 MESSAGE **The quoted string after REDO must contain at least one char (SDERR 1435)**

CAUSE The string specified after the REDO command contains no characters (or only blanks). A qualifying value must contain at least one non-blank character.

ACTION Reenter the REDO command specifying a qualifying string to be used to determine the appropriate command to select from the redo history stack.

1436 MESSAGE **No match found for the requested string in the redo stack (SDERR 1436)**

CAUSE No match was found for the specified string in the redo history stack.

ACTION Use the SHOWREDO command to obtain a list of the commands currently in the redo history stack. If the desired command is present, select it by number (either absolute or relative) or provide a qualifying string that matches the start of the command.

1437 MESSAGE **No command is currently stored in the referenced location (SDERR 1437)**

CAUSE No command is stored in the redo stack at the location referred to by the absolute or relative number offset specified.

ACTION Use the SHOWREDO command to obtain a list of the commands currently in the redo history stack. Specify an absolute or relative command number that will refer to one of the commands listed by the command or issue a command that performs the desired action.

Renumber/Resequence Messages (1450-1474)

1450 MESSAGE **Cannot resequence the relationship due to a collision (SDERR 1450)**

CAUSE The relationship cannot be resequenced as there is no relationship-position number available at the destination location (either the relationship must go before a relationship with relationship-position of 1, after a relationship with relationship-position of the maximum value, or between two relationships numbered consecutively).

ACTION Use the RENUMBER command to renumber the relationships of the type with an INCREMENT of at least 2. Then issue the command again.

1451 MESSAGE **Error while modifying the relationship-position (SDERR 1451)**

CAUSE An error was encountered while the relationship-position was being changed.

ACTION Contact the DA. Internal error.

1452 MESSAGE **Error in retrieving the relationship (SDERR 1452)**

CAUSE Cannot retrieve the indicated relationship to resequence.

ACTION Refer to the associated error and correct the problem.

1453 MESSAGE **Error in retrieving the before relationship (SDERR 1453)**

CAUSE Cannot retrieve the indicated relationship to resequence before.

ACTION Refer to the associated error and correct the problem.

1454 MESSAGE **Error in retrieving the relationship-position (SDERR 1454)**

CAUSE An error was encountered while retrieving the relationship-position.

ACTION Contact the DA. Internal Error.

1455 MESSAGE **BEFORE-ENTITY and relationship to be moved must be different (SDERR 1455)**

CAUSE The relationship to be resequenced was requested to be moved before itself.

ACTION The command does nothing so can be skipped.

1456 MESSAGE **Cannot resequence as only one relationship of the type exists (SDWARN 1456)**

CAUSE A resequence was requested when there is only one relationship of the type.

ACTION The command does nothing so can be skipped.

1457 MESSAGE **Unexpected error. RENUMBER is only partially completed (SDERR 1457)**

CAUSE An error was encountered while updating the relationship-positions. The sequence of the relationships remains the same but their relationship-position may be changed.

ACTION Contact the DA. Internal Error.

1458 MESSAGE **Combination of START-POSITION and INCREMENT leads to overflow (SDERR 1458)**

CAUSE With the given START-POSITION and INCREMENT and the number of relationships of the type, the relationship-position field will overflow by the end of the list.

ACTION Modify the START-POSITION and/or INCREMENT so that they will not lead to overflow during the processing of the list.

1459 MESSAGE **There are no relationships to renumber (SDERR 1459)**

CAUSE There are no relationships of the type to renumber.

ACTION The command does nothing so can be skipped.

1460 MESSAGE **Current scope needs modify access to all relationships of the type (SDERR 1460)**

CAUSE The current scope does not have modify access to all of the required subset of relationships of the specified relationship type. The required subset consists of all relationships having the given entity in the first position. In order to RENUMBER or RESEQUENCE a relationship, the user must have modify access to all the relationships in this subset.

ACTION Open the dictionary as either the DA scope or as a scope with modify access to all the relationships of the specified relationship type.

1461 MESSAGE **BEFORE-ENTITY has incorrect number of entities for the rel-type (SDERR 1461)**

CAUSE Too few entities were specified in the BEFORE-ENTITY entity list when the command was entered. The entity in the clause must consist of a list of the entities having one less entity than the relationship type has, as there is no entity corresponding to the parent entity.

ACTION Reenter the command, specifying a valid child entity (entity list) for the relationship type of the relationship being resequenced.

Macro Processor Messages (1475-1499)

1475 MESSAGE **Restoring macros from previous SDMACRO file (SDWARN 1475)**

CAUSE Due to the noted error encountered during loading of the newly specified macro file, the macros from the previous macro file will be restored.

ACTION No action is needed. The indicated error must be corrected to allow the specified file to be used as a macro file in the future.

1476 MESSAGE **Invalid name for a macro at line ! in SDMACRO file (SDERR 1476)**

CAUSE The name specified for the macro definition beginning at the indicated line is not a valid name.

ACTION The indicated macro is invalid at this time. The macro file should be edited to correct the invalid name so the macro can be used in future runs of the program.

1477 MESSAGE **No name specified for macro at line ! in SDMACRO file (SDERR 1477)**

CAUSE There was no name specified for the macro definition beginning at the indicated line.

ACTION The indicated macro is invalid at this time. The macro file should be edited to correct the invalid name so the macro can be used in future runs of the program. (There is probably just an extra blank line in the file that should be removed.)

1478 MESSAGE **Errors in header for macro at line ! in SDMACRO file (SDERR 1478)**

CAUSE One or more errors were detected in the header for the macro beginning on the indicated line.

ACTION See the associated error messages for an explanation of the errors found. The indicated macro is invalid at this time. The macro file should be edited to correct the invalid header so the macro can be used in future runs of the program.

1479 MESSAGE **The macro call has more parameters than the definition (SDERR 1479)**

CAUSE The macro call specifies more parameters than are specified in the definition in the macro file.

ACTION Use the HELP macro-name command to find the definition of the macro and correct the call to correspond to the number of parameters defined in the macro. If HELP shows that the macro definition is not what you wanted, the next time you exit the system, edit the macro file to adjust the definition as needed and the macro will be ready for all future runs.

1480 MESSAGE **The value ! could not be substituted into the definition (SDERR 1480)**

CAUSE The indicated value could not be substituted into the macro definition as the line it is being substituted on cannot be split.

ACTION Edit the macro definition to have some kind of punctuation (blank, period, comma, etc.) on the affected line. This allows the system to break the line at a consistent point and perform the substitution.

1481 MESSAGE **No macro file has been specified (SDERR 1481)**

CAUSE A SHOWMACRO command was issued but no macro file has been initialized. So, there are no macro names to report.

ACTION Use the OPTIONS command to load a macro file and then issue the SHOWMACRO to see what macro names were loaded. No action is required if you do not wish to use any macros.

1482 MESSAGE **Macro name ! conflicts with a command name (SDERR 1482)**

CAUSE The indicated macro name conflicts with an SDMAIN command name.

ACTION Change the name of the macro to one that does not conflict with an SDMAIN command name. The indicated macro will not be available for the current execution.

1483 MESSAGE **No value specified for required parameter ! (SDERR 1483)**

CAUSE The specified parameter is a required parameter but no value is provided on the call.

ACTION Provide a value for the specified parameter. If the parameter really should be optional, you can make it optional by editing the macro header to provide a default value of a null value (that is, parm=). This allows you to omit a value for the parameter if desired and the system will substitute nothing in its place.

1484 MESSAGE **Invalid macro option (SDERR 1484)**

CAUSE The specified macro option is not one of the valid macro options.

ACTION Check the valid macro options in the documentation and enter a valid one. The indicated macro will not be available for the current execution.

1485 MESSAGE **Conflicting macro option. Default value ! used (SDWARN 1485)**

CAUSE The indicated macro option conflicts with a previous macro option.

ACTION No action is required at this time. The macro will be defined for the current execution using the specified default value for the conflicting options. You should edit the macro file to remove the conflicting options for future uses of the macro file.

1486 MESSAGE **Must supply a value for the required parameter (SDWARN 1486)**

CAUSE The parameter being prompted for is a required parameter. A value is required for all required parameters.

ACTION The system reprompts for the parameter. To execute the macro, enter an appropriate value for this parameter. To cancel the macro, press [[Control]] Y. If the parameter is required, no further action is needed; however, if the parameter is optional or has a default value that can be used if no new value is entered, you should edit the macro header to provide the desired default value before the program is run the next time.

1487 MESSAGE **Expected a simple value or a quoted string for parameter value (SDERR 1487)**

CAUSE The value entered for the last prompt contains a character not allowed in a valid name or contains more than 32 characters. The value you enter for a macro parameter must be a name, number, or quoted text string.

ACTION If the value is supposed to be a simple value, correct the error and reenter the value at the prompt. If the value contains an illegal character, such as a blank, then reenter the value at the prompt specifying the value within a pair of quotes.

Create/Copy/Modify Messages (1500-1599)

1500 MESSAGE **The text for Variable attribute ! was too long (truncated) (SDWARN 1500)**

CAUSE The text for the indicated variable length attribute is greater than the maximum allowed by the system. The maximum length allowed is accepted and the remainder truncated.

ACTION No action is needed as the maximum length is stored with any excess truncated.

1501 MESSAGE **The text for fixed attribute ! was too long (truncated) (SDWARN 1501)**

CAUSE The text for the fixed length character string is longer than the attribute length. The maximum length is stored and the remainder truncated.

ACTION No action is needed as the attribute length number of characters are stored with any excess truncated.

1502 MESSAGE **Exceeded max attr per type. Attributes after ! ignored (SDERR 1502)**

CAUSE More attributes are specified than are allowed to be associated to a type.

ACTION Check to make sure that all the attributes specified are linked to the type or are alias or variable attributes. If the check is OK, split the command into two pieces and assign values to all of the fixed attributes in one half and the alias and variable attributes in the other.

1503 MESSAGE **Exceeded buffer maximum. Attributes after ! ignored (SDERR 1503)**

CAUSE The buffer holding the attribute values overflowed.

ACTION Break the attributes into two pieces and use two commands to perform the operation desired.

1504 MESSAGE **Attribute ! does not exist (SDERR 1504)**

CAUSE The indicated attribute does not exist.

ACTION Create the attribute before proceeding or specify an attribute that does exist.

1505 MESSAGE **Couldn't restore variable attributes (SDERR 1505)**

CAUSE The variable attributes cannot be restored.

ACTION Contact the DA. Internal Error.

1506 MESSAGE **Unable to lock System Dictionary (SDERR 1506)**

CAUSE The dictionary lock failed.

ACTION Contact the DA. Internal Error.

1507 MESSAGE **System Dictionary Integrity may be lost (SDERR 1507)**

CAUSE Dictionary operations performing a portion of the operations indicated by the command completed. The remainder did not completed due to an error and the operations that completed cannot be reversed.

ACTION Contact the DA. Internal Error.

1508 MESSAGE **Cannot retrieve source version ! (SDERR 1508)**

CAUSE The source version does not exist in the current domain.

ACTION Refer to the associated error and correct the indicated problem.

1509 MESSAGE **Cannot create the target version (SDERR 1509)**

CAUSE The target version could not be created.

ACTION Refer to the associated error and correct the indicated problem.

1510 MESSAGE **Cannot get Relationship-Type attribute list (SDERR 1510)**

CAUSE Error encountered while retrieving the attribute list for a relationship type.

ACTION Contact the DA. Internal Error.

1511 MESSAGE **Cannot retrieve relationships of the source entity (SDERR 1511)**

CAUSE Error encountered while retrieving a relationship involving the source entity.

ACTION Contact the DA. Internal Error.

1512 MESSAGE **Cannot create a relationship (SDERR 1512)**

CAUSE Cannot create one of the target relationships.

ACTION Refer to the associated error and correct the indicated problem.

1513 MESSAGE **Cannot get Entity-Type attribute list (SDERR 1513)**

CAUSE Error encountered while retrieving the attribute list for the entity type.

ACTION Contact the DA. Internal Error.

1514 MESSAGE **Cannot retrieve the source entity (SDERR 1514)**

CAUSE Error encountered while retrieving the source entity.

ACTION Refer to the associated error and correct the indicated problem.

1515 MESSAGE **Cannot create the target entity (SDERR 1515)**

CAUSE Error encountered while creating the target entity.

ACTION Refer to the associated error and correct the indicated problem.

1516 MESSAGE **Cannot retrieve needed attribute(s) (SDERR 1516)**

CAUSE Error encountered while retrieving an attribute needed for command execution.

ACTION Contact the DA. Internal Error.

1517 MESSAGE **Cannot retrieve a variable length attribute value (SDERR 1517)**

CAUSE Error encountered while retrieving a variable length attribute value.

ACTION Contact the DA. Internal Error.

1518 MESSAGE **Cannot create a variable length attribute value (SDERR 1518)**

CAUSE Error encountered while creating a new variable attribute.

ACTION Refer to the associated error and correct the indicated problem.

1519 MESSAGE **Position specified is too high for the relationship (SDERR 1519)**

CAUSE POSITION specified by the user is larger than the number of entities involved in the relationship.

ACTION Change the POSITION value to one that is between 1 and the number of entities involved in the relationship (inclusive).

1520 MESSAGE **Cannot retrieve the source report (SDERR 1520)**

CAUSE Error encountered while retrieving the source report.

ACTION Refer to the associated error and correct the indicated problem.

1521 MESSAGE **Cannot create the target report (SDERR 1521)**

CAUSE Error encountered while creating the target report.

ACTION Refer to the associated error and correct the indicated problem.

1522 MESSAGE **No attributes were added (SDERR 1522)**

CAUSE No attributes are added to the entity type/relationship type.

ACTION Refer to the associated error and correct the indicated problem.

1523 MESSAGE **Attribute ! was not added (SDERR 1523)**

CAUSE The indicated attribute is not added to the entity type/relationship type.

ACTION Refer to the associated error and correct the indicated problem.

1524 MESSAGE **No attribute pairs were modified (SDERR 1524)**

CAUSE No modifications are made to any of the entity type/relationship type pairs.

ACTION Refer to the associated error and correct the indicated problem.

1525 MESSAGE **The pair with attribute ! was not modified (SDERR 1525)**

CAUSE The indicated attribute of the entity type/relationship type pair is not modified.

ACTION Refer to the associated error and correct the indicated problem.

1526 MESSAGE **No attributes were removed (SDERR 1526)**

CAUSE No attributes are removed from the entity type/relationship type.

ACTION Refer to the associated error and correct the indicated problem.

1527 MESSAGE **Attribute ! was not removed (SDERR 1527)**

CAUSE The indicated attribute is not removed from the entity type/relationship type.

ACTION Refer to the associated error and correct the indicated problem.

1528 MESSAGE **Illegal character in value for attribute ! (SDERR 1528)**

CAUSE An illegal character is found in the value for the indicated attribute.

ACTION Remove the illegal character and reissue the command.

1529 MESSAGE **No integer or fraction found in attribute ! (SDERR 1529)**

CAUSE No integer (base) or fraction (exponent) is found in the indicated number.

ACTION Correct the number so that a valid number is specified.

1530 MESSAGE **Number > largest representable value for attribute ! (SDERR 1530)**

CAUSE The value for the indicated attribute is too large to be represented in the system.

ACTION Reduce the size of the number and reissue the command.

1531 MESSAGE **Number < smallest representable value for attribute ! (SDERR 1531)**

CAUSE The value for the indicated attribute is too small to be represented in the system.

ACTION Increase the size of the number and reissue the command.

1532 MESSAGE **Number too large and illegal character for attribute ! (SDERR 1532)**

CAUSE The value for the indicated attribute is too large to be represented in the system and contains an invalid character.

ACTION Remove the invalid character and reduce the size of the number.

1533 MESSAGE **Number too small and illegal character for attribute ! (SDERR 1533)**

CAUSE The value for the indicated attribute is too small to be represented in the system and contains an invalid character.

ACTION Remove the invalid character and increase the size of the number.

1534 MESSAGE **Invalid alias specified for attribute ! (SDERR 1534)**

CAUSE An invalid alias value is specified for the indicated attribute.

ACTION Correct the value and reissue the command.

1535 MESSAGE **Invalid boolean specified for attribute ! (SDERR 1535)**

CAUSE An invalid boolean value is specified for the indicated attribute.

ACTION Correct the value (to TRUE or FALSE) and reissue the command.

1536 MESSAGE **Invalid character string specified for attribute ! (SDERR 1536)**

CAUSE An invalid character string value is specified for the indicated attribute.

ACTION Correct the value and reissue the command. Remember, quotes are required if the value wouldn't be accepted as a valid name.

1537 MESSAGE **Invalid date specified for attribute ! (SDERR 1537)**

CAUSE An invalid date value is specified for the indicated attribute.

ACTION Correct the value and reissue the command. Remember that all dates should be specified within quotes in the customized format of the Native Language Intrinsics.

1538 MESSAGE **Invalid time specified for attribute ! (SDERR 1538)**

CAUSE An invalid time value is specified for the indicated attribute.

ACTION Correct the value and reissue the command. Remember that all dates and time should be specified within quotes in the customized format of the Native Language Intrinsics.

1539 MESSAGE **Invalid integer specified for attribute ! (SDERR 1539)**

CAUSE An invalid integer value is specified for the indicated attribute.

ACTION Correct the value and reissue the command.

1540 MESSAGE **Invalid floating point value for attribute ! (SDERR 1540)**

CAUSE An invalid floating point value is specified for the indicated attribute.

ACTION Correct the value and reissue the command.

1541 MESSAGE **Invalid value for attribute ! (SDERR 1541)**

CAUSE An invalid value is specified for the indicated attribute.

ACTION Correct the value and reissue the command. Remember that valid values for SENSITIVITY are PRIVATE, READ, and MODIFY.

1542 MESSAGE **Text for ! must be specified within quotes (SDERR 1542)**

CAUSE Text for the indicated variable attribute must be specified within quotes.

ACTION Add the quotes and reissue the command.

1543 MESSAGE **Entity type ! does not exist (SDERR 1543)**

CAUSE The indicated entity type specified as a part of the relationship type does not exist in the dictionary.

ACTION Create the entity type before proceeding or specify an entity type that does exist.

1544 MESSAGE **Entity ! does not exist (SDERR 1544)**

CAUSE The indicated entity specified as part of the relationship does not exist in the dictionary.

ACTION Create the entity before proceeding or specify an entity that does exist.

1545 MESSAGE **EDIT-VALUE is not allowed for attributes of this type (SDWARN 1545)**

CAUSE Attributes of type ALIAS and VARIABLE cannot have attribute edit-values.

ACTION No action is needed. The attribute is created/modified as if no EDIT-VALUE clause was specified.

1546 MESSAGE **There is a problem with edit-value number ! (SDWARN 1546)**

CAUSE The indicated edit value in the EDIT-VALUE clause has a problem.

ACTION Refer to the associated error to see what the problem with the value is. Then, either ignore the problem if it is merely a warning and is OK or correct it and reenter the command with the corrected value. Note that if the associated problem is merely a warning, the attribute will be created or modified while if the problem is an error, no action will be taken.

1547 MESSAGE **There are too many edit-values specified for the attribute (SDERR 1547)**

CAUSE With the padding of the edit-values to the length of the attribute, the number of edit-values specified overflowed the maximum allowance of 255 characters for the edit buffer.

ACTION The EDIT-VALUE list should be inspected and reduced to a number that can fit in the buffer and the attribute modified to supply the new edit list. Remember, each value is padded to the length of the attribute and there is a maximum buffer length of 255 characters.

1548 MESSAGE **LENGTH required for types CHARACTER, FLOATING, and INTEGER (SDERR 1548)**

CAUSE An attribute of type CHARACTER, FLOATING, or INTEGER is being created, but no LENGTH keyword is used. When creating a new attribute, the length of the attribute must be specified if there is a choice between two or more lengths for the attribute.

ACTION Reenter the same command but include a LENGTH keyword clause that specify a length for the attribute being created.

1549 MESSAGE **Cannot specify variable attribute if have/establish common link (SDERR 1549)**

CAUSE An entity or relationship is being created with a common link and a variable length attribute assigned a value.

ACTION Create the entity or relationship with the common link and do not assign any value to the variable length attribute. Create the entity or relationship as a local entity or relationship without a common link. Or create the entity or relationship with the common link and then modify the common occurrence linked to add the variable length attribute text.

CAUSE An entity or relationship is being modified with a common link existing or being added and a variable length attribute assigned a value.

ACTION Modify the entity or relationship to add the common link and then modify the common entity or relationship to change the variable length attribute or modify the local entity or relationship to remove the common link (or not establish it) and assign the variable length text to the local occurrence.

Display/Report Messages (1600-1699)

1600 MESSAGE **No SECURE capability. Cannot qualify other SCOPE-OWNERs (SDWARN 1600)**

CAUSE The current open scope does not have SECURE capability. Accordingly, it can only retrieve the name of the object's owner scope if the current scope is the owner (otherwise, the owner name is blank). Because of this, if the qualifying SCOPE-OWNER qualifies retrieval to the current scope, everything will be fine. If the qualification contains anything else, the match against a blank name will fail and so the objects are NOT retrieved.

ACTION If the SCOPE-OWNER clause qualifies to the current scope, no action is needed. Also, if the SCOPE-OWNER contains selection criteria and if the current scope matches the criteria and it is OK that the list of objects retrieved will be limited to the ones owned by the current scope, no action is needed. However, if it is NOT OK to limit to the objects owned by the current scope, either switch to a scope that does have SECURE capability or don't use the SCOPE-OWNER field to qualify the owner in the current scope.

1601 MESSAGE **Attribute list is incomplete for entity type ! (SDERR 1601)**

CAUSE The attribute list being retrieved for the indicated entity type is not complete.

ACTION Correct the associated error and reissue the command.

1602 MESSAGE **Lock failed so partial results may be returned (SDWARN 1602)**

CAUSE The dictionary lock failed. Accordingly, the list of items retrieved may not include all items in the dictionary at the current time.

ACTION No action is required. The system returns results but there is no guarantee that they are complete. The DA should be notified of the problem. Internal Error.

1603 MESSAGE **Unlock failed so must terminate to prevent system hang (SDERR 1603)**

CAUSE The unlock failed. Accordingly, the system is terminated to exit the dictionary to insure that this run does not hold a lock on the dictionary.

ACTION Contact the DA. Internal Error.

1604 MESSAGE **Error encountered during a list retrieval (SDERR 1604)**

CAUSE An error was encountered during a list retrieval.

ACTION Refer to the associated error and correct the problem found.

1605 MESSAGE **Error while switching to or from the report domain (SDERR 1605)**

CAUSE An error was encountered while switching to or from the domain holding all of the SDMAIN stored reports.

ACTION Contact the DA. Internal Error.

1606 MESSAGE **NAME-ONLY used instead of ALL (SDWARN 1606)**

CAUSE The NAME-ONLY keyword is used and the ALL keyword ignored.

ACTION No action is needed. The next time the command is issued, specify only one information granularity option.

1607 MESSAGE **NAME-ONLY used instead of LIST (SDWARN 1607)**

CAUSE The NAME-ONLY keyword is used and the LIST keyword ignored.

ACTION No action is needed. The next time the command is issued, specify only one information granularity option.

1608 MESSAGE **LIST used instead of ALL (SDWARN 1608)**

CAUSE The LIST keyword is used and the ALL keyword ignored.

ACTION No action is needed. The next time the command is issued, specify only one information granularity option.

1609 MESSAGE **! is not allowed as a LIST parameter for RELATIONSHIPS (SDWARN 1609)**

CAUSE The indicated LIST option is not allowed as a LIST parameter when reporting on relationships.

ACTION No action is needed as the option is ignored. Remove the indicated option from the LIST options.

1610 MESSAGE **ATTRIBUTES entry overrides specific parameters (SDWARN 1610)**

CAUSE The ATTRIBUTES option for the LIST keyword overrides any specific attributes listed.

ACTION No action is needed. The next time the command is issued, if all the attributes are desired, omit specific attributes from the LIST list.

1611 MESSAGE **Cannot qualify a variable attribute (SDERR 1611)**

CAUSE Cannot qualify a variable length attribute.

ACTION Remove the variable length attribute qualifier from the ATTRIBUTE-LIST clause.

1612 MESSAGE **Attribute ! not related to the entity (SDERR 1612)**

CAUSE The indicated attribute is not related to the entity.

ACTION Remove the unrelated attribute or change the mistyped attribute to the correct attribute.

1613 MESSAGE **Attribute ! not related to the relationship (SDERR 1613)**

CAUSE The indicated attribute is not related to the relationship.

ACTION Remove the unrelated attribute or change the mistyped attribute to the correct attribute.

1614 MESSAGE **Invalid operation on a boolean for attribute ! (SDERR 1614)**

CAUSE The operation specified for the indicated Boolean attribute is invalid. Boolean attribute can only be qualified using relational operators = and <>.

ACTION Change the type of comparison to one including only = and/or <>.

1615 MESSAGE **Duplicate Item List entry ! (SDWARN 1615)**

CAUSE The indicated list option is listed twice in the LIST keyword clause.

ACTION No action is needed as the duplicate is ignored.

1616 MESSAGE **Duplicate Attribute List entry ! (SDERR 1616)**

CAUSE The indicated attribute is listed twice in the ATTRIBUTE-LIST keyword clause.

ACTION Remove the duplicate attribute (or combine the two into one) and reissue the command.

1617 MESSAGE **Must own Domain to report on scopes having access (SDWARN 1617)**

CAUSE The current scope must be the owner of the domain to allow retrieval of scope information.

ACTION Open the dictionary with the owner scope as the open scope.

1618 MESSAGE **Must own Entity to report on scopes having access (SDWARN 1618)**

CAUSE The current open scope must be the owner of the entity to allow retrieval of scope information.

ACTION Open the dictionary with the owner scope as the open scope.

1619 MESSAGE **Must own Relationship to report on scopes having access (SDWARN 1619)**

CAUSE The current open scope must be the owner of the relationship to allow retrieval of scope information.

ACTION Open the dictionary with the owner scope as the open scope.

1620 MESSAGE **Domain deleted or changed during processing (SDWARN 1620)**

CAUSE The domain being processed has been deleted or changed since the start of the operation.

ACTION Do nothing if this is not a problem. If a problem, open the dictionary in EXCLUSIVE-UPDATE mode and repeat the command.

1621 MESSAGE **Entity deleted or changed during processing (SDWARN 1621)**

CAUSE The entity being processed has been deleted or changed since the start of the operation.

ACTION Do nothing if this is not a problem. If a problem, open the dictionary in EXCLUSIVE-UPDATE mode and repeat the command.

1622 MESSAGE **Relationship deleted or changed during processing (SDWARN 1622)**

CAUSE The relationship being processed has been deleted or changed since the start of the operation.

ACTION Do nothing if this is not a problem. If a problem, open the dictionary in EXCLUSIVE-UPDATE mode and repeat the command.

1623 MESSAGE **Too many entities in ! for the relationship-type (SDERR 1623)**

CAUSE The number of entities specified as composing the relationship is greater than the number of entity types making up the relationship type.

ACTION Reduce the number of entities to no more than the number of entity types in the relationship type.

1624 MESSAGE **List entry ! is not allowed for compiled dictionaries (SDWARN 1624)**

CAUSE The list entry specified is not allowed for compiled dictionaries. That functionality is not available there.

ACTION No action is needed. The option will be ignored. This information is not implemented in compiled dictionaries so you must move to a master dictionary to retrieve this information.

Start/Save Messages (1700-1724)

1700 MESSAGE **START command has been cancelled (SDWARN 1700)**

CAUSE The START/SAVE pair is cancelled so you are returned to the normal command processing mode.

ACTION No action is needed.

1701 MESSAGE **Create capability is required to SAVE a report (SDERR 1701)**

CAUSE The current open scope does not have CREATE capability (scope right) and so cannot SAVE a report. Accordingly, you are not allowed to initiate a START/SAVE pair.

ACTION Switch to a scope having create capability.

1702 MESSAGE **A ! command has already been defined (SDERR 1702)**

CAUSE Only one of the indicated command (CONFIGURE, FORMAT, or REPORT) can be issued in each stored report.

ACTION No action is needed. The first instance of the command is stored with the report.

1703 MESSAGE **Cannot specify ! after a REPORT command (SDERR 1703)**

CAUSE The specified command (CONFIGURE or FORMAT) cannot be issued after the REPORT command has been issued for a stored report.

ACTION Cancel the current report (using [[Control]] Y) if interactive. Then START the report again issuing the indicated command before the REPORT command. Or, SAVE the current report and then use the EDIT REPORT command to insert the indicated command into the report.

1704 MESSAGE **A START command is already active (SDERR 1704)**

CAUSE A START/SAVE pair is already active so another cannot be initiated.

ACTION Complete the current START/SAVE pair and then another START can be issued. Remember, [[Control]] Y can be used to cancel the current START/SAVE pair.

1705 MESSAGE **No START command has been issued (SDERR 1705)**

CAUSE A SAVE cannot be issued here because no START has been issued to initiate the storage of commands.

ACTION Issue a START command and enter the command to be stored. Then, the SAVE command will be accepted.

1706 MESSAGE **No REPORT command so START is active. <Control-Y> to stop. (SDERR 1706)**

CAUSE A REPORT command is required within a START/SAVE pair. As one has not been issued, the START remains active so one can be entered.

ACTION Enter a REPORT command and then issue the SAVE. Or, use the [[Control]] Y (if interactive) to cancel the START.

1707 MESSAGE **No REPORT command was specified. START is cancelled (SDERR 1707)**

CAUSE A REPORT command is required within a START/SAVE pair. As input is NOT interactive, the START is cancelled so further processing can take place.

ACTION Update the command file to include a REPORT within the START/SAVE.

1708 MESSAGE **Cannot create REPORT so START is active. <Control-Y> to stop (SDERR 1708)**

CAUSE The indicated report cannot be created. Accordingly, the START remains active so the problem with the report can be corrected without losing the commands to be saved.

ACTION Refer to the associated error and correct the problem indicated. Then, issue the SAVE command. Or, issue a [[Control]] Y (if interactive) to cancel the START.

1709 MESSAGE **Cannot create REPORT. START is cancelled (SDERR 1709)**

CAUSE The indicated report cannot be created. As input is NOT interactive, the START is cancelled so further processing can take place.

ACTION Refer to the associated error and correct the problem indicated.

1710 MESSAGE **Definition too long so REPORT not stored and START cancelled (SDERR 1710)**

CAUSE The commands to be stored are too long to store in the dictionary. Because there is no way to shrink the commands, the START is cancelled.

ACTION Reissue the START and enter the commands again. However, this time use simpler commands with fewer lines or simply put more data on each line so there are less lines to the command.

1711 MESSAGE **Definition SAVE failed so REPORT not stored and START cancelled (SDERR 1711)**

CAUSE The report cannot be stored. Because the data could not be placed in the dictionary, the START is cancelled.

ACTION Refer to the associated error and correct the problem indicated. The problem is probably a database capacity error.

1712 MESSAGE **Description was too long. It was truncated and stored (SDWARN 1712)**

CAUSE The description supplied with the report is too long so it is truncated.

ACTION No action is needed. As much of the description as can be handled is stored and the remainder discarded.

1713 MESSAGE **Description SAVE failed. Report saved without Description (SDERR 1713)**

CAUSE The report's description cannot be saved. Because it cannot be placed in the dictionary, it is discarded and the report saved without a description.

ACTION Refer to the associated error and correct the indicated problem. The problem is probably a database capacity error. The description can be added later, if desired, using the MODIFY REPORT command.

Stored Report Messages (1725-1749)

1725 MESSAGE **Report does not exist (SDERR 1725)**

CAUSE The report specified by the user does not exist in the dictionary.

ACTION Work with a report that exists or create this report before operating on it.

1726 MESSAGE **Report already exists (SDERR 1726)**

CAUSE The report specified by the user already exists in the dictionary.

ACTION Either no action is needed as the report already exists or a modify operation on the report should be used.

1727 MESSAGE **Internal report name exists (SDERR 1727)**

CAUSE The internal name specified for the report already exists in the dictionary.

ACTION Specify a unique internal name for the report.

1728 MESSAGE **External report name exists (SDERR 1728)**

CAUSE The external name specified for the report already exists in the dictionary.

ACTION Specify a unique external name for the report.

1729 MESSAGE **Current scope is not the report owner (SDERR 1729)**

CAUSE The current scope is not the owner of the specified report and ownership is required for the desired operation.

ACTION Open the dictionary with the owner scope to perform the operation.

1730 MESSAGE **Scope lacks sufficient access to the report (SDERR 1730)**

CAUSE The current scope does not have modify access to the specified report.

ACTION Only the owner can modify a report so open the dictionary as the owner to modify.

Execute Messages (1750-1799)

1750 MESSAGE **Both ENTITY and RELATIONSHIP keywords cannot be specified (SDERR 1750)**

CAUSE As the underlying report is either a REPORT ENTITY or a REPORT RELATIONSHIP, only one of the ENTITY/RELATIONSHIP keywords can be specified.

ACTION Reissue the command deleting the unneeded keyword clause.

1751 MESSAGE **Could NOT retrieve the report named ! (SDERR 1751)**

CAUSE Cannot retrieve the specified report.

ACTION Refer to the associated error and correct the indicated problem.

1752 MESSAGE **A ! command cannot be issued after a REPORT command (SDERR 1752)**

CAUSE The specified command (CONFIGURE or FORMAT) cannot be placed after the REPORT command in a stored report.

ACTION Use the EDIT REPORT command to move the indicated command to before the REPORT command (or to delete it completely if there is already one before the REPORT command).

1753 MESSAGE **Cannot issue two ! commands (SDERR 1753)**

CAUSE Two of the specified commands (CONFIGURE, FORMAT, or REPORT) are not allowed in the same stored report.

ACTION Use the EDIT REPORT command to delete one of the indicated commands.

1754 MESSAGE **! is an invalid command in a stored report (SDERR 1754)**

CAUSE The specified command is invalid in a stored report.

ACTION Use the EDIT REPORT command to remove the invalid command.

1755 MESSAGE **The keyword is incompatible with the report type (SDERR 1755)**

CAUSE Either the ENTITY keyword is specified and the stored command is a REPORT RELATIONSHIP or the RELATIONSHIP keyword is specified and the stored command is a REPORT ENTITY.

ACTION Reissue the command with the keyword corresponding to the stored command.

1756 MESSAGE **Source and indicated entity-types do not match for report ! (SDERR 1756)**

CAUSE The entity type of the REPORT command and the entity type specified by the SOURCE-POSITION keyword are not the same.

ACTION Revise the REPORT command so that the specified source entity type matches the entity type of the REPORT command.

1757 MESSAGE **SOURCE-POSITION required as is > 1 match for report ! (SDERR 1757)**

CAUSE There are two or more possible position matches for the source entity so the SOURCE-POSITION keyword must be specified.

ACTION Reissue the command with the SOURCE-POSITION keyword specified.

1758 MESSAGE **Source entity-type is NOT in the relationship-type for report ! (SDERR 1758)**

CAUSE The RELATIONSHIP-TYPE to be followed does not involve the source ENTITY-TYPE.

ACTION Revise the relationship type to be followed so that it involves the source entity type.

1759 MESSAGE **REPORT-POSITION required as > 2 entity-types for report ! (SDERR 1759)**

CAUSE A 3-way (or more) relationship is being followed to the nested report. Accordingly, the REPORT-POSITION keyword is required.

ACTION Reissue the command with the REPORT-POSITION keyword added.

1760 MESSAGE **Error while retrieving entities for report ! (SDERR 1760)**

CAUSE Encountered an error while retrieving the entities to be used for the nested report.

ACTION Refer to the associated error and correct the problem indicated.

1761 MESSAGE **Illegal REPORT-POSITION value given for report ! (SDERR 1761)**

CAUSE The REPORT-POSITION specified in the indicated report is not legal (there is no such position in the relationship type to follow).

ACTION Change the REPORT-POSITION to a value that is present in the relationship to be followed.

1762 MESSAGE **Stored and indicated entity-types don't match for report ! (SDERR 1762)**

CAUSE The entity types in the REPORT command and the stored report do not match.

ACTION Revise the REPORT command so the entity types match or specify a different stored report with the correct entity type (create one if needed).

1763 MESSAGE **Error found in stored report ! (SDERR 1763)**

CAUSE An error was encountered while processing the indicated report.

ACTION Refer to the associated error and correct the indicated problem.

1764 MESSAGE **Report and Source position cannot be the same for report ! (SDERR 1764)**

CAUSE The same position was specified for both SOURCE-POSITION and REPORT-POSITION.

ACTION Reissue the report with the position values updated so they are not both the same.

1765 MESSAGE **The stored report must be a REPORT ENTITY (SDERR 1765)**

CAUSE For nested EXECUTEs, the stored report must be a REPORT ENTITY while in this case it is a REPORT RELATIONSHIP.

ACTION Change the report chained to so that the chained report is a REPORT ENTITY.

Miscellaneous Messages (1800-1899)

1800 MESSAGE **Catalog error ! encountered during catalog read (SDERR 1800)**

CAUSE An error was encountered while retrieving information from the message catalog.

ACTION Contact the DA.

1801 MESSAGE **Could not find message set ! in the catalog (SDERR 1801)**

CAUSE The indicated message set could not be found in the catalog.

ACTION Contact the DA to get a new catalog with the indicated set present.

1802 MESSAGE **Could not find message (set:number) ! in the catalog (SDERR 1802)**

CAUSE Cannot find the desired message in the specified set in the catalog.

ACTION Contact the DA to get a new catalog with the indicated message present.

1810 MESSAGE **Bad error message from FErrMsg Intrinsic (SDERR 1810)**

CAUSE A file system error occurred, but the associated message cannot be retrieved from the FErrMsg intrinsic.

ACTION Contact the DA.

1811 MESSAGE **Error message ! not found in the catalog. Catalog error ! (SDERR 1811)**

CAUSE The indicated error message cannot be found in the message catalog.

ACTION Contact the DA to get a new catalog with the indicated error message present.

1812 MESSAGE **Native language error (SDERR 1812)**

CAUSE An error was detected during a call to the Native Language Intrinsics.

ACTION Contact the DA. Internal Error.

1813 MESSAGE **Date returned from the dictionary was invalid (SDERR 1813)**

CAUSE The data for the DATE-CREATED or DATE-CHANGED attribute field is not in the valid format for a System Dictionary date field.

ACTION Contact the DA. Internal error.

1814 MESSAGE **Sort Intrinsic error number ! (SDWARN 1814)**

CAUSE An error was detected during a call to the Sort intrinsics.

ACTION Use the NOSORT option in all cases so the error is avoided. In addition, contact the DA. Internal Error.

1820 MESSAGE **Heap overflow. Command is too complex (SDERR 1820)**

CAUSE All of the available heap space in the system has been allocated. The command is too long and/or complex.

ACTION Simplify the command by breaking it into two or more pieces and reissue the pieces.

CAUSE All of the available heap space in the system has been allocated. There are too many nested levels in the REPORT command.

ACTION Limit the number of nested levels in the report and/or limit the amount of information and qualification at each level of the report.

1821 MESSAGE **Library error ! encountered. Terminate program (SDERR 1821)**

CAUSE A Pascal error was encountered.

ACTION Contact the DA (remember to supply the library error number). Internal Error.

1830 MESSAGE **MPE command is too long (SDERR 1830)**

CAUSE The entered MPE command is too long.

ACTION Limit the MPE command to 256 characters and reenter it.

1831 MESSAGE **Invalid MPE command (SDERR 1831)**

CAUSE The specified MPE command is not a legal MPE command.

ACTION Enter a legal MPE command.

CAUSE The specified MPE command is not allowed to be issued from within SDMAIN.

ACTION Enter an MPE command that can be issued through the command intrinsic.

1832 MESSAGE **MPE Executor error number ! (SDERR 1832)**

CAUSE An error was encountered while executing the MPE command.

ACTION Use the error number to look up the error for the specified command in the MPE Intrinsics Manual.

1840 MESSAGE **Intrinsic logging is already ! (SDWARN 1840)**

CAUSE Intrinsic logging is already set in the indicated position.

ACTION None is needed as logging is already set as desired.

1841 MESSAGE **Version is already set to ! status (SDWARN 1841)**

CAUSE The status of the version is already set to the status specified.

ACTION None is needed as the status of the version is already what is desired. However, a check should be made to make sure that the version name is correct and you are not trying to set the status of a different version than expected.

B SDMAIN Command Abbreviations

SDMAIN Command Abbreviations

Abbreviations provide a shorthand method for specifying commands in SDMAIN. Because of the need to preserve uniqueness in the abbreviations, some commands do not have an abbreviated form. In these cases, the full command and the abbreviation are the same. You can use an abbreviation anywhere the corresponding command word is used. It can also be localized.

Note that abbreviations follow the same rules as user-defined names do, except that they can contain from one to five characters. Within each of the following categories, abbreviations are unique; they may be similar between sets, however.

Alignment Options

ALIGNMENT OPTION	ABBREVIATION
ALIGNED	AL
FIXED-ALIGNED	FA
UNALIGNED	UN

Attribute Data Type

DATA TYPE	ABBREVIATION
ALIAS	A
BOOLEAN	B
CHARACTER	C
FLOATING	F
INTEGER	I
VARIABLE	V

Booleans

BOOLEAN	ABBREVIATION
TRUE	T
FALSE	F

Commands

COMMAND	ABBREVIATION
ADD	A
COMMENT	COM
CONFIGURE	CON
COPY	CO
CREATE	C
DEFINE	DEF
DELETE	DEL
DISPLAY	DIS
EDIT	ED
EXECUTE	EX
EXIT	E
FORMAT	F
HELP	H
INCLUDE	IN
MODIFY	M
OPTIONS	OP
REDO	REDO
REMOVE	REM
RENAME	REN
RENUMBER	RENU
REPORT	REP
RESEQUENCE	RESE
RESTRUCTURE	REST
SAVE	SA
SETVERSION	SET
SHOW	SH
SHOWMACRO	SM
SHOWREDO	SR

COMMAND	ABBREVIATION
START	ST

Conjunctions

CONJUNCTION	ABBREVIATION
AND	AND
OR	OR

Domain Access

DOMAIN ACCESS	ABBREVIATION
PRIVATE	PRI
PUBLIC	PUB

Keywords

KEYWORD	ABBREVIATION
ALIGNMENT	ALIGN
ALL	ALL
ATTRIBUTE	A
ATTRIBUTE-LIST	AL
BEFORE-ENTITY	BE
BOTTOM	BOT
COMMAND-LOG	CLOG
COMMON	C
DESCRIPTION	DESC
DICTIONARY	DICT
DOMAIN	D
DOUBLE-SPACE	DS
EDIT-VALUE	EV
ENTITY	E
ENTITY-TYPE	ET

KEYWORD	ABBREVIATION
FORMAT-VARIABLE	FV
INCREMENT	INC
INTERNAL	INT
INTRINSIC-LOG	ILOG
LAST-ONLY	LAST
LEFT	LEFT
LENGTH	LEN
LIST	L
LOG	LOG
MACRO	MAC
NAME-MODE	NM
NAME-ONLY	NAME
NEW-PASSWORD	NP
NEW-SCOPE	NS
NOSORT	NO
OPEN-MODE	OM
OUTPUT	OUT
PAGE	PAGE
PAGE-HEADER	PH
PAGE-LENGTH	PL
PAGE-WIDTH	PW
PASSWORD	P
POSITION	POS
PROMPT	PROM
PURGE	PURGE
RELATIONSHIP	R
RELATIONSHIP-CLASS	RC
RELATIONSHIP-TYPE	RT
REPORT-POSITION	RP
RIGHT	RIGHT

KEYWORD	ABBREVIATION
SCOPE	S
SCOPE-ACCESS	SA
SCOPE-OWNER	SO
SCOPE-RIGHTS	SR
SCREEN-LENGTH	SL
SCREEN-WIDTH	SW
SENSITIVITY	SEN
SINGLE-SPACE	SS
SOURCE-POSITION	SP
START-POSITION	SPOS
STATUS	STAT
SUB-REPORT	SUB
TITLE	TI
TOP	TOP
TYPE	T
VERSION	V
VERSION-INTERNAL	VINT

List Items

LIST ITEM	ABBREVIATION
ATTRIBUTES	A
COMMON	C
NAME	N
PRIMARY	PRIM
RELATIONDATA	RD
RELATIONSHIPS	R
SCOPES	S
SCOPE-ACCESS	SA
SYNONYMS	SYN
VERSIONS	V

Name Modes

NAME MODE	ABBREVIATION
INTERNAL	INT
EXTERNAL	EXT

Open Modes

OPEN MODE	ABBREVIATION
CUSTOMIZATION	CU
EXCLUSIVE-UPDATE	EU
READ-ALLOW-READ	RAR
READ-ONLY	RO
SHARED-UPDATE	SU

Option Flags

OPTION FLAG	ABBREVIATION
OFF	OFF
ON	ON

Output Destination

DESTINATION	ABBREVIATION
PRINTER	PRINT
TERMINAL	TERM

Responses

RESPONSE	ABBREVIATION
NO	N
YES	Y

Scope Access

ACCESS	ABBREVIATION
MODIFY	M
OWNER	O
READ	R

Scope Rights

RIGHT	ABBREVIATION
CREATE	C
DOMAIN	D
EXTEND	E
READ	R
SECURE	S
VERSION	V

Sensitivity

SENSITIVITY	ABBREVIATION
MODIFY	M
PRIVATE	P
READ	R

Subcommands

SUBCOMMMAND	ABBREVIATION
ATTRIBUTE	A
DOMAIN	D
ENTITY	E
ENTITY-TYPE	ET
ENTITY-TYPE-ATTRIBUTE	ETA
RELATIONSHIP	R
RELATIONSHIP-CLASS	RC

SUBCOMMMAND	ABBREVIATION
RELATIONSHIP-TYPE	RT
RELATIONSHIP-TYPE-ATTRIBUTE	RTA
REPORT	REP
SCOPE	S
SCOPE-DOMAIN	SD
SCOPE-ENTITY	SE
SCOPE-RELATIONSHIP	SR
SYNONYM	SYN
VERSION	V

Version Status

STATUS	ABBREVIATION
ARCHIVAL	A
PRODUCTION	P
TEST	T