SAS Publishing

# SYSTEM 2000®
# Quick Reference Guide

Version 12
First Edition

The Power to Know®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 1991. *SYSTEM 2000® Quick Reference Guide, Version 12, First Edition*. Cary, NC: SAS Institute Inc.

**SYSTEM 2000® Quick Reference Guide, Version 12, First Edition**

ISBN 1-55544-187-4

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, August 1991

# Contents

# Using This Book

## Purpose

This book provides quick reference information about the SYSTEM 2000 Self-Contained Facility (SCF) commands, Version 12. SYSTEM 2000 software is SAS Institute's data management system for mainframe computer systems running under MVS and CMS operating systems.

The commands are organized first by processor, then alphabetically within each processor. If you cannot remember the keyword for a command, you can refer to the index to locate a command by task.

You can use this quick reference guide to find forgotten syntax, a list of options, or rules that affect issuing a specific command. This guide assumes that you are a knowledgeable SYSTEM 2000 user and that you understand how the various commands work. Given this assumption, some explanations concerning commands are omitted, and the rules are written in a concise style.

This guide addresses commands; it does not document SYSTEM 2000 features. For example, Coordinated Recovery is not explained in this guide.

## Organization of This Book

This book has quick reference material and Appendices. The following sections list the chapters they contain.

**Quick Reference Chapters** The quick reference chapters show the purpose, syntax, rules, and examples for each SCF command.

Chapter 1, "CONTROL Language Commands"

Chapter 2, "DEFINE Language Commands"

Chapter 3, "QUEST Language Commands"

Chapter 4, "System-Wide Commands

**Appendices** The appendices provide information about various special areas.

Appendix A, "Special Characters"

Appendix B, "Reserved Words and Characters"

Appendix C, "Reserved File Names"

Appendix D, "Collating Sequence and Character Set"

## Reference Aids

The following reference aid is located at the end of the book:

| | |
|---|---|
| Index | identifies pages where specific commands and topics are discussed. |

## Typographical Conventions

You will notice several type styles throughout the book. Their different purposes are summarized here.

| | |
|---|---|
| roman | is the basic type style used for most text. |
| UPPERCASE ROMAN | is used for references in the text to SYSTEM 2000 command keywords and database component names. |
| *italic* | is used for terms that are defined in the text, to emphasize important information, and for comments in sample code. |
| `MONOSPACE` | is used to show examples of commands. This book uses uppercase type for SYSTEM 2000 commands. You can enter your own commands in lowercase, uppercase, or a mixture of the two. |

## Syntax Conventions

SYSTEM 2000 command syntax uses the following syntax notation:

| | |
|---|---|
| UPPERCASE ROMAN | indicates keywords. They must be spelled exactly as given in the syntax shown. Abbreviations for keywords are also shown in uppercase. If you have more than one choice, the choices are stacked vertically with a bar to the left. Select only one. A keyword without brackets is a required keyword. |
| *italic* | indicates generic terms representing words or symbols that you supply. If the term is singular, supply one instance of the term. If the term is plural, you can supply a list of terms, separated by commas. If you have more than one choice, the choices are stacked vertically with a bar to the left. Select only one. A term without brackets is a required term. |
| [ ] | enclose an optional portion of syntax. The brackets are not part of the command syntax. Multiple choices are stacked vertically with a bar to the left. Select only one. |

| | |
|---|---|
| \| (vertical bar) | in syntax, means to select one of the vertically stacked choices. If the choices are in brackets, the choice is optional; if not, a choice is required. Then continue to the next portion of syntax, shown on the top line of the command syntax.<br><br>In margins, indicates a technical change in the text for the latest version of the software. |
| . . . (ellipsis)<br>.<br>.<br>. | indicates that the previous syntax can be repeated. In examples, either vertical or horizontal ellipses indicates an omitted portion of output or a command. |
| b̸ | indicates a significant blank in syntax or output. Generally, spaces indicate blanks, and the b̸ is used only for emphasis. |
| * (asterisk) | is the default system separator throughout this book. |
| END | is the default entry terminator word throughout this book. |

Note that symbols within syntax (such as parentheses, asterisks, or commas) are required unless enclosed in brackets or specifically noted as optional.

## Example Conventions

The examples show you how to use the commands. You can run most of the examples using the EMPLOYEE database. Comments appear in italics and within parentheses.

Examples are shown in uppercase. However, you can enter your own commands in lowercase, uppercase, or a mixture of both. By default, SYSTEM 2000 software translates keywords and component names entered in lowercase to all uppercase before processing, except when the data come from an alternate Command File or Data File. Due to this translation, you cannot have, for example, uppercase C3 referring to component number 3 and lowercase c3 referring to component name c3. Lowercase c3 will always be recognized as component number 3.

## Page Numbering Conventions

Pages are numbered sequentially within the chapter number, for example, 3-1, 3-2, and so on represents page 1, 2, and so on in Chapter 3.

## Additional Documentation

There are many SYSTEM 2000 publications available. To receive a free *Publications Catalog*, write to the following address:

SAS Institute Inc.
Book Sales Department
SAS Campus Drive
Cary, NC 27513

The books listed below should help you find answers to questions you may have about SYSTEM 2000 software.

- *SYSTEM 2000 Introductory Guide for SAS Software Users, Version 6, First Edition* (order #A55010) provides introductory information for SYSTEM 2000 software. It also explains how to use the SAS System with SYSTEM 2000 databases and provides examples.

- *SAS/ACCESS Interface to SYSTEM 2000 Data Management Software: Usage and Reference, Version 6, First Edition* (order #A56064) documents the ACCESS procedure, the DBLOAD procedure, and the QUEST procedure for the SAS/ACCESS interface to SYSTEM 2000 software. It explains how to create SAS/ACCESS descriptor files and how to use SAS programs with SYSTEM 2000 databases. Examples are provided.

- *SYSTEM 2000 DEFINE Language, Version 12, First Edition* (order #A55012) serves as a usage and reference manual for the DEFINE language. This manual explains how to create and modify a SYSTEM 2000 database definition. It includes a detailed description and examples for each DEFINE command.

- *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* (order #A55011) provides concepts and reference material for the SYSTEM 2000 QUEST language and for system-wide commands. For example, this book documents retrieval commands, such as LIST and PRINT, the where-clause, the ordering-clause, update commands, such as INSERT TREE and CHANGE value, and so on.

- *SYSTEM 2000 CONTROL Language, Version 12, First Edition* (order #A55013) describes how to use the CONTROL language to perform administrative tasks for SYSTEM 2000 databases, such as saving and restoring databases, assigning password security, specifying input and output files, and using the Coordinating Recovery feature.

- *SYSTEM 2000 REPORT Language, Version 12, First Edition* (order #A55014) explains how to use the REPORT language to produce reports from information contained in a SYSTEM 2000 database. This manual contains detailed reference material for using REPORT commands under MVS and CMS. Also included are examples of each command, of report composition, and of actual output.

- *SYSTEM 2000 PLEX Manual, Version 12, First Edition* (order #A55017) explains how to use the SYSTEM 2000 PLEX language. This manual details the use of PLEX commands and the methods of executing PLEX programs. It provides specific COBOL, PL/I, FORTRAN, and Assembler PLEX reference material and examples. The manual assumes a working knowledge of the programming language with which you will use SYSTEM 2000 software.

- *SYSTEM 2000 Messages and Codes, Version 12, First Edition* (order #A55015) contains all messages and codes issued by SYSTEM 2000 software. Probable causes and corrective actions accompany all error messages and warnings.

- *SYSTEM 2000 Product Support Manual, Version 12, First Edition* (order #A55016) describes how to configure SYSTEM 2000 software for single-user and Multi-User environments, how to specify SYSTEM 2000 files, and how to use user exits. Also included are descriptions of the executable modules, all execution parameters, the Accounting Log, and the Diagnostic Log.

- *SYSTEM 2000 CICS Interface, Release 11.5 under IBM OS* (order #A5511) provides supplemental details for using SYSTEM 2000 software under CICS at the macro level.

- *SYSTEM 2000 Interface to CICS (Command Level), Version 12, First Edition* (order #A55018) provides supplemental details for using SYSTEM 2000 software under CICS (command level). It includes system requirements, execution parameters, and available options.

- *SYSTEM 2000 CMS Supplement, Version 12, First Edition* (order #A55019) contains techniques, options, and commands that pertain to SYSTEM 2000 software under CMS.

- *SYSTEM 2000 DBMS Interactive Report Writing with Genius, Release 11.5 under IBM OS and CMS* (order #A5577) explains how to use the interactive Genius facility for producing reports and listings of data from SYSTEM 2000 databases. This manual contains detailed reference material and examples for all Genius prompts and user responses under MVS and CMS.

- *SYSTEM 2000 DBMS QueX User's Guide, Release 11.5A under IBM CICS, TSO, and CMS* (order #A5563) explains how to use the menu-driven QueX facility to enter data into and retrieve data from SYSTEM 2000 databases. This manual includes a step-by-step tutorial and reference information for QueX sessions.

- *SYSTEM 2000 DBMS QueX DBA Guide, Release 11.5A under IBM CICS, TSO, and CMS* (order #A5588) explains how to build user views for QueX sessions. This manual includes instructions for building, rebuilding, and deleting user views, for defining and deleting links in user views, and for maintaining the QueX system.

- *Technical Report: S2-106 Multi-User Tuning Tools for SYSTEM 2000 Software, Release 11.6 under IBM OS and CMS* (order #A55001) describes the Tuning Tools feature for SYSTEM 2000 software. The Multi-User status console commands display details about thread, scratch pad, buffer, and queue usage. The Accounting and Diagnostic Log reports display information about system performance or about specific jobs and job types running in the system. Data are extracted from these logs into a SAS data set for use in reports, and they are summarized into historical SAS data sets for more general monthly, quarterly, or annual reports.

- *Technical Report: S2-107 XBUF Caching Feature in SYSTEM 2000 Software, Release 11.6 under IBM OS* (order #A55002) documents the Extended Buffer Manager (XBUF) feature, which provides several caching techniques for both single-user and Multi-User jobs. These techniques include simple "front-end" XBUF macros, with which you can take advantage of dual logging, statistics gathering, cache modeling, and caching of database files without having to employ the more complicated CACHE macro. Later, if you need to tailor caching for special applications, you can refer to the CACHE macro documentation also included in this report.

- *Technical Report: S2-110 QUEUE Language for SYSTEM 2000 Software, Release 11.6 under IBM OS and CMS* (order #A55009) documents all aspects of the QUEUE language. Used together with the QUEST language manual, this technical report provides complete information on using the QUEUE facility in SYSTEM 2000 software.

- *SAS Technical Report S2-111, SYSTEM 2000 Internals: Database Tables, COMMON Blocks, and Debug Utilities, Version 12* (order #A55022) describes the structure of the internal tables in a SYSTEM 2000 database. The report details all table entries and the fields within the entries. Sample illustrations are included. In addition, the Update Log and Rollback Log formats are given. An Appendix lists all SYSTEM 2000 COMMON Data Areas, with an index and cross reference. This material is supplemental and is not required to use SYSTEM 2000 software.

- *SYSTEM 2000 Glossary and Subject Index, Version 12, First Edition* (order #A55021) defines specific terminology in SYSTEM 2000 software publications and includes a comprehensive index by subject to all Institute publications documenting SYSTEM 2000 software.

# Changes and Enhancements

## Introduction

This section lists the changes and enhancements that affect SCF commands for Version 12. This information is intended for users who have previous experience with the SCF commands.

## Mixed Case Data

Version 12 accepts and recognizes keywords and component names entered in lowercase. By default, the software translates data to all uppercase before processing, except when the data come from an alternate Command File or Data File.

The one restriction with this enhancement is that all component names are uppercased by the software when you define the database. Prior to Version 12, uppercase C3 referred to component number 3 while lowercase c3 referred to component name c3. For example, it was possible for component number 1 to have a component name of c3. With Version 12, lowercase c3 will be uppercased by the software and therefore will always be recognized as component number 3.

To disable uppercase translation, you can set the execution parameter OPT043 to YES.

## Maximum Number of Components

The maximum number of components for a database is 10,000. This number includes strings and functions as well as schema items and schema records. Therefore, the range of component numbers is 1 to 9999. Also, the numbered label for ad hoc function output starts at 10001 rather than 5001.

## IN and CTNSIN Operators

Two new where-clause operators, IN and CTNSIN, are available. These predicate operators imply two or more conditions joined with the OR operator. The IN operator implies an EQ condition, and the CTNSIN operator implies the CONTAINS operator. The IN and CTNSIN operators save time and avoid typing errors when you want to specify several similar conditions that have different values for the same schema item.

## Dynamic Allocation of User Files

You do not have to allocate the Message File and Report File (S2KMSG); SYSTEM 2000 software allocates S2KMSG dynamically if it does not find it in the JCL or CLIST. Also, the Command File and Data File (S2KCOMD) do not have to be allocated in a CLIST, but S2KCOMD must be specified in the batch JCL.

## ALLOC and FREE Commands

The new CONTROL language ALLOC and FREE commands enable you to allocate and deallocate database files dynamically within an SCF session. If SYSTEM 2000 software does not find DD statements in the JCL or CLIST, it automatically allocates the database files with the default data set name, or it uses the specifications given in an ALLOC command. The FREE command dynamically deallocates database files.

You can also set up an S2KDBCNT table of database names and restrict access to only those databases by setting the new ALLOC execution parameter to ALLOC=TBL. For more information about new execution parameters and dynamic allocation of database files, as well as all work files, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition* and the *SYSTEM 2000 Product Support Manual, Version 12, First Edition.*

## Simplified SAVE and RESTORE Syntax

The SAVE and RESTORE command syntax no longer requires any options. SYSTEM 2000 software can dynamically allocate the Savefile and Keepfile using the prefix and the default data set name. The DDname of the Savefile is the first seven characters of the database name (with Xs replacing blanks) plus a suffix of S; the Keepfile DDname is similar with a suffix of K. TAPES2K and KEEPFILE are no longer honored as DDnames. The new DDnames allow several users to save or restore different databases concurrently in a Multi-User session.

Also, KEEP and APPLY processing for different databases can run simultaneously. In addition, all update logging is in indirect mode; INDIRECT and DIRECT syntax is no longer valid.

If you do not include DD statements for the Savefile or Keepfile in the JCL or CLIST, SYSTEM 2000 software allocates them using the defaults or the options you specified in the SAVE or RESTORE command. The Savefile is now a QSAM file.

The following commands save the database currently in use. In the second example, the slash activates update logging.

```
SAVE:
SAVE/:
```

The following RESTORE commands restore the EMPLOYEE database. In the second example, the slash activates update logging.

```
RESTORE EMPLOYEE:
RESTORE EMPLOYEE/:
```

Chapter 1

# CONTROL LANGUAGE COMMANDS

# ALLOC

Allocates new or existing database files dynamically within an SCF session if they are not allocated in the JCL or CLIST.

```
ALLOC database [,DSN=dsn], DISP=NEW  [,options]  :

ALLOC database [,DSN=dsn], DISP=|OLD    :
                                |SHR
```

*database* is the name of a new database to be created or an existing database. It can have 1 to 16 characters.

*dsn* is a data set name of up to 44 characters. Use quotes for a fully qualified name. If you do not specify the data set name, the default is the prefix followed by the first seven characters of the database name (with embedded blanks replaced by Xs) and a suffix of S for the Savefile and K for the Keepfile.

DISP specifies the disposition of NEW for new database files. For existing database files, the disposition is OLD or SHR. DISP=SHR in a single-user job causes read-only access to the database. OLD is the default.

*options* is a list of one or more parameters separated by commas. You can use these options to change the default attributes for all new files or specific new files. They are ignored for existing database files. The options are shown below; *n* is an integer 1 through 8, representing one of the database files.

| Option | Description | Default |
|---|---|---|
| BLK= | any valid SYSTEM 2000 block size | 6216 |
| BLK*n*= | any valid SYSTEM 2000 block size | the BLK value |
| UNIT= | any valid unit for your site | SYSDA |
| UNT*n*= | any valid unit for your site | the UNIT value |
| VOL= | any valid volser | scratch volume |
| VOL*n*= | any valid volser | the VOL value |
| FILES= | ALL | Files 1 through 8 |
| FILES= | 1, 2, 3, 4, 5, 6, 7, or 8 | that file only |
| | FILES option not specified | Files 1 through 6 only |
| SPACE= | SMALL or LARGE | SMALL |
| SPC*n*= | (TRK \| CYL,*prim*,*sec*) | the SPACE values |

The SPACE option determines the database file sizes for Files 1 through 8. For the SPC*n* option, *prim* and *sec* are integers that specify primary and secondary space, respectively, for the file indicated by *n*. The SPC*n* option overrides the SPACE value.

Here are the default values for the new database file SPACE options:

| Database Files | SMALL | LARGE |
|---|---|---|
| Files 1, 3, and 4 | (TRK,(3,5)) | (TRK,(30,50)) |
| Files 2, 5, 6, 7, and 8 | (CYL,(1,5)) | (CYL,(10,50)) |

## Examples

ALLOC PERSONNEL, DISP=NEW:

ALLOC EMPLOYEE:

ALLOC FINANCE,FILES=ALL,VOL=SAS834,DISP=NEW:

ALLOC CARS,SPACE=LARGE,DISP=NEW:

ALLOC SALES,BLK5=23464,BLK6=23464,DISP=NEW:

## Rules

- You can issue one or more ALLOC commands at any time before opening a database with the NEW DATA BASE IS command or the DATA BASE NAME IS command.
- The ALLOC execution parameter controls dynamic allocation of database files. If ALLOC=YES, dynamic allocation is allowed. If ALLOC=TBL, only databases listed in the S2KDBCNT table can be accessed dynamically. ALLOC=NO disallows dynamic allocation. For details about the S2KDBCNT table, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.
- The FREE command enables you to deallocate database files that were dynamically allocated.
- In a Multi-User environment, the VARY console command allows the operator to vary databases offline and online. For more information about the VARY console command, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.
- For details about default data set names, space options, and so on that the software uses for dynamic allocations, see the discussion of the ALLOC command in *SYSTEM 2000 CONTROL Language, Version 12, First Edition*.

# APPLY

Applies recorded updates from the Keepfile to a restored database.

```
APPLY  |ALL               :
       |THROUGH CYCLE n
```

*n* is the last update cycle number to be applied to a saved database. The number must be greater than 0.

## Examples

```
APPLY ALL:

APPLY THRU CYCLE 55:
```

## Rules

- Abbreviation: THROUGH = THRU
- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- You must have exclusive use of the database to issue APPLY.
- Issue RESTORE before issuing APPLY.
- Do not update a database between RESTORE and APPLY commands.
- You can apply all update cycles or a range of update cycles.
- You cannot skip update cycles. (For example, you cannot apply cycles 20 through 29 and then cycles 32 through 35.)
- You can issue APPLY only if the cycle number in the Keepfile matches the current cycle number for the restored database.
- Only one Keepfile can be specified for each restored database. The Keepfile is the one specified in the most recent SAVE or RESTORE command for the database.
- You cannot issue APPLY for a damaged database.

## ASSIGN AUTHORITIES

Assigns, changes, or removes authorities for secondary passwords.

```
ASSIGN |authorities TO |component [, component] ... FOR |ALL PASSWORDS :
       |N              |ALL COMPONENTS                  |passwords
```

```
ASSIGN |authorities TO component THROUGH component  FOR |ALL PASSWORDS :
       |N                                               |passwords
```

| | |
|---|---|
| *authorities* | can be any of the following: R, U, and W in any combination, separated by commas; or N used alone.<br><br>R-authority allows retrieval of specific components.<br><br>U-authority allows updating of specific items or records.<br><br>W-authority allows specific items or records in where-clauses.<br><br>N-authority restricts knowledge of specific components and removes previously assigned authorities. |
| *component* | is a component name or component number. |
| *passwords* | is a list of one or more secondary passwords, separated by commas. |

### Examples

```
AS R TO C101 FOR ABLE:

ASSIGN W,U TO C7,C8,C202 FOR ALL PASSWORDS:

ASSIGN R TO C201 THRU C203 FOR BAKE:

AS R,W,U TO ALL COMPONENTS FOR +:

ASSIGN N TO GENDER, ETHNIC ORIGIN FOR +,CHAR:
```

## Rules

- Abbreviations:
  ASSIGN = AS
  COMPONENTS = COMPONENT
  PASSWORDS = PASSWORD
  THROUGH = THRU

- Only the master password holder can assign authorities.

- Assigning authority to a record does not give the same authority to its items.

- You cannot simply add a new authority to a set of existing authorities. Any time you change authorities, you need to specify all authorities. For example, if you want to add W-authority to an existing R-authority, both W and R must be specified.

- If you use the THROUGH option and the word THROUGH is part of a component name to be specified, use the component number.

- If you use the THROUGH option, component names or numbers must be listed in DESCRIBE order.

- You can assign any authority to a string or function, regardless of its actual use, in order to access the string or function. When the string or function is invoked, items and records within it must have the proper authorities for the current password.

## CHANGE PASSWORD

Changes the master password, the secondary passwords, and the DBA password.

```
CHANGE PASSWORD  old-password  TO  new-password  :
```

| | |
|---|---|
| *old-password* | is an existing password. |
| *new-password* | is a new password using 1 to 4 alphanumeric characters with no blanks. The password can also be a single special character or lowercase letter. See Appendix A for the list of special characters. |

### Examples

```
CHANGE PASSWORD ABLE TO BAK:
```

```
CH PASSWORD + TO $:
```

### Rules

- Abbreviation: CHANGE = CH
- Only the master password holder can change a password.
- All authorities associated with an old password are transferred to the new password.
- The new password must be unique within the database.

# CREATE INDEX

Changes items from non-key to key and creates an index for each.

```
[CREATE] INDEX items   :
```

| | |
|---|---|
| *items* | is a list of one or more non-key item names or component numbers, separated by commas. The maximum is 96 items. |

## Examples

```
CREATE INDEX C3:

INDEX C106, PROFICIENCY, MAJOR FIELD:
```

## Rules

- Only the master password holder can issue CREATE INDEX.
- You cannot repeat an item name or a component number.
- In a Multi-User environment, you do not need exclusive use of the database to create an index. (CREATE INDEX puts a global hold on the database to prevent access.)
- Issuing this command increments the definition number by one, but the database cycle number does not change.
- Issuing this command does not affect update logging.
- You cannot issue CREATE INDEX for a damaged database.

## DATA BASE NAME IS

Opens an active database.

```
[EXCLUSIVE]  DATA BASE NAME IS  database  :
```

| | |
|---|---|
| EXCLUSIVE | specifies exclusive use of the database; no other user can open the database until you close it. |
| *database* | is a name previously assigned to a database. It can have 1 to 16 characters. |

### Examples

```
DATA BASE NAME IS EMPLOYEE:

EXCL DBN IS STATUS:

DBN IS SALES:
```

### Rules

- Abbreviations:
  DATA BASE NAME = DBN
  EXCLUSIVE = EXCL

- After this command is executed, the QUEST processor is attached.

- The database does not have to be defined or have values for you to issue DATA BASE NAME IS.

- You can open only one database at a time. To open a different database that shares the current password, attach the CONTROL processor, then issue DATA BASE NAME IS again to specify the other database.

  To open another database with a different password, attach the CONTROL processor, issue USER to specify the different password, then issue DATA BASE NAME IS to specify the other database.

  In both cases, the software closes the first database and opens the next database, in that order.

- You can dynamically allocate existing database files with the ALLOC command in an SCF session. Or you can let SYSTEM 2000 software search for the files automatically. If the JCL or CLIST contains DD statements for the files, the software uses those allocations. For more information about dynamic allocation of existing database files, see the discussion of the ALLOC command in *SYSTEM 2000 CONTROL Language, Version 12, First Edition* and the discussions of database file allocation in the *SYSTEM 2000 Product Support Manual, Version 12, First Edition.*

# DBA PASSWORD IS

Assigns the DBA password so that the database administrator can perform administrative functions.

```
DBA PASSWORD IS  password  :
```

| | |
|---|---|
| *password* | is a word of 1 to 4 alphanumeric characters with no blanks. The password can also be a single special character or lowercase letter. See Appendix A for the list of special characters. |

## Examples

```
DBA PASSWORD IS FOX:

DBA PASSWORD IS ?:

DBA PASSWORD IS A5:
```

## Rules

- Only the master password holder can issue DBA PASSWORD IS.
- A database can have only one DBA password, and it must be unique among all passwords for the database.
- The DBA password holder can automatically issue any system-wide command and the CONTROL language KEEP command.
- The master password holder can authorize the DBA password holder to issue certain CONTROL language commands with ENABLE DBA FOR.
- The DBA password holder cannot issue commands that access data or any DEFINE language commands.

## ENABLE/DISABLE DBA FOR

Controls the DBA password authorization of commands chosen from a list of available commands.

```
|ENABLE  DBA FOR  |ALL COMMANDS :
|DISABLE          |command
```

*command* is any of the following:

| | |
|---|---|
| APPLY | RESET ROLLBACK |
| FULL PASSES | RESTORE |
| MULTIPLE HOLDS | ROLLBACK |
| PRINT SIZE | SAVE |
| RELEASE | SUSPEND |
| RELOAD | VALUES PADDING |
| REORGANIZE | |

### Examples

```
ENABLE DBA FOR ALL COMMANDS:

DISABLE DBA FOR RELEASE:

ENABLE DBA FOR PRINT SIZE, RELOAD, REORGANIZE:
```

### Rules

- Only the master password holder can issue this command.
- The ALL COMMANDS option authorizes or removes authorization for all commands on the list of available commands.
- The DBA password holder can automatically issue any system-wide command and the CONTROL language KEEP command.
- The DBA password holder cannot issue commands that access data or any DEFINE language commands.
- The DBA password is assigned with DBA PASSWORD IS.

## ENABLE/DISABLE MULTIPLE HOLDS

Allows PLEX users to acquire multiple local holds simultaneously on a database in a Multi-User environment, or prohibits the same.

```
ENABLE MULTIPLE HOLDS  :

DISABLE MULTIPLE HOLDS  :
```

### Examples

```
ENABLE MULTIPLE HOLDS:

DISABLE MULTIPLE HOLDS:
```

### Rules

- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- You must have exclusive use of a database to enable or disable multiple holds.
- Each database is initialized to DISABLE. The current setting remains in effect until the opposite command is issued.
- If the Rollback Log is enabled for a database, the MULTIPLE HOLDS option is automatically enabled.
- If the Rollback Log is disabled for a database, the MULTIPLE HOLDS option reverts to its previous setting.

## ENABLE/DISABLE ROLLBACK

Controls the Rollback Log to allow or prevent recovery of a database.

```
ENABLE ROLLBACK   :

DISABLE ROLLBACK   :
```

### Examples

```
ENABLE ROLLBACK:

DISABLE ROLLBACK:
```

### Rules

- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- You must have exclusive use of the database to enable or disable rollback.
- Each database is initialized to DISABLE. The current setting remains in effect until the opposite command is issued.
- For each database having the Rollback Log enabled, database File 8 must be allocated with either a DD statement or dynamic file allocation. (Tapes are not allowed for the Rollback Log.)
- If the Rollback Log is enabled, the LHOLD=YES execution option is ignored.
- You cannot issue this command for a damaged database.

## ENABLE/DISABLE ROUTINE

Enables or disables user exits.

```
|ENABLE   ROUTINE  names  FOR  EXITS  range  [, range] ...  :
|DISABLE
```

*names* is a list of 1 to 10 routine names, separated by commas. Each name is 1 to 8 characters and is designated by the installation systems programmer responsible for coding user exits.

*range* *nn* [THROUGH *mm*]

*nn* is an integer from 00 to 63, depending on how EXIT01 is coded. (You can enter a single-digit integer with or without the leading 0, that is, 3 or 03.)

*mm* is a positive integer greater than nn and less than or equal to 63, depending on how EXIT01 is coded. (You can enter a single-digit integer with or without the leading 0.)

### Examples

```
ENABLE ROUTINE A3, C4, Z8 FOR EXITS 09 THRU 12:

DISABLE ROUTINE ALL FOR EXITS 17 THRU 20:

ENABLE ROUTINE FIVE FOR EXIT 5:
```

### Rules

- Abbreviations:
  EXITS = EXIT
  THROUGH = THRU

- The user-exit feature must be installed to enable or disable user exits.

- For more information about user exits, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition.*

## ENTRY KEY IS

Invokes or cancels the Security by Entry feature, which restricts the access of secondary password holders to specific logical entries, based on a unique value for the specified entry key.

```
ENTRY KEY IS item   :
```

```
ENTRY KEY IS |ENTRY                 :
             |CO
             |user-defined-CO-name
```

| | |
|---|---|
| *item* | is a level 0 key item name or component number designated as the entry key. |
| ENTRY | cancels Security by Entry. |
| CO | cancels Security by Entry. |
| *user-defined-CO-name* | is an ENTRY record that has been renamed. Cancels Security by Entry. |

### Examples

```
ENTRY KEY IS C1:

ENTRY KEY IS ENTRY:

ENTRY KEY IS CO:
```

### Rules

- Only the master password holder can issue this command.
- After Security by Entry is specified, it is in effect for the database for all sessions.
- Each value for an Entry Key item must be unique for each logical entry.
- The secondary password holder must know at least one unique value for the Entry Key item to gain access to other data in a logical entry.
- When Security by Entry is in effect, you cannot issue TALLY unless you are the master password holder or your secondary password begins with a numeral.
- If the first character of a secondary password is a numeral, the software ignores Security by Entry. That is, the entire database is available for retrieval and update.
- Do not assign U-authority to an Entry Key item for a secondary password.

# FREE

Closes the specified database and deallocates Files 1 through 8.

```
FREE database :
```

| | |
|---|---|
| *database* | is the name of the database to be deallocated. It must be a valid database name of 1 to 16 characters. |

## Example

```
FREE EMPLOYEE:
```

## Rules

- The FREE command deallocates database files that were dynamically allocated within the SCF session.
- In a Multi-User environment, you must have exclusive use of the database or be the only user accessing it in order to deallocate the files.
- In a Multi-User environment, the VARY console operator command is available to deallocate database files that are in the S2KDBCNT table. VARY OFFLINE closes the database, deallocates the files, and flags the database as being offline. For more details about the VARY console command, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition.*
- The FREE execution parameter is also available. FREE=YES instructs SYSTEM 2000 software to deallocate the database files at the end of the session. FREE=NO prevents that automatic deallocation. Therefore, if FREE=NO, and you want the files to be deallocated, you must use the FREE command. For more details about setting the FREE execution parameter, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition.*
- Savefiles and Keepfiles that were dynamically allocated are deallocated automatically when a SAVE, RESTORE, APPLY, or KEEP command completes.

## FULL PASSES

Allows or prohibits complete passes through the database Data Table.

```
|[ALLOW] FULL PASSES  :
|[NO]
```

### Examples

```
NO FULL PASSES:

FULL PASSES:
```

### Rules

- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- Each database is initialized to ALLOW. The current setting remains in effect until the opposite command is issued.
- The setting affects all users in a Multi-User environment.

# INVALID PASSWORD IS

Deletes a secondary password or the DBA password.

```
INVALID PASSWORD IS password :
```

| | |
|---|---|
| *password* | is a valid secondary or DBA password that you want to delete. |

## Examples

```
INVALID PASSWORD IS ABLE:

INVALID PASSWORD IS +:

INVALID PASSWORD IS X9:
```

## Rules

- Only the master password holder can issue this command.
- The master password cannot be deleted.

## KEEP

Requests the transfer of updates from the Update Log to the Keepfile.

```
KEEP    :
```

### Example

```
KEEP:
```

### Rules

- During processing of KEEP, the database is in hold status to prevent updating.
- If the Rollback Log is enabled, KEEP causes a user synchpoint for Coordinated Recovery.
- You can issue KEEP for a damaged database.

## LIST DBA

Displays the DBA password.

```
LIST DBA  :
```

### Example

```
LI DBA:
```

### Rules

- Abbreviation: LIST = LI
- Only the master password holder can issue this command.
- The output is written to the Message File.

## LIST DBA AND COMMANDS

Displays the DBA password and commands that are authorized for the DBA password.

```
LIST DBA AND COMMANDS   :
```

### Example

```
LI DBA AND COMMANDS:
```

### Rules

- Abbreviation: LIST = LI
- Only the master password holder can issue this command.
- The output shows only the commands authorized with ENABLE DBA FOR.
- The output is written to the Message File.

## LIST PASSWORDS

Displays the secondary passwords.

```
LIST PASSWORDS   :
```

### Example

```
LI PASSWORDS:
```

### Rules

- Abbreviations:
  LIST = LI
  PASSWORDS = PASSWORD
- Only the master password holder can issue this command.
- The master password and the DBA password are not listed.
- Passwords appear in the order they were assigned.
- The output is written to the Message File.

## LIST PASSWORDS AND AUTHORITIES

Displays the secondary passwords and their authorities by component.

```
LIST PASSWORDS AND AUTHORITIES  :
```

### Example

```
LI PASSWORD AND AUTH:
```

### Rules

- Abbreviations:
  AUTHORITIES = AUTH
  LIST = LI
  PASSWORDS = PASSWORD
- Only the master password holder can issue this command.
- The master password and the DBA password are not listed.
- Passwords and authorities appear in the order they were assigned.
- At the intersection of each row and column of output, a three-character authority symbol represents R, U, and W (in that order). A period represents authorities not assigned. When periods are present for all authorities, N is the assigned authority.
- The output is written to the Message File.

# NEW DATA BASE IS

Assigns a name to a new database.

```
NEW DATA BASE IS  database  [/n]  :
```

| | |
|---|---|
| *database* | is a name to be assigned to a database. The database name can have 1 to 16 characters. The first seven characters are used for the DDname, which must be unique within either the MVS or CMS environment. |
| *n* | is an optional number from 1 to 10,000 specifying the maximum number of components (including the C0 component, strings, and functions) that can be defined. The default is 430. |

## Examples

```
NEW DATA BASE IS EMPLOYEE:

NDB IS EMPLOYEE/800:

NDB IS FREIGHT-CAR:
```

## Rules

- Abbreviation: NEW DATA BASE = NDB
- The USER command issued before this command assigns the master password.
- After this command is executed, the DEFINE processor is attached giving you exclusive use of the database.
- All characters in the standard character set can be used for the name except for these characters:

  /
  :
  =

- Use the RESTORE command if you need to change a database name.
- You can use the ALLOC command to allocate the new database files dynamically within an SCF session. Or you can allocate the new files with JCL or a CLIST. For more information about allocating database files dynamically, see the ALLOC command discussion in *SYSTEM 2000 CONTROL Language, Version 12, First Edition* and database file allocation discussions in the *SYSTEM 2000 Product Support Manual, Version 12, First Edition.*

## PRINT SIZE

Displays database size statistics.

```
PRINT SIZE  :
```

### Example

```
PR SIZE:
```

### Rules

- Abbreviation: PRINT = PR
- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- The output is written to the Message File.

# RELEASE

Renders the active database disk files unusable until the database is restored.

```
RELEASE   :
```

## Example

```
RELEASE:
```

## Rules

- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- You must have exclusive use of the database to issue RELEASE.
- Issue RELEASE before restoring a damaged database.
- After a database is released, the database files are still allocated. However, File 1 is reformatted, making the database unusable.
- If you want to save the database, be sure to issue SAVE before issuing RELEASE.
- Issuing RELEASE suspends update logging.
- Setting the execution parameter OPT009=YES forces the software to clear database Files 2 through 6 of a released database to binary zero. For details see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition.*

## RELOAD

Reconstructs the database files by unloading the database internally and then rebuilding the database tables.

```
RELOAD [[ordering-clause] where-clause]  :
```

| | |
|---|---|
| *ordering-clause* | specifies the order of selected data records. Use only level 0 items. A where-clause is required if an ordering-clause is specified. See **Ordering-Clause** on page 3-42 for details. |
| *where-clause* | selects specific data records. The SAME operator cannot be included. See **Where-Clause** on page 3-62 for details. |

### Examples

```
RELOAD:

RELOAD WHERE EMPLOYEE NUMBER LT 5075:

RELOAD ORDERED BY C1 WHERE C1 EXISTS:
```

### Rules

- Only the master password holder or the DBA password holder (with authorization) can issue this command. However, the DBA password holder cannot issue a RELOAD that contains a where-clause.
- You can issue RELOAD from either the CONTROL or the QUEST processor. In either case, the CONTROL processor is attached after execution.
- Reloading requires the same amount of scratch file space as loading the database.
- If you include a where-clause and a LIMIT command was issued previously, the number of records selected for reloading must conform to the specified limits.
- RELOAD resets the cycle number to zero at the beginning of the reload process and increments it by one when the reloading is finished. However, if a specified where-clause produces no qualified data records, the cycle number remains reset to zero.
- Issuing RELOAD suspends update logging.
- If the RECORD format option is in effect, only level 0 data records are reloaded.
- In a Multi-User environment, you must have exclusive use of the database to issue RELOAD.
- Do not issue RELOAD for a damaged database.

## REMOVE INDEX

Changes items from key to non-key and removes the index for each.

```
REMOVE INDEX  |[/LOSE/]  items  :
              |[/REUSE/]
```

| | |
|---|---|
| LOSE | specifies that removed Multiple Occurrence Table space is lost until restructuring, reorganizing, or reloading occurs. LOSE is the default. |
| REUSE | specifies that removed Multiple Occurrence Table space is placed on the reusable space lists. |
| *items* | is a list of one or more key item names or component numbers, separated by commas. The maximum is 96 items. |

### Examples

```
REMOVE INDEX C413:

RE INDEX /REUSE/ C423, POSITION TYPE:
```

### Rules

- Abbreviation: REMOVE = RE
- Only the master password holder can issue this command.
- Exclusive use is not required in a Multi-User environment, because REMOVE INDEX puts a global hold on the database to prevent access by anyone else.
- REMOVE INDEX increments the definition number by one, but the database cycle number does not change.
- This command does not affect update logging.
- You cannot repeat an item name or a component number.
- You cannot issue REMOVE INDEX for a damaged database.

## REORGANIZE

Compacts and reorders the index. Use this command when database entries become scattered due to incremental loads or many updates.

```
REORGANIZE  |DVT          :
            |MOT
            |ALL
            |INDEX
```

### Examples

```
REORGANIZE DVT:

REORGANIZE ALL:
```

### Rules

- Abbreviation: REORGANIZE = REORG
- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- Issuing REORGANIZE does not change update logging or the database cycle number.
- In a Multi-User environment, a global hold is placed on the database until the reorganization process is complete.
- You cannot issue REORGANIZE for a damaged database.
- Either ALL or INDEX reorganizes both the Distinct Values Table and Multiple Occurrence Table.

# RESET ROLLBACK AFTER

Controls the number of pages that are written to the Rollback Log and the extent of recovery that occurs in the event of a transaction or system failure.

```
RESET ROLLBACK AFTER |x  PERCENT                    :
                     |y  SYNCHPOINTS
                     |x  PERCENT, y SYNCHPOINTS
```

*x* is the percentage of primary space that can be written to the Rollback Log before it is reset. Zero percent clears the database and resets the log at the start of each synchpoint. One hundred percent resets the log after primary allocation is used. The default is 50.

*y* is the number of user synchpoints that can occur between clearing the database and resetting the Rollback Log. y can be any number from 1 to 999999. The default is 999999.

## Examples

```
RESET ROLLBACK AFTER 80 PCT, 20000 SYNCHPOINTS:

RESET ROLLBACK AFTER 75 PERCENT:

RESET RB AFTER 15000 SPTS:
```

## Rules

- Abbreviations and synonyms:
  PERCENT = PCT
  ROLLBACK = RB
  SYNCHPOINTS = SYNCHPOINT, SPTS, or SPT
- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- You must have exclusive use of the database to issue RESET ROLLBACK AFTER.
- The database is cleared and the Rollback Log is reset after either x-percent or y-synchpoints are reached.
- You can issue this command regardless of whether the Rollback Log is enabled.
- You cannot issue RESET ROLLBACK AFTER for a damaged database.

## RESTORE

Copies a database from a Savefile onto the active database files. This command can also activate update logging and change the name of a database.

```
| RESTORE old-database [= new-database] [FROM Savefile-identifier]

|    [/] [Keepfile-identifier]  :
```

| | |
|---|---|
| *old-database* | is the name of an existing database. |
| *new-database* | is the new database name using 1 to 16 characters. The first seven characters are used for the DDname, which must be unique within the MVS or CMS environment. |
| *Savefile-identifier* | [*volume-list*] [UNIT=*unit*] [DSN=*data-set-name*] [DS=*sequence-number*] If specified, *volume-list* must be the first parameter. UNIT, DSN, and DS can be specified in any order. |
| *volume-list* | *volume* [,*volume-list*] You can specify up to 10 volumes for the Savefile. |
| *volume* | is a standard volume identification label of up to six characters. If you do not specify a volume and you specify UNIT=TAPE, the default is a scratch tape. |
| *unit* | is a device unit of 1 to 8 characters, such as TAPE, RIO, VIO. The default is disk (SYSDA). |
| *data-set-name* | is a data set name of up to 44 characters. Use quotes for a fully qualified name. If you do not specify the data set name, the default is the prefix followed by the first seven characters of the database name (with embedded blanks replaced by Xs) and a suffix of S for the Savefile and K for the Keepfile. |
| *sequence-number* | is a data set sequence number for the Savefile or Keepfile. The number can be 1 to 255. This number applies only to the first volume of a multi-volume data set. Recording continues at the beginning of additional volumes (DS=1). The default is 1. |
| / | activates update logging. |
| *Keepfile-identifier* | [*volume*] [UNIT=*unit*] [DSN=*data-set-name*] [DS=*sequence-number*] Only one volume can be specified for the Keepfile. If specified, *volume* must be the first parameter. UNIT, DSN, and DS can be specified in any order. |

## Examples

```
RESTORE EMPLOYEE:

RESTORE BOOKS FROM SAS290:

RESTORE BOOKS FROM DSN=BOOKSAVE:

RESTORE LIBRARY = STACKS:

RESTORE CARS /:
```

## Rules

- After RESTORE is executed, the QUEST processor is attached.
- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- If update logging was active when the database was saved, it remains active, even if you do not specify it.
- Issue SAVE and then RELEASE before issuing RESTORE.
- Update logging cannot be activated if you are changing the database name.
- Changing a database name suspends update logging.
- The Savefile is a QSAM file. The Keepfile is a BSAM file.

  DDnames for the Savefiles and Keepfiles are similar to the DDnames for database Files 1 through 6. They consist of the first seven characters of the database name (with blanks replaced by Xs) plus a suffix of S for a Savefile and K for a Keepfile. TAPES2K and KEEPFILE DDnames are not valid with Version 12 and subsequent releases.

## SAVE

Copies a database to save it. This command can also activate or suspend update logging.

```
SAVE [DATA BASE] [ON Savefile-identifier] [/] [Keepfile-identifier]  :
```

| | |
|---|---|
| *Savefile-identifier* | [*volume-list*] [UNIT=*unit*] [DSN=*data-set-name*] [DS=*sequence-number*] If specified, *volume-list* must be the first parameter. UNIT, DSN, and DS can be specified in any order. |
| *volume-list* | *volume* [,*volume-list*] You can specify up to 10 volumes for the Savefile. |
| *volume* | is a standard volume identification label of up to six characters. If you do not specify a volume and you specify UNIT=TAPE, the default is a scratch tape. |
| *unit* | is a device unit of 1 to 8 characters, such as TAPE, RIO, VIO. The default is disk (SYSDA). |
| *data-set-name* | is a data set name of up to 44 characters. Use quotes for a fully qualified name. If you do not specify the data set name, the default is the prefix followed by the first seven characters of the database name (with embedded blanks replaced by Xs) and a suffix of S for the Savefile and K for the Keepfile. |
| *sequence-number* | is a data set sequence number for the Savefile or Keepfile. The number can be 1 to 255. This number applies only to the first volume of a multi-volume data set. Recording continues at the beginning of additional volumes (DS=1). The default is 1. |
| / | activates update logging. |
| *Keepfile-identifier* | [*volume*] [UNIT=*unit*] [DSN=*data-set-name*] [DS=*sequence-number*] Only one volume can be specified for the Keepfile. If specified, *volume* must be the first parameter. UNIT, DSN, and DS can be specified in any order. |

### Examples

```
SAVE:
```
*(Update logging is not specified.)*

```
SAVE /:
```
*(Update logging is activated.)*

```
SAVE DATA BASE ON ML001/ML002:
```

## Rules

- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- If update logging was previously active but you omit the slash, update logging is suspended.
- If the Rollback Log is enabled, a SAVE command creates a user synchpoint for Coordinated Recovery.
- If update logging is active and there have been updates, issue KEEP before issuing the SAVE command.
- In a Multi-User environment, this command does not require exclusive use of the database.
- You can SAVE a damaged database, but the damaged condition exists in the saved copy of the database.
- The Savefile is a QSAM file. The Keepfile is a BSAM file.

  DDnames for the Savefiles and Keepfiles are similar to the DDnames for database Files 1 through 6. They consist of the first seven characters of the database name (with blanks replaced by Xs) plus a suffix of S for a Savefile and K for a Keepfile. TAPES2K and KEEPFILE filerefs are not valid with Version 12 and subsequent releases.

## SUSPEND

Stops the recording of database updates.

```
SUSPEND   :
```

### Example

```
SUSPEND:
```

### Rules

- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- When update logging has been activated, SUSPEND makes the Update Log unavailable.

# USER

Accesses SYSTEM 2000 software and gives the password for the database to be opened, either setting the master password for a new database or verifying the password for an existing database.

```
USER, password :
```

| | |
|---|---|
| *password* | is a word of 1 to 4 alphanumeric characters with no blanks. The password can also be a single special character or lowercase letter. See Appendix A for the list of special characters. |

## Examples

```
USER, DEMO:  DATA BASE NAME IS EMPLOYEE:

USER, +:  NEW DATA BASE NAME IS OIL SPILLS:
```

## Rules

- USER, followed by NEW DATA BASE IS, assigns the master password for a database.
- To open an existing database, the password must be the master password, a secondary password, or the DBA password.
- To open a different database that shares the current password, attach the CONTROL processor, then issue DATA BASE NAME IS again to specify the other database.
- To open a different database that does not share the current password or to use a different password for the same database, attach the CONTROL processor, then reissue USER, followed by DATA BASE NAME IS.

## VALID PASSWORD IS

Assigns secondary passwords for a database.

```
VALID PASSWORD IS  password  :
```

| | |
|---|---|
| *password* | is a word of 1 to 4 alphanumeric characters with no blanks. The password can also be a single special character or lowercase letter. See Appendix A for the list of special characters. |

### Examples

```
VALID PASSWORD IS ABLE:

VALID PASSWORD IS +:

VALID PASSWORD IS X9:
```

### Rules

- Only the master password holder can issue this command.
- A maximum of 200 secondary passwords is allowed per database.
- Assign only one secondary password at a time. Each secondary password must be unique within the database.
- You must assign access authorities for the secondary passwords with the ASSIGN authorities command.
- If the first character of a password is a numeral, the software ignores Security by Entry (if in effect) for that password holder. That is, the entire database is available for retrieval and update.

Chapter 2

# DEFINE LANGUAGE COMMANDS

## CHANGE COMPONENT

Changes the number, name, or description (or any combination) of a component.

```
CHANGE  old-number  TO  new-number  :
```

```
CHANGE  old-number  TO  |new-number *  new-name  :
                        |old-number
```

```
CHANGE  old-number  TO  |new-number *  |new-name (new-description) :
                        |old-number    |old-name
```

| | |
|---|---|
| *old-number* | is the number of an existing component, without the C. |
| *new-number* | is a new number for a component, without the C. It can be from 1 to 9999, and it cannot duplicate any existing number. |
| *new-name* | is a new name for a component using 1 to 250 characters. It cannot duplicate any existing name, and it cannot contain reserved words or characters. See Appendix B for the list of reserved words and characters. |
| *old-name* | is the name of an existing component. |
| *new-description* | defines a new description for a component. Changes to record inclusion are not allowed. You must specify either a new name or the old name when specifying a new description. |

### Examples

```
CHANGE 400 TO 500:

CHANGE 0 TO 0*EMPLOYEE:

CH 201 TO 210*SKILLS (CHAR X(24) IN 200 WITH MANY FUTURE
   OCCURRENCES):

CHANGE 2000 TO 2000*FUNA(FUNCTION $((COUNT C600)/*1*)$):
```

## Rules

- Abbreviation: CHANGE = CH
- If you do not include the old name, only the component number is changed.
- If you do not include a new description, only the component number, the name, or both are changed.
- If you are changing any part of a component description, the new description must include all specifications, or the system will use the defaults.
- For record C0, you can change only the component name.
- All constraints on defining new components apply to changing components.
- When you change a schema record number, the software also changes the IN specification record number for its items and descendant records.
- You can change non-key items to key or key items to non-key.
- You cannot change a parent-child record relationship or the record membership of an item.
- You cannot change a component type (for example, item to record). However, you can change a string to a function or a function to a string.
- Most changes to an item type or picture are acceptable as long as there are no data in the record (and no reusable space for the record) that the item belongs to. However, REAL, DOUBLE, and UNDEFINED items cannot be changed to other item types and vice versa.
- If data exist for the item, you can change the following:

  INTEGER to DECIMAL or MONEY
  DECIMAL to INTEGER or MONEY
  MONEY to INTEGER or DECIMAL
  TEXT to CHARACTER
  CHARACTER to TEXT
  DATE to INTEGER, DECIMAL, or MONEY

- If data exist in the record that an item belongs to, you cannot change the picture of DECIMAL, MONEY, or INTEGER items. However, you can move the decimal point.
- If data exist in the record that an item belongs to, you can change the picture for CHARACTER and TEXT items. However, the picture must be at least X(4). The picture for an UNDEFINED item cannot be changed.
- If a DATE item is changed to a numeric item, the picture must have seven digits.

## DELETE

Deletes components and their data from a database definition.

```
DELETE   |COMPONENTS          number  |[, number] ...    :
         |SCHEMA COMPONENTS           |[THROUGH number]
         |STRINGS
         |FUNCTIONS
```

*number* is the number of an existing component, without the C.

### Examples

```
DELETE COMPONENT 100:

DELETE COMPONENTS 410, 408, 412:

DELETE STRING 4000:

DELETE COMPONENTS 600 THRU 605:
```

### Rules

- Abbreviations and synonyms:
  COMPONENTS = COMPONENT
  FUNCTIONS = FUNCTION
  STRINGS = STRING
  THROUGH = THRU or TO
- You can delete individual components or a range of components.
- Each keyword designates one or more types of components: COMPONENT accepts record, item, string, and function numbers, SCHEMA COMPONENT accepts record and item numbers, STRING accepts string numbers only, and FUNCTION accepts function numbers only.
- If you specify THROUGH (for example, 600 THROUGH 605), the first number must precede the second number in DESCRIBE output.
- Components separated by commas can be in any order.
- If you delete a schema record, all member items and descendant records are also deleted.

## ENABLE/DISABLE EXECUTION

Allows the scanning of commands while allowing or preventing the execution of those commands.

```
ENABLE EXECUTION  :
DISABLE EXECUTION  :
```

### Examples

```
ENABLE EXECUTION:

DISABLE EXECUTION:
```

### Rules

- At the beginning of each Self-Contained Facility session, execution is enabled.
- You can enable or disable execution from the DEFINE, QUEST, or QUEUE processor. Commands in all three processors are affected, regardless of which processor the command is issued from.

## FUNCTION DEFINITION

A function stores an arithmetic expression. Specify the component number and name and arithmetic expression, using the syntax below. This type of function is referred to as a stored function.

```
number * name (|[DECIMAL] FUNCTION delimiter (expression) delimiter) :
               |[INTEGER]
               |[MONEY]
               |[REAL]
               |[DOUBLE]
               |[DATE]
```

| | |
|---|---|
| *number* | is a component number from 1 to 9999. |
| *name* | is a component name using 1 to 250 characters. It cannot be a reserved word or character or the current system separator. See Appendix B for the list of reserved words and characters. |
| DECIMAL<br>INTEGER<br>MONEY<br>REAL<br>DOUBLE<br>DATE | specifies the function type. The default is DECIMAL. |
| *delimiter* | is a single special character to be used as the expression delimiter. The delimiter must appear at both ends of the function expression and cannot appear within the expression. It cannot be the current system separator or command terminator. See Appendix A for the list of special characters. |
| *expression* | defines an arithmetic expression containing one or more item names or component numbers, constants, system functions (AVG, COUNT, MAX, MIN, SUM, SIGMA), arithmetic operators (+, -, *, /!), other functions or strings, the system string *FTODAY*, parameters, parenthetical arithmetic expressions, or Collect File items called by retrieval commands. |

### Examples

```
3004* YRS EMPLOYED (INTEGER FUNCTION
    $ ((END DATE-START DATE)/365.25)$):

3050* DEDUCTION-ADJ (MONEY FUNCTION/(C114 - 5.50)/):
```

## Rules

- Abbreviations and synonyms:
  DECIMAL = DEC
  INTEGER = INT
  REAL = FLOAT
  DOUBLE = DOUBLE PRECISION

- Each component must have a unique name and number.

- Components do not have to be defined in any order.

- Functions are included in the maximum number of components specified with NEW DATA BASE IS.

- Each system function within a stored function must be enclosed by parentheses.

- Functions can be simple or parametric. Parametric functions contain parameters, whose values must be supplied when the function is invoked. Simple functions have no parameters and are issued by their component name or component number.

- You can specify up to 31 levels of nesting, 31 parameters, any combination of the two yielding a total of 31, or a maximum of 1200 characters of function expansion and/or parameters. Parameters can be supplied within a function definition calling another string or function with parameters.

- You invoke a function in the action-clause of QUEST retrieval and update commands or in a REPORT action-clause. See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for information about invoking functions.

## ITEM DEFINITION

A schema item represents a class of values. You specify the component number and name, key status, item type, item picture, record membership, and padding, using the following syntax.

```
number  *  name  ([NON-KEY]  |CHARACTER [picture]  [IN sr-number]
                             |TEXT
                             |UNDEFINED
                             |INTEGER
                             |DECIMAL
                             |MONEY
                             |REAL
                             |DOUBLE
                             |DATE

     [WITH  |NO  FUTURE OCCURRENCES] )  :
            |FEW
            |SOME
            |MANY
```

| | |
|---|---|
| *number* | is a component number from 1 to 9999. |
| *name* | is a component name using 1 to 250 characters. It cannot be a reserved word or character or the current system separator. See Appendix B for the list of reserved words and characters. |
| NON-KEY | specifies non-key processing. The default is KEY. |
| *picture* | is the picture for data to be entered. |

CHARACTER default is X(7) with a maximum of 250.
TEXT default is X(7) with a maximum of 250.
UNDEFINED default is X(4) with a maximum of 250; no overflow.
INTEGER default is 9(7) with a maximum of 15.
DECIMAL default is 9(6).9(2).
MONEY default is 9(6).9(2) and can be preceded by $.
REAL never needs a specified picture.
DOUBLE never needs a specified picture.
DATE never needs a specified picture.

Note: For DECIMAL and MONEY items, if the picture is of the format 9(n).9(m), then n and m must each be greater than or equal to zero and less than 11. Their sum must be a positive integer less than 16.

| | |
|---|---|
| *sr-number* | is the record component number (previously defined) that the new item belongs to. Do not include the C. The default is 0. |
| NO<br>FEW<br>SOME<br>MANY | specifies future occurrences for padding. The default is NO. |

## Examples

```
1* EMPLOYEE NUMBER (INTEGER NUMBER 9999):
2* LAST NAME (CHAR X(20) WITH FEW FUTURE OCCURRENCES):
4* HIRE DATE (DATE):
11* ACCRUED VACATION (NON-KEY DECIMAL NUMBER 999.99):
303* COMMENT (NON-KEY TEXT X(15) IN 300):
124* GROSS PAY (NON-KEY MONEY $9(4).9(2) IN 120):
511* SAMPLE (UNDEF X(12) IN 500):
513* FPS (NON-KEY REAL IN 500):
514* FPD (DOUBLE IN 500):
```

## Rules

- Abbreviations and synonyms:
  CHARACTER = CHAR
  DECIMAL = DEC
  DOUBLE = DOUBLE PRECISION
  INTEGER = INT
  NON-KEY = NK
  REAL = FLOAT
  UNDEFINED = UNDEF

- You must specify an item type.

- All components must have unique names and numbers.

- Items do not have to be defined in any order. [IN *sr-number*] determines an item's record membership. However, the parent record must be defined before its items or descendant records.

- Items are included in the maximum number of components specified with NEW DATA BASE IS.

- If CHARACTER or TEXT values exceed the picture, those items must have at least a 4-character picture. Numeric and UNDEFINED items cannot overflow.

- The pictures of INTEGER, DECIMAL, and MONEY items do not need a position for a + or - sign.
- Only key items can have padding.
- The order in which you define items determines their top-to-bottom order within a record.
- For clarity, you can specify the keyword NUMBER after INTEGER, DECIMAL, MONEY, REAL, and DOUBLE items. (You can abbreviate NUMBER as NUMB or NUM.)

# MAP

Processes DEFINE commands and places the results in the database definition.

```
MAP  [[ordering-clause]  where-clause]  :
```

| | |
|---|---|
| *ordering-clause* | specifies the order of selected data records. Use only level 0 items. A where-clause is required if an ordering-clause is specified. See **Ordering-Clause** on page 3-42 for details. |
| *where-clause* | selects specific data records. The SAME operator cannot be included. See **Where-Clause** on page 3-62 for details. |

## Examples

```
MAP:

MAP WHERE HIRE DATE GT 01/15/90:

MAP ORDERED BY LAST NAME WHERE HIRE DATE GT 01/15/90:
```

## Rules

- Use a where-clause and an ordering-clause only when a database is to be restructured.
- If the RECORD format option is in effect and you issue a MAP command that causes restructuring, only level 0 records are retained. Descendant records are lost.
- If a where-clause is included and LIMIT is in effect, only the number of selected data records within the specified limits are honored by MAP.
- After you issue this command, the software changes to the QUEST processor.

## RECORD DEFINITION

A schema record represents a class of data records. You specify the component number and name and the parent-child relationships between records, using the following syntax.

```
number  *  name  (SCHEMA RECORD  [IN sr-number])  :
```

| | |
|---|---|
| *number* | is a component number from 1 to 9999. |
| *name* | is a component name using 1 to 250 characters. It cannot be a reserved word or character or the current system separator. See Appendix B for the list of reserved words and characters. |
| *sr-number* | is the record component number (previously defined) that the new record belongs to. Do not include the C. The default is 0. |

### Examples

```
100* POSITION WITHIN COMPANY (RECORD):
110* SALARY WITHIN POSITION (RECORD IN 100):
130* ADDITIONAL INFORMATION (SR IN 100):
```

### Rules

- Abbreviations: SCHEMA RECORD = RECORD or SR
- All components must have unique names and numbers.
- You must define a parent record before you define its descendant records or items. Descendant records do not have to be defined in any order.
- Records are included in the maximum number of components specified with NEW DATA BASE IS.
- You can define 32 levels of schema records (including level 0).
- [IN *sr-number*] is required except for level 1 records.
- The order in which you define siblings determines their left-to-right order in DESCRIBE output.

# RENUMBER

Renumbers components in a database definition.

```
RENUMBER  [STARTING WITH  number  [INCREMENTING BY  n]]  :
```

*number* is a positive number from 1 to 9999 to be assigned to the first component in DESCRIBE order after C0. The default is 1.

*n* specifies the increment for renumbering. It can be a number from 1 to 9999. You cannot use 0, and the default is 1.

## Examples

```
RENUMBER:

RENUMBER STARTING WITH 100:

RENUMBER STARTING WITH 100 INCREMENTING BY 10:
```

## Rules

- You cannot renumber the C0 record.
- No component number in a definition can be greater than 9999.
- Components are renumbered consecutively in DESCRIBE order. Schema components are renumbered first, then strings, then functions.
- When a record number changes, the IN specification for all member items and descendant records automatically changes, too.
- RENUMBER does not affect component numbers within stored strings and functions. Use the DEFINE language CHANGE command to change those numbers.

## STOP AFTER SCAN IF ERRORS OCCUR

Prevents mapping of DEFINE commands if errors occur. You can then edit the commands to correct errors. If no errors are encountered, mapping occurs.

```
STOP AFTER SCAN IF ERRORS OCCUR  :
```

### Example

```
STOP AFTER SCAN IF ERRORS OCCUR:
```

### Rules

- The default is that processing continues. It is set at the beginning of each Self-Contained Facility session.
- If an error is encountered after you issue this command, the software changes to the QUEST processor without mapping the definition.
- When both STOP AFTER SCAN IF ERRORS OCCUR and DISABLE EXECUTION are issued, DISABLE EXECUTION prevails.
- You can issue this command from the DEFINE, QUEST, or QUEUE processor. Commands in all three processors are affected, regardless of which processor STOP IF is issued from.

## STRING DEFINITION

A string stores a SYSTEM 2000 command, one part of a command, or a series of commands. Specify the component number and name and command stream, using the syntax below. Strings are sometimes referred to as stored strings.

---

*number* * *name* (STRING *delimiter* *command-stream* *delimiter*) :

---

| | |
|---|---|
| *number* | is a component number from 1 to 9999. |
| *name* | is a component name using 1 to 250 characters. It cannot be a reserved word or character or the current system separator. See Appendix B for the list of reserved words and characters. |
| *delimiter* | is a single special character to be used as the command stream delimiter. The delimiter must appear at both ends of the command stream and cannot appear within the command stream. It cannot be the current system separator or command terminator. See Appendix A for the list of special characters. |
| *command-stream* | defines a SYSTEM 2000 command stream using 1 to 250 characters. The command stream can consist of partial commands, one command, or several commands. It also can contain parameters, calls to other strings or functions, or system strings (*NOW*, *FTODAY*, *TODAY*, *DATA*). |

### Examples

```
1100* EMP-NO (STRING / PRINT C1: /):

1000* LIST OF DATES (STRING $ LIST C1, C4, OB C4
     WH HIRE DATE GT 01/01/91: $):

1050* LATEST POSITION (STRING $ C101 EXISTS AT 1 $):
```

## Rules

- Each component must have a unique name and number.
- Strings do not have to be defined in any order.
- Strings are included in the maximum number of components specified with NEW DATA BASE IS.
- Strings can be simple or parametric. Parametric strings contain parameters, whose values are supplied when you invoke the string. Simple strings have no parameters and are invoked by their component name or component number.
- You can specify up to 31 levels of nesting, 31 parameters, any combination of the two yielding a total of 31, or a maximum of 1200 characters of string expansion and/or parameters. Parameters can be supplied within a string definition calling another string or function with parameters.
- You can invoke a string in one of three situations: by itself, in a retrieval clause, in the where-clause of a QUEST or QUEUE retrieval or update command. See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for information about invoking strings.

Chapter 3

# QUEST LANGUAGE COMMANDS

# ADD

Replaces nulls with values in existing data records.

```
ADD  item  full-trace  EQ  |value *   :
                           |PREVIOUS
```

```
ADD  item  full-trace  =  |item                      :
                          |* function *
                          |* function (parameters)
                          |(adhoc-function)
                          |system-function
```

```
ADD  item  [partial-trace]  EQ  |value *  where-clause  :
                                |PREVIOUS
```

```
ADD  item  [partial-trace]  =  |item                 where-clause  :
                               |* function *
                               |* function (parameters)
                               |(adhoc-function)
                               |system-function
```

```
ADD  record  full-trace  EQ  |value-stream END * :
                             |PREVIOUS
```

```
ADD  record  [partial-trace]  EQ  |value-stream END * where-clause:
                                  |PREVIOUS
```

| | |
|---|---|
| *item* | is a schema item name or component number. |
| *full-trace* | is a list of one or more positive integers or 0, each preceded by the system separator (for example, *1*4*0). In a left-to-right scan, each integer designates the position of the child record under its parent at each level. The rightmost integer designates the level 0 position. Zero means last. |
| *value* | is a literal value or a system string (*NOW*, *TODAY*, *FTODAY*). |
| PREVIOUS | repeats the value stream or the item value in the preceding update command. |
| *function* | is a stored function name or component number. |
| *parameters* | are the values that replace the numbered parameters in the stored function. See **Function Definition** on page 2-6 for details. |

| | |
|---|---|
| *adhoc-function* | is a mathematical expression enclosed in parentheses. See **Ad Hoc Function** on page 3-6 for details. |
| *system-function* | allows you to obtain simple arithmetic statistics about values stored in a database. System functions include: AVG, COUNT, MAX, MIN, SIGMA, and SUM. See **System Function** on page 3-56 for details. |
| *partial-trace* | is a trace in which the last integer designates a position other than level 0. |
| *record* | is a schema record name or component number. |
| *value-stream* | is a series of item numbers and values to be added, with each item number followed by its associated value. The system separator must follow each item number and each value. The item numbers do not have to be in DESCRIBE order. |
| *where-clause* | selects specific data records. See **Where-Clause** on page 3-62 for details. |

## Examples

```
AD C412 EQ BS* WH C1 = 1005:

ADD EDUCATION EQ 413*01/07/85*414*ENGLISH*415*MARKETING*END*
   WH C2 EQ BROOKS:

ADD C126 = (C124 - C114) WH C1 EXISTS AND C121 EQ 06/30/91:

ADD C105 = (*TODAY*) WH C2 = GIBSON:
```

## Rules

- Abbreviation: ADD = AD
- Use ADD only to add data to existing data records that are occurrences of a single schema record.
- If you need to add multiple values, all values must be occurrences of items in the specified record.
- If you use the syntax 'ADD item = item', the item types must be compatible, that is, both dates, both numeric, or both nonnumeric. For numeric types, the second item assumes the picture of the first item. Also, if the value of the second item is too large for the picture of the first item, an error message appears. If the item name for either includes the equals sign, use the component number for the item.

  Also for the 'ADD item = item' syntax, if one item is type UNDEFINED, the other item must also be type UNDEFINED. That is, you can add only UNDEFINED values to other UNDEFINED items. You cannot add CHARACTER or TEXT values to UNDEFINED items.
- If you use the '=' syntax, the item, stored function, system function, or ad hoc function must be in the path of the target record (the record containing the specified item). If the system function includes a by-phrase, the record specified in the by-phrase must be in the path of the target record.
- You must have U-authority for components specified in the action-clause of an ADD command.
- The command referenced by PREVIOUS cannot contain errors.
- When specifying PREVIOUS, the item or record preceding EQ must be the same item or record as in the previous update command.
- You cannot use Collect File items in an ADD action-clause.
- You cannot issue ADD for a damaged database.

## AD HOC FUNCTION

Specifies mathematical operations to be performed on data. The following operands are available:

| | |
|---|---|
| * | multiplication |
| / | division |
| + | addition |
| - | subtraction |

### Examples

```
PRINT (C114 + C126) WHERE C1 EQ 1010:
```

```
PR (((MIN C126) + (MAX C126))/2):
```

### Rules

- You can specify an ad hoc function in the action-clause of a retrieval or update command or in a by-clause.
- An ad hoc function must be enclosed in parentheses. Parentheses control the order of operations; the innermost pair of parentheses encloses the first operation performed.
- An ad hoc function can contain system functions (enclosed in parentheses), the system string *FTODAY*, stored functions, numeric data items (in the same path), constants, and dates. Specify dates in the form MM.DD.YY to avoid confusing the slashes with the division operand.
- Multiplication and division operations are performed before addition and subtraction operations
- If both multiplication and division operations are specified, the software does the calculations on the left and works toward the right.
- You can nest ad hoc functions.
- You can use E-notation for constants.
- If you include more than one item type in an ad hoc function, the software determines the output format by the highest item type, going from highest to lowest in this order: DOUBLE, REAL, DECIMAL, MONEY, INTEGER, and DATE.

- You can store an ad hoc function in the definition by assigning a component name, number, and type. See **Function Definition** on page 2-6 for details.

- You can include Collect File items in an ad hoc function with retrieval commands but not with update commands.

- You can include component names in an ad hoc function as long as they do not include slashes, plus signs, or minus signs.

- If one or more values for specified items are null, no output is produced unless the NULL or ZERO format option is in effect. If NULL is set, the output for the missing values is -NULL-. If ZERO is set, the missing values are replaced with zeros, and the calculations are made and displayed. Nulls for dates are not converted to zero.

## ASSIGN

Replaces values or nulls with values in existing data records.

```
ASSIGN  item  full-trace  EQ  |value *  :
                              |PREVIOUS
```

```
ASSIGN  item  full-trace  =  |item                      :
                             |* function *
                             |* function (parameters)
                             |(adhoc-function)
                             |system-function
```

```
ASSIGN  item  [partial-trace]  EQ  |value *  where-clause :
                                   |PREVIOUS
```

```
ASSIGN  item  [partial-trace]  =  |item            where-clause :
                                  |* function *
                                  |* function (parameters)
                                  |(adhoc-function)
                                  |system-function
```

```
ASSIGN  record  full-trace  EQ  |value-stream END * :
                                |PREVIOUS
```

```
ASSIGN  record  [partial-trace]  EQ  |value-stream END * where-clause :
                                     |PREVIOUS
```

| | |
|---|---|
| *item* | is a schema item name or component number. |
| *full-trace* | is a list of one or more positive integers or 0, each preceded by the system separator (for example, *1*4*0). In a left-to-right scan, each integer designates the position of the child record under its parent at each level. The rightmost integer designates the level 0 position. Zero means last. |
| *value* | is a literal value or a system string (*NOW*, *TODAY*, *FTODAY*). |
| PREVIOUS | repeats the value stream or the item value in the preceding update command. |
| *function* | is a stored function name or component number. |
| *parameters* | are the values that replace the numbered parameters in the stored function. See **Function Definition** on page 2-6 for details. |

| | |
|---|---|
| *adhoc-function* | is a mathematical expression enclosed in parentheses. See **Ad Hoc Function** on page 3-6 for details. |
| *system-function* | allows you to obtain simple arithmetic statistics about values stored in a database. System functions include: AVG, COUNT, MAX, MIN, SIGMA, and SUM. See **System Function** on page 3-56 for details. |
| *partial-trace* | is a trace in which the rightmost integer designates a position other than level 0. |
| *record* | is a schema record name or component number. |
| *value-stream* | is a series of item numbers and values to be assigned, with each item number followed by its associated value. The system separator must follow each item number and each value. The item numbers do not have to be in DESCRIBE order. |
| *where-clause* | selects specific data records. See **Where-Clause** on page 3-62 for details. |

## Examples

```
AS OFFICE-EXTENSION EQ 410 XT369* WH C1 = 1120:

AS C14 EQ 2311 HANSFORD* WH C3 EQ MOLLY I. AND C2 EQ GIBSON:

ASSIGN JOB SKILLS EQ 201*RUSSIAN*202*EXCELLENT*203*15*END*
   WH C2 EQ BOWMAN:
```

## Rules

- Abbreviation: ASSIGN = AS
- Use ASSIGN only to replace data in existing data records that are occurrences of a single schema record.
- If you specify a record and if any original data are to be retained, those data must be included as a part of the value stream.
- If you need to assign multiple values, all values must be occurrences of items in the specified record.
- If you use the syntax 'ASSIGN item = item', the item types must be compatible, that is, both dates, both numeric, or both nonnumeric. For numeric types, the second item assumes the picture of the first item. Also, if the value of the second item is too large for the picture of the first item, an error message appears. If the item name for either includes the equals sign, use the component number for the item.

  Also for the 'ASSIGN item = item' syntax, if one item is type UNDEFINED, the other item must also be type UNDEFINED. That is, you can assign only UNDEFINED values to other UNDEFINED items. You cannot assign CHARACTER or TEXT values to UNDEFINED items.
- If you use the '=' syntax, the item, stored function, system function, or ad hoc function must be in the path of the target record (the record containing the specified item). If the system function includes a by-phrase, the record specified in the by-phrase must be in the path of the target record.
- You must have U-authority for components specified in the action-clause of an ASSIGN command.
- The command referenced by PREVIOUS cannot contain errors.
- When specifying PREVIOUS, the item or record preceding EQ must be the same item or record as in the previous update command.
- You cannot use Collect File items in an ASSIGN action-clause.
- You cannot issue ASSIGN for a damaged database.

# ASSIGN TREE

Replaces existing data trees (including the data) with new data trees.

```
ASSIGN TREE record  full-trace  EQ  |value-stream END* :
                                    |PREVIOUS
```

```
ASSIGN TREE record [partial-trace]  EQ |value-stream END* where-clause :
                                       |PREVIOUS
```

| | |
|---|---|
| **record** | is a schema record name or component number; the target record. |
| *full-trace* | is a list of one or more positive integers or 0, each preceded by the system separator (for example, *1*4*0). In a left-to-right scan, each integer designates the position of the child record under its parent at each level. The rightmost integer designates the level 0 position. Zero means last. |
| *value-stream* | is a series of component numbers and associated item values in the same format as for the loader stream. The record number to the left of the EQ operator is not included in the value stream. See **LOAD** on page 3-40 for the format of a loader stream. |
| PREVIOUS | repeats the value stream or the item value in the preceding update command. |
| *partial-trace* | is a trace in which the rightmost integer designates a position other than level 0. |
| *where-clause* | selects specific data records. See **Where-Clause** on page 3-62 for details. |

## Example

```
AT C100 EQ 101*PROGRAMMER*102*INFORMATION SYSTEMS*103*MYJ*104*
   PROFESSIONAL*105**TODAY**110*111*1100.00*112*MONTHLY*113*
   *TODAY**114*194.50*120*120*130*131*1.0*132*NEW EMPLOYEE*END*
   WH C1 = 1345:
```

## Rules

- Abbreviation: ASSIGN TREE = AT
- If any of the original data are to be retained, they must be included as part of the value stream. Otherwise, the data will be removed from the database.
- If a value stream is not included, the selected record and its descendants are replaced by an empty record without descendants.
- You must have U-authority for components specified in an ASSIGN TREE command.
- The command referenced by PREVIOUS cannot contain errors.
- When specifying PREVIOUS, the item or record preceding EQ must be the same item or record as in the previous update command.
- You cannot use Collect File items in an ASSIGN TREE action-clause.
- You cannot issue ASSIGN TREE for a damaged database.

## BY-CLAUSE

Specifies that only certain items within a data tree are to be displayed together in logical order. Enables you to specify which values are to be retrieved, to retrieve data from disjoint records, or to specify computations on subsets of data.

```
by-phrase, units
```

```
unit by-phrase
```

```
by-phrase   BY |record
               |DATA BASE

     unit   |item
            |record
            |* function *
            |* function (parameters)
            |* string *
            |* string (parameters)
            |(adhoc-function)
            |system-function
            |cf-item
```

| | |
|---|---|
| *item* | is a schema item name or component number. |
| *record* | is a schema record name or component number. |
| *function* | is a stored function name or component number. |
| *string* | is a stored string name or component number. |
| *parameters* | are the values that replace the numbered parameters in the stored function. See **Function Definition** on page 2-6 for details. |
| *adhoc-function* | is a mathematical expression enclosed in parentheses. See **Ad Hoc Function** on page 3-6 for details. |
| *system-function* | allows you to obtain simple arithmetic statistics about values stored in a database. System functions include: AVG, COUNT, MAX, MIN, SIGMA, and SUM. See **System Function** on page 3-56 for details. |
| *cf-item* | is a Collect File item name. |

## Examples

```
LIST C1, C2, BY C100, C102, C132 WH C7 EQ FEMALE:

LIST C1, C2, BY C100, C102, C132, BY C110, C121 WH C7 = FEMALE:

PRINT BY C0, C102, C201 WH C7 EQ FEMALE:

PRINT COUNT C0, BY DATA BASE, COUNT C0 WHERE C7 = FEMALE:

PRINT C111 BY C0 WHERE SAME:

PRINT BY C0, C111 BY C100 WHERE SAME:
```

## Rules

- In a by-phrase, BY DATA BASE qualifies the entire database for processing.
- Only numeric item types can be used with the AVG, SIGMA, and SUM system functions.
- You can use schema records with the COUNT system function, but you cannot use schema records with other system functions.
- The retrieval objects in a by-clause must be in the tree topped by the record specified in the by-phrase.

## CHANGE VALUE

Replaces existing values with new values.

```
CHANGE item  full-trace  EQ  |value *   :
                             |PREVIOUS
```

```
CHANGE item  full-trace  =  |item                 :
                            |* function *
                            |* function (parameters)
                            |(adhoc-function)
                            |system-function
```

```
CHANGE item  [partial-trace]  EQ  |value *  where-clause  :
                                  |PREVIOUS
```

```
CHANGE item  [partial-trace]  =  |item             where-clause  :
                                 |* function *
                                 |* function (parameters)
                                 |(adhoc-function)
                                 |system-function
```

```
CHANGE record  full-trace  EQ  |value-stream END * :
                               |PREVIOUS
```

```
CHANGE record  [partial-trace]  EQ  |value-stream END *  where-clause :
                                    |PREVIOUS
```

| | |
|---|---|
| *item* | is a schema item name or component number. |
| *full-trace* | is a list of one or more positive integers or 0, each preceded by the system separator (for example, *1*4*0). In a left-to-right scan, each integer designates the position of the child record under its parent at each level. The rightmost integer designates the level 0 position. Zero means last. |
| *value* | is a literal value or a system string (*NOW*, *TODAY*, *FTODAY*). |
| PREVIOUS | repeats the value stream or the item value in the preceding update command. |
| *function* | is a stored function name or component number. |
| *parameters* | are the values that replace the numbered parameters in the stored function. See **Function Definition** on page 2-6 for details. |

| | |
|---|---|
| *adhoc-function* | is a mathematical expression enclosed in parentheses. See **Ad Hoc Function** on page 3-6 for details. |
| *system-function* | allows you to obtain simple arithmetic statistics about values stored in a database. System functions include: AVG, COUNT, MAX, MIN, SIGMA, and SUM. See **System Function** on page 3-56 for details. |
| *partial-trace* | is a trace in which the rightmost integer designates a position other than level 0. |
| *record* | is a schema record name or component number. |
| *value-stream* | is a series of values to be changed, each preceded by its associated item number. The system separator must follow each item number and each value. The item numbers do not have to be in DESCRIBE order. |
| *where-clause* | selects specific data records. See **Where-Clause** on page 3-62 for details. |

## Examples

```
CH PAY SCHEDULE EQ MONTH*  WH C112 EQ MONTHLY:

CHANGE C11 EQ 80.00* WHERE C1 EQ 1120:

CHANGE C131 = (C131/10) WH C1 = 1043:

CH JOB SKILLS EQ 202*GOOD*203*3*END* WH C201 EQ GRAPHICS AND
   C1 EQ 1043:

CH CO EQ 14* 1302 LAZY LANE* 16* 78752* END* WH C1 = 1120:
```

## Rules

- Abbreviation: CHANGE = CH
- Use CHANGE only to change data within existing data records that are occurrences of a single schema record.
- If you need to change multiple values, all values must be occurrences of items in the specified record.
- If you use the syntax 'CHANGE item = item', the item types must be compatible, that is, both dates, both numeric, or both nonnumeric. For numeric types, the second item assumes the picture of the first item. Also, if the value of the second item is too large for the picture of the first item, an error message appears. If the item name for either includes the equals sign, use the component number for the item.

  Also for the 'CHANGE item = item' syntax, if one item is type UNDEFINED, the other item must also be type UNDEFINED. That is, you can change only UNDEFINED values to other UNDEFINED items. You cannot change CHARACTER or TEXT values to UNDEFINED items.
- If you use the '=' syntax, the item, stored function, system function, or ad hoc function must be in the path of the target record (the record containing the specified item). If the system function includes a by-phrase, the record specified in the by-phrase must be in the path of the target record.
- You must have U-authority for components specified in the action-clause of a CHANGE command.
- The command referenced by PREVIOUS cannot contain errors.
- When specifying PREVIOUS, the item or record preceding EQ must be the same item or record as in the previous update command.
- You cannot use Collect File items in a CHANGE action-clause.
- You cannot issue CHANGE for a damaged database.

## CLEAR

Specifies when modified database pages, or Update Log pages, or both are to be written from main memory to disk.

```
CLEAR :
```

```
[CLEAR] AUTOMATICALLY :
```

```
CLEAR UPDATE LOG :
```

```
[CLEAR] UPDATE LOG AUTOMATICALLY :
```

### Examples

```
CLEAR UPDATE LOG AUTO:

CLEAR:

CLEAR UPDATE LOG:
```

### Rules

- Abbreviation: AUTOMATICALLY = AUTO
- If you do not issue a CLEAR command, clearing is done by the buffer manager when space occupied by the updated database pages, Update Log pages, or both needs to be made available. Or, clearing is done by the software when certain commands are issued.
- When you issue CLEAR AUTOMATICALLY, the software clears all modified database pages and continues to do so after each update operation.
- When you issue CLEAR UPDATE LOG AUTOMATICALLY, the software clears all Update Log buffers and continues to do so after each update operation.

## COLLECT

Creates (or cancels) a Collect File, which is a relational table containing values extracted from one or more databases.

```
COLLECT [/[options] [,TITLE comment]/]  retrieval-clause

     [[,ordering-clause]  where-clause]   :
```

```
COLLECT  :
```

*options* is a list of one or more of the following format options, separated by commas. Defaults are listed in the first column.

| | |
|---|---|
| INDENT | BLOCK |
| NULL SUPPRESS | NULL |
| NUMBER | NAME |
| REPEAT | REPEAT SUPPRESS |
| SINGLE SPACE | DOUBLE SPACE |
| STUB | STUB SUPPRESS |
| TREE | RECORD |
| ZERO SUPPRESS | ZERO |
| HEX OFF | HEX ON |

See **FORMOP** on page 3-30 for details on format options.

TITLE allows you to insert a comment within the output. (If you do not specify format options, do not include a comma before the keyword TITLE.)

*comment* can be any string of characters not containing the command terminator or a slash. The comment cannot begin with (, B(, D(, F(, L(, R(, or S(, and it must be less than 130 characters long, although leading, trailing, and extraneous blanks are not counted. The comment is displayed with the echo of the command in the Message File.

*retrieval-clause* *cf-retrieval-object* [, *cf-retrieval-object* ]

*cf-retrieval-object* [*cf-item* = ] *retrieval-object*

| | |
|---|---|
| *cf-item* | is a unique Collect File item name, which can be any character string. If reserved words or component numbers are used, the Collect File cannot be used to create a new database. The Collect File item name can be omitted if it is the same as the retrieval object's name. |
| *retrieval-object* | is a schema item name or component number, Collect File item name, stored function, system function, or ad hoc function. You cannot specify a schema record unless it is embedded in a system function. |
| *ordering-clause* | specifies the order of selected data records. A where-clause is required if an ordering-clause is specified. See **Ordering-Clause** on page 3-42 for details. |
| *where-clause* | selects specific data records. See **Where-Clause** on page 3-62 for details. |

## Examples

```
COLLECT LAST NAME, HIRE DATE, ACCRUED VACATION:

COLLECT:

CO NAME = LAST NAME, HIRED = HIRE DATE, VACATION = ACCRUED VACATION,
   OB HIRE DATE WH HIRE DATE SPANS 01/01/1988 * 01/01/1989:
```

## Rules

- Abbreviation: COLLECT = CO
- To cancel a Collect File, issue COLLECT without other syntax, or end the session.
- You can display Collect File items with the DESCRIBE command.
- You can specify Collect File items in the retrieval-clause of PRINT, LIST, UNLOAD, and COLLECT commands, and in any where-clause except with MAP and RELOAD.
- If the retrieval-clause specifies items in disjoint records and if a where-clause is specified, the retrieval-clause must contain a by-phrase.
- You cannot specify a schema record as a Collect File item.
- Each Collect File item name must be unique.
- A maximum of 59 Collect File items can be collected in a Collect File record.
- If you specify an ad hoc function in a COLLECT command, the Collect File item type will be the type of the last item (or constant) in the function. For example, if C1 is a DATE item type, COLLECT (C1 + 20) produces an INTEGER Collect File item whose values equal the elapsed days in the C1 values plus 20. However, COLLECT (20 + C1) produces a DATE Collect File item whose values are the C1 dates plus 20 days.

## COPY TREE

Duplicates a data record and its descendants from one position in a database to another.

```
COPY TREE record |where-clause                TO |before-clause                   :
                 |partial-trace where-clause     |after-clause
                 |full-trace                     |partial-trace where-clause
                                                 |full-trace
```

| | |
|---|---|
| *record* | is the name or component number of the schema record at the top node of the tree to be copied. |
| *where-clause* | selects specific data records. See **Where-Clause** on page 3-62 for details. |
| *partial-trace* | is a trace in which the rightmost integer designates a position other than level 0. |
| *full-trace* | is a list of one or more positive integers or 0, each preceded by the system separator (for example, *1*4*0). In a left-to-right scan, each integer designates the position of the child record under its parent at each level. The rightmost integer designates the level 0 position. Zero means last. |
| *before-clause* | is the same as a where-clause except that the keyword BEFORE replaces the keyword WHERE. See **Where-Clause** on page 3-62 for details. |
| *after-clause* | is the same as a where-clause except that the keyword AFTER replaces the keyword WHERE. See **Where-Clause** on page 3-62 for details. |

### Examples

```
COPY TREE C110 WHERE C113 = 03/01/90 AND C1 = 1120
   TO AFTER C113 EQ 03/01/91 AND C1 EQ 1120:
CT C0*1 TO *3:
```

## Rules

- Abbreviations:
  AFTER = AF
  BEFORE = BE
  COPY TREE = CT

- You must have U-authority for components specified in a COPY TREE command.

- The copy operation requires a one-to-one relationship. That is, a data tree is copied from its old position to a single new point of attachment. It cannot be copied to multiple new positions.

- The data tree to be copied can be an entire logical entry or a subtree.

- Copying a data tree changes the Hierarchical Table and the Data Table. The index tables are also changed if the data tree copied contains key values.

- Although you cannot specify format options with COPY TREE, this command honors the TREE/RECORD format option in effect. That is, if RECORD is specified in a previous command, COPY TREE duplicates only the top node of the selected data tree.

- You can use SAME in either or both where-clauses. The DYNAMIC and STATIC settings are honored in the same way as any other consecutive where-clauses. Therefore, the "from" where-clause can affect the "to" where-clause, depending on which setting is used for SAME. After a COPY TREE completes processing, the SAME file contains the results of the specified "to" where-clause if SAME is set to DYNAMIC.

- You cannot issue COPY TREE for a damaged database.

- If Security by Entry is in effect for the database, you can use a secondary password to copy a subtree from one logical entry to another, provided you know the entry key values for both logical entries. The last where-clause used in setting up the "to" position determines the current entry key after the COPY TREE command completes processing.

## DATE FORMAT IS

Specifies alternate input and output formats for dates.

```
DATE FORMAT IS   |MM/DD/YYYY :
                 |MM/DD/YY
                 |DD/MM/YYYY
                 |DD/MM/YY
                 |YY/MM/DD
                 |YYYY/MM/DD
```

### Example

```
DATE FORMAT IS MM/DD/YY:
```

### Rules

- At the beginning of each session, the date format is set to MM/DD/YYYY.
- Issuing this command changes the date format for any database you open during the session.
- This command changes the date format for your session only. Other users of the database are not affected.

## DESCRIBE

Displays all or part of a database definition.

```
DESCRIBE :
```

```
DESCRIBE  |[/DEFINE/]                  |component [THROUGH component] :
          |[/AUTHORITIES [n]/]         |RECORD unit
          |[/DEFINE,AUTHORITIES/]
```

```
DESCRIBE  [/DEFINE/]  |STRINGS   :
                      |FUNCTIONS
```

```
DESCRIBE /COLLECT FILE [,DEFINE]/  [cf-item [(CF)]

   [THROUGH  cf-item  [(CF)]]]  :
```

| | |
|---|---|
| *component* | is an item, record, string, or function name or component number. |
| *n* | is a positive integer indicating the column in which component authorities are to be printed. The number must be between one and the line length, minus three. The default is 68. |
| *unit* | is the component name or number of a schema record or schema item for the RECORD option. If you specify an item, the software displays the parent schema record along with its items and subordinate records. |
| *cf-item* | is a Collect File item name. |

## Examples

```
DESCRIBE:

DESCRIBE /DEFINE/ SECURITY CLEARANCE:

DESC C110:

DESCRIBE /CF/ C13 (CF):

DESCRIBE C400 THRU C415:

DESCRIBE STRINGS:

DESCRIBE FUNCTIONS:

DESCRIBE /AUTHORITIES 45/:

DESCRIBE /AUTH 50/C100 THROUGH C110:

DESCRIBE RECORD C100:

DESCRIBE /AUTH 41/ RECORD C100:

DESCRIBE /DEFINE/ RECORD C100:

DESCRIBE /DEFINE, AUTH/ C0 THRU C10:
```

## Rules

- Abbreviations:
  - AUTHORITIES = AUTH
  - COLLECT FILE = CF
  - DESCRIBE = DESC
  - THROUGH = THRU

- DESCRIBE can be issued by the master password holder or a secondary password holder. If the DBA password holder issues DESCRIBE, the software displays header information only.

- If a command specifies THROUGH (for example, DESCRIBE C400 THROUGH C415:), the first component name or number must precede the second component name or number in the database definition.

- If the component name consists of several words and one of them is either THROUGH or THRU (for example, THROUGH STREETS), specify the component number instead of the component name. This rule does not apply to component names such as THROUGHPUT or THRUWAY, but it does apply to THROUGH-STREETS or THRU-WAY.

- The software displays Collect File items only if you specify the /COLLECT FILE/ option. Use the optional (CF) syntax if the Collect File item name is the same as a component in the database.

- Do not mix database components and Collect File items in this command.

- You cannot specify the /AUTHORITIES/ option with the /COLLECT FILE/ option or when describing strings or functions.

- To display component C0 (ENTRY), you must change ENTRY to another name.

- You can enter the /DEFINE,AUTHORITIES/ option as /AUTHORITIES,DEFINE/.

- For DESCRIBE RECORD, the output includes descriptions of schema records that are part of the specified record but not the items in those subordinate records.

## DITTO

Repeats the most recently executed action-clause of a retrieval or update command.

```
DITTO  [where-clause]  :
```

| | |
|---|---|
| *where-clause* | selects specific data records. See **Where-Clause** on page 3-62 for details. |

### Example

```
PR COUNT ENTRY WH C7 EQ FEMALE AND C124 GE $1000.00 AT 1:
   .
   .
   .
DI WH GENDER EQ MALE AND C124 GE $1000.00 AT 1:
```

### Rules

- Abbreviation: DITTO = DI
- You can use DITTO after the following commands: ADD, ASSIGN, ASSIGN TREE, CHANGE, COLLECT, DESCRIBE, INSERT TREE, LIST, PRINT, REMOVE, REMOVE TREE, TALLY, UNLOAD.
- If the previous retrieval or update command does not contain a where-clause, the DITTO command must not contain one. However, if the previous retrieval or update command contains a where-clause, the DITTO command must contain one.
- The keyword identifying the where-clause (WHERE, BEFORE, AFTER) must be the same in the DITTO command as in the previous retrieval or update command.
- If the action-clause in the previous retrieval or update command contains an ordering-clause, it is repeated by DITTO. However, if the action-clause in the previous retrieval or update command does not contain an ordering-clause, one cannot be added with DITTO.

## ENABLE/DISABLE EXECUTION

Allows or prevents the execution of QUEST, DEFINE, or QUEUE retrieval and update commands. However, ENABLE EXECUTION and DISABLE EXECUTION allow scanning of those commands for syntax errors.

```
ENABLE EXECUTION  :
```

```
DISABLE EXECUTION  :
```

### Examples

```
ENABLE EXECUTION:

DISABLE EXECUTION:
```

### Rules

- At the beginning of each session, execution is enabled.
- You can issue this command from the DEFINE, QUEST, or QUEUE processor. Commands in all three processors are affected, regardless of which processor ENABLE/DISABLE EXECUTION is issued from.

## ENABLE/DISABLE VALUES PADDING

Controls the placing of new values of key items in the Distinct Values Table.

```
ENABLE VALUES PADDING  :
```

```
DISABLE VALUES PADDING  :
```

### Example

```
ENABLE VALUES PADDING:
```

### Rules

- Only the master password holder or the DBA password holder (with authorization) can issue this command.
- The default is disabled values padding. Once ENABLE VALUES PADDING is issued, it remains in effect until DISABLE VALUES PADDING is issued.
- Enable values padding before doing massive loading with the LOAD command or with a PLEX program. After the load processing completes, you should disable values padding or too much table space will be added during ad hoc updates.
- ENABLE VALUES PADDING has no effect on the indexes for the items defined WITH MANY FUTURE OCCURRENCES.

## FORMOP

Sets format options without your having to issue a PRINT, LIST, UNLOAD, or COLLECT command.

```
FORMOP /options/   :
```

*options* is a list of one or more of the following format options, separated by commas. Defaults are listed in the first column.

| | |
|---|---|
| INDENT | BLOCK |
| NULL SUPPRESS | NULL |
| NUMBER | NAME |
| REPEAT | REPEAT SUPPRESS |
| SINGLE SPACE | DOUBLE SPACE |
| STUB | STUB SUPPRESS |
| TREE | RECORD |
| ZERO SUPPRESS | ZERO |
| HEX OFF | HEX ON |

The rules pertaining to the format options also apply when you specify format options with the PRINT, LIST, UNLOAD, or COLLECT command.

### Example

```
FORMOP /REPEAT SUPPRESS, DOUBLE SPACE/:
```

### Rules

- Abbreviations:
  NUMBER = NUM, NUMB
  SUPPRESS = SUP
- This command changes the format options until the end of the session or until you issue another FORMOP, PRINT, LIST, UNLOAD, or COLLECT command specifying new format options.
- If conflicting format options are specified, the one specified last will be in effect.
- INDENT/BLOCK controls the indentation of lower level items.
- SINGLE SPACE/DOUBLE SPACE controls vertical spacing of output.

- Specifying DOUBLE SPACE in an UNLOAD command causes errors because of unwanted blanks between lines.

- STUB/STUB SUPPRESS controls the display of item labels. If STUB is in effect, NUMBER/NAME affects whether a component number or name precedes a value. STUB SUPPRESS eliminates the name or number, as well as the default column headings in a LIST command.

- NUMBER/NAME specifies that the item name or number is to be displayed when a PRINT command is issued and STUB is in effect. If STUB SUPPRESS is in effect, NUMBER/NAME does not affect output.

- NULL SUPPRESS/NULL controls the display of null values for a PRINT command. If NULL SUPPRESS is in effect, null items do not display. If NULL is in effect, the word -NULL- is displayed for items with no values.

- NULL SUPPRESS/NULL and REPEAT/REPEAT SUPPRESS have no affect on records written to a Collect File.

- ZERO SUPPRESS/ZERO specifies whether nulls are to be treated as zeros in arithmetic expressions. ZERO SUPPRESS causes null values in INTEGER, DECIMAL, and MONEY item types to be treated as zeros in ad hoc or stored functions. ZERO causes nulls to be changed to zero for items included in computations. ZERO/ZERO SUPPRESS affects a system function if the object of the system function is enclosed in parentheses.

- TREE/RECORD controls the output of lower level data records when you specify a record in the retrieval-clause of a PRINT or UNLOAD command. TREE displays data trees whose top records are occurrences of the specified record. RECORD displays only data records that are occurrences of the specified record.

- If RECORD is in effect for a RELOAD, only level 0 records are reloaded. If RECORD is in effect for a MAP that causes restructuring, only level 0 records are retained. Descendant records are lost.

- REPEAT/REPEAT SUPPRESS controls the repeated display of data from common ancestors of selected data records. When REPEAT is in effect and two or more selected data records have common ancestors, data from those ancestor records are displayed each time data from a selected record are displayed. With REPEAT SUPPRESS, if successive selected data records have a common ancestor, values from that ancestor record are displayed only once, and the values from each data tree appear in logical order.

- HEX OFF/HEX ON determines whether UNDEFINED output values appear in HEX notation or display format.

- Although you cannot specify format options in a COPY TREE command, the command honors the TREE/RECORD format option in effect. That is, if RECORD is in effect, COPY TREE duplicates only the top node of the selected data tree.

## FRAME AND END FRAME

Establishes a logical unit of work for Coordinated Recovery, or places a global hold on the database, or both. If the Rollback Log is enabled, each FRAME and END FRAME command causes a database synchpoint.

```
FRAME [/IMMEDIATE/]   :

END FRAME   :
```

### Examples

```
FRAME /IM/:

END FR:
```

### Rules

- Abbreviations:
  IMMEDIATE = IM
  FRAME = FR
- Use the IMMEDIATE syntax to maintain a previously obtained local hold.
- You cannot issue the commands DEFINE, CONTROL, REPORT, COMPOSE, or EXIT between FRAME and END FRAME commands.

# INSERT TREE

Inserts new data trees where trees do not exist.

```
INSERT TREE  record  EQ  |value-stream END*  |before-clause :
                         |PREVIOUS           |after-clause
```

```
INSERT TREE  record  full-trace  EQ  |value-stream END* :
                                     |PREVIOUS
```

```
INSERT TREE  record  partial-trace  EQ  |value-stream END* where-clause:
                                        |PREVIOUS
```

| | |
|---|---|
| *record* | is a schema record name or component number. |
| *value-stream* | is a series of component numbers and associated data values in the same format as the loader stream. The record number to the left of the EQ operator is not included in the value stream. See **LOAD** on page 3-40 for the format of a loader stream. |
| PREVIOUS | repeats the value stream in the preceding update command. |
| *before-clause* | is the same as a where-clause, except that the keyword BEFORE replaces the keyword WHERE. See **Where-Clause** on page 3-62 for details. |
| *after-clause* | is the same as a where-clause, except that the keyword AFTER replaces the keyword WHERE. See **Where-Clause** on page 3-62 for details. |
| *full-trace* | is a list of one or more positive integers or 0, each preceded by the system separator (for example, *1*4*0). In a left-to-right scan, each integer designates the position of the child record under its parent at each level. The rightmost integer designates the level 0 position. Zero means last. |
| *partial-trace* | is a trace in which the rightmost integer designates a position other than level 0. |
| *where-clause* | selects specific data records. See **Where-Clause** on page 3-62 for details. |

## Examples

```
IT C420 EQ 421*422*COBOL*423*09/09/1986*END* AFTER C2 = REID
   AND C421 = TECNIK:

INSERT TREE C120 *1*1*1 EQ 121*06/30/91* END* WH C1 EXISTS:

IT C120 *1*1*1 EQ 121*05/29/91*122*184.00*123*10.00*124*1950.00*
   125*153.20*126*1560.35*END* WHERE C1 EQ 1120:

IT C200 *2 EQ 201*COBOL*202*GOOD*203*1*END* WH C1 = 1120:

IT C0 *0 EQ 100*101*PROGRAMMER*102*INFORMATION SYSTEMS*130*131*
   1.0*132*NEW EMPLOYEE*END*:
```

## Rules

- Abbreviations:
  AFTER = AF
  BEFORE = BE
  INSERT TREE = IT
- If the value stream is not included, an empty record is added.
- If INSERT TREE contains a before-clause or an after-clause, the record at the top of the new subtree is placed immediately before or after each selected record.
- If the value stream contains a record number immediately followed by the system separator and another record number, an empty record is inserted for the former record.
- You must have U-authority for components specified in an INSERT TREE command.
- The command referenced by PREVIOUS cannot contain errors.
- When specifying PREVIOUS, the record preceding EQ must be the same record as in the previous update command.
- You cannot use Collect File items in an INSERT TREE action-clause.
- You cannot issue INSERT TREE for a damaged database.

## LIMIT AND END LIMIT

Does three things: sets a specific number of records on which retrieval or update commands are to be executed, sets a minimum and maximum for the number of records on which retrieval or update commands are to be executed, or controls the output of selected records in retrieval commands.

```
LIMIT  k  :

LIMIT  k , m |[/CANCEL ]   :
             |[/TRUNCATE]

LIMIT  k , m |[/CANCEL ] , j [/fileref]  :
             |[/TRUNCATE]

END LIMIT  :
```

| | |
|---|---|
| *k, m, j* | are positive integers or 0. Zero means infinite for *m* and *j* . *Also, k* and *j* must be less than or equal to *m* except when *m equals 0. The j* intermediate limit has no effect on Collect File commands. |
| CANCEL<br>TRUNCATE | controls the output of selected records. The default is CANCEL; that is, if the number of selected records falls outside the range specified, the command is cancelled. |
| *fileref* | is a DDname declared in the JCL. The default is the Report File. |
| END LIMIT | resets the software to the default, which is unlimited. |

### Examples

```
LIMIT 3:

LIMIT 3, 5:

LIMIT 3, 3/TRUNCATE:

LIMIT 3, 7, 5:

LIMIT 3, 7, 5/ALTFILE:

LIMIT 3, 7/TRUNCATE, 5:

LIMIT 3, 7/TRUNCATE, 5/ALTFILE:

END LIMIT:
```

## Rules

- At the beginning of each session, all limits are set to 0 (that is, unlimited).
- If you set *j*, a Collect File where-clause is not allowed for a retrieval command.
- If you set any limits, an update command with a Collect File where-clause is not allowed.

## LIST

Retrieves database and Collect File information and displays it in columnar format.

```
LIST [/[options] [,TITLE specifications ]/] retrieval-clause

     [[,ordering-clause] where-clause] :
```

| | |
|---|---|
| *options* | is a list of one or more of the following format options, separated by commas. Defaults are listed in the first column. |

| | |
|---|---|
| INDENT | BLOCK |
| NULL SUPPRESS | NULL |
| NUMBER | NAME |
| REPEAT | REPEAT SUPPRESS |
| SINGLE SPACE | DOUBLE SPACE |
| STUB | STUB SUPPRESS |
| TREE | RECORD |
| ZERO SUPPRESS | ZERO |
| HEX OFF | HEX ON |

See **FORMOP** on page 3-30 for details on format options.

| | |
|---|---|
| TITLE | allows you to include layout specifications. Use a comma in front of the keyword TITLE only if you have specified a format option. Title and paging specifications can occur anywhere after the keyword TITLE. |
| *specifications* | See **Layout Specifications** on page 3-38. |
| *retrieval-clause* | specifies a list of one or more retrieval objects, separated by commas. See **Retrieval-Clause** on page 3-52 for details. |
| *ordering-clause* | specifies the order of selected data records. A where-clause is required if an ordering-clause is specified. See **Ordering-Clause** on page 3-42 for details. |
| *where-clause* | selects specific records. See **Where-Clause** on page 3-62 for details. |

### Examples without Layout Specifications

```
LIST EMPLOYEE NUMBER, LAST NAME WHERE EMPLOYEE NUMBER LE 1005:

LIST C2, C1, C6, ORDERED BY C2 WHERE C1 <= 1003:

LIST LAST NAME, *C2001*, (C106 - C105), AVG C111 WHERE C106 EXISTS:

LIST/DOUBLE SPACE, REPEAT SUP/ C2,C4,C7,C15,C16 WHERE C1 <= 1003:
```

## Layout Specifications

| | |
|---|---|
| *specifications* | [(*page-width*)] [*title*] [*paging*] [*rows*] |

| | |
|---|---|
| (*page-width*) | is the maximum number of characters for a line, 4 to 131. The defaults are 131 for batch and 71 for interactive. |
| *title* | D (*starting-column*) [*title-line-text*] |
| *paging* | F (*lines-per-page*) [*footing-line-text*]<br>Lines-per-page must be from 10 to 60. |

```
rows            |L (width) [column-heading]  [,S-specification]
                |R
                |B
```

| | |
|---|---|
| **L** | specifies left-justified column heading. |
| **R** | specifies right-justified column heading. |
| **B** | specifies a blank column; optional heading is left-justified.<br><br>Not specifying L, R, or B directs the software to left-justify the heading text and to allow the longest of the three lines possible to determine the width of the columns. |
| (*width*) | is the column heading width. |
| *column-heading* | [[*text*] + ] [[*text*] + ] *text* |
| *S-specification* | tells the software where to start a new row, how to space between rows of column headings, and how to space between corresponding rows of values. For syntax, see LIST command in *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition*. |

## Examples with Layout Specifications

```
LIST /TITLE R(16)MAJOR FIELD, L(16)MINOR FIELD/ C414, C415 WHERE
   C415 EXISTS AND C1 LE 1100:

LIST /TITLE (45) D(15) REPORT TITLE, F(15)     --FOOTING LINE--,
   L(10) LAST NAME, EMPLOYEE+NUMBER, R(15) SOCIAL     +SECURITY
   NUMBER/ C2, C1, C6 WHERE C1 =< 1005:

LIST /DOUBLE SPACE, TITLE (40)  D(7)REPORT TITLE,
   F(40)     --FOOTING LINE--,
   L(10) LAST NAME,S(1,2),B(10),
   L(8)EMPLOYEE+ NUMBER,R(15)SOCIAL     +SECURITY NUMBER,S(2)/
   C2, C1, C6 WHERE C1 LE 1005:

LI/TITLE D(20)*NOW*/C1, C2, C3 WH C1 <= 1010:
```

## Rules

- Do not use a record name or number in the retrieval-clause, except with the BY operator or with a COUNT system function.
- Use commas to separate title, paging, and row specifications.
- Page width must accommodate title and footing line text; the page number on the title line uses five spaces.
- Do not include the command terminator, a comma, or a slash in the title line text and footing line text.
- Column heading text cannot include a comma, plus sign, command terminator, or slash.
- You can specify up to three lines of column heading text. With the S-specifications, you can define up to five rows of column headings.
- Total width of columns must be at least two spaces less than page width in order to contain row identifiers and space at the left of each row.
- You can include any number of B-specifications but no more than 59 specifications for data columns. B-specifications can be adjacent.
- You cannot use the system string *TODAY* in a layout specification because slashes in this case are treated as delimiters. You can, however, use the system strings *FTODAY* and *NOW* in a layout specification.

## LOAD

Loads data into a database in logical order.

```
LOAD   :
```

### Example

```
LOAD:
```

### Rules

- When you issue this command, the software looks to the Data File for a loader stream. Therefore, you must first issue the DATA FILE IS command.
- A loader stream consists of a group of values, with each value preceded by an item number and followed by a system separator. Each group of values (a data record) is preceded by a schema record number. The item numbers and record numbers are not preceded by a C.
- You indicate the end of a data record by either another schema record number or an entry terminator (END*).
- You indicate the end of the loader stream by following the last entry terminator with a system separator. For example,

```
1*1120*2*REID*3*DAVID G.*4*01/16/1988*5*08/15/1957*6*441-04-0121*
    .
    .
    .
420*421*TECNIK*422*SYSTEM RS ADVANCED*423*03/18/1989*END**
```

- Enable values padding before doing a LOAD operation. After the load processing completes, you should disable values padding or too much table space will be added during ad hoc updates. See ENABLE/DISABLE VALUES PADDING command.
- To check for errors before loading, issue the DISABLE EXECUTION command or the STOP IF command before issuing the LOAD command.
- When the loading process is complete, the database cycle number is incremented by one.

## MOVE TREE

Moves a data record and its descendants from one position in a database to another without affecting the database tables or index.

```
MOVE TREE record |where-clause                TO |before-clause                  :
                 |partial-trace where-clause     |after-clause
                 |full-trace                     |partial-trace where-clause
                                                 |full-trace
```

| | |
|---|---|
| *record* | is a schema record name or component number; the target record. |
| *where-clause* | selects specific records. See **Where-Clause** on page 3-62. |
| *partial-trace* | is a trace in which the rightmost integer designates a position other than level 0. |
| *full-trace* | is a list of one or more positive integers or 0, each preceded by the system separator (for example, *1*4*0). In a left-to-right scan, each integer designates the position of the child record under its parent at each level. The rightmost integer designates the level 0 position. Zero means last. |
| *before-clause* | is the same as a where-clause except that the keyword BEFORE replaces the keyword WHERE. See **Where-Clause** on page 3-62. |
| *after-clause* | is the same as a where-clause except that the keyword AFTER replaces the keyword WHERE. See **Where-Clause** on page 3-62 for details. |

### Example

```
MT C110 WH C113 EQ 03/01/90 AND C1 EQ 1120 TO AFTER C113 = 03/01/91
   AND C1 = 1120:
```

### Rules

- Abbreviations:
  AFTER = AF
  BEFORE = BE
  MOVE TREE = MT
- You must have U-authority for components specified in a MOVE TREE command.
- If Security by Entry is in effect, a secondary password holder can move a subtree from one logical entry to another, provided the secondary password holder knows both entry key values.
- You cannot issue MOVE TREE for a damaged database.

## ORDERING-CLAUSE

Specifies the output order of data to be retrieved with PRINT, LIST, UNLOAD, and COLLECT commands. Can also be used with RELOAD and MAP.

```
,ORDERED BY  sort-keys
```

| | |
|---|---|
| *sort-keys* | is a list of one or more items, records, ad hoc functions, stored functions, or system functions in the following format, separated by commas.<br><br>`|[LOW]  unit`<br>`|[HIGH]` |
| LOW | specifies low-to-high (ascending) order, for example, A,B,C,D or 1,2,3,4. The default is LOW. |
| HIGH | specifies high-to-low (descending) order, for example, Z,Y,X,W or 9,8,7,6. |
| *unit* | \|*item*<br>\|*record*<br>\|*stored-function*<br>\|(*adhoc-function*)<br>\|*system-function* |
| *item* | is a schema item name or component number. |
| *record* | is a schema record name or component number. |
| *stored-function* | is a stored function name or component number with optional parameters. |
| *adhoc-function* | is a mathematical expression enclosed in parentheses. See **Ad Hoc Function** on page 3-6 for details. |
| *system-function* | allows you to obtain simple arithmetic statistics about values stored in a database. System functions include: AVG, COUNT, MAX, MIN, SIGMA, and SUM. A by-phrase is required if you use a system function in an ordering-clause. See **System Function** on page 3-56 for details. |

## Examples

```
PRINT C1, C2, C10, ORDERED BY C1, C10 WHERE C1 EXISTS:

UNLOAD C1, OB C1 WH C1 EXISTS:

PR C1, C2, C11, OB HIGH C1, LOW C2, HIGH C11 WH C11 EXISTS:
```

## Rules

- Abbreviation: ORDERED BY = OB
- A comma must precede the keywords ORDERED BY.
- If you specify an ordering-clause, a where-clause is required.
- The order in which you enter sort keys determines the order that the retrieved data are displayed.
- An item can be key or non-key.
- An ordering-clause that specifies a system function requires a by-phrase. See **By-Clause** on page 3-13 for details.
- Components in an ordering-clause must be in the same path as the target record in the retrieval-clause and as the focal record in the where-clause.
- Digits in numeric values for nonnumeric items are treated as characters (not numbers) for sorting purposes.
- Collect File items cannot be specified in an ordering-clause.

## PRINT

Retrieves database or Collect File information and displays it in a simple, sequential list.

```
PRINT [/[options] [,TITLE comment]/] retrieval-clause

     [[,ordering-clause] where-clause]  :
```

| | |
|---|---|
| *options* | is a list of one or more of the following format options, separated by commas. Defaults are listed in the first column. |

| | |
|---|---|
| INDENT | BLOCK |
| NULL SUPPRESS | NULL |
| NUMBER | NAME |
| REPEAT | REPEAT SUPPRESS |
| SINGLE SPACE | DOUBLE SPACE |
| STUB | STUB SUPPRESS |
| TREE | RECORD |
| ZERO SUPPRESS | ZERO |
| HEX OFF | HEX ON |

See **FORMOP** on page 3-30 for details on format options.

| | |
|---|---|
| TITLE | allows you to insert a comment within the output. (If you do not specify format options, do not include a comma before the keyword TITLE.) |
| *comment* | can be any string of characters not containing the command terminator or a slash. The comment cannot begin with (, B(, D(, F(, L(, R(, or S(, and it must be less than 130 characters long, although leading, trailing, and extraneous blanks are not counted. The comment is displayed with the echo of the command in the Message File. |
| *retrieval-clause* | specifies a list of one or more retrieval objects, separated by commas. See **Retrieval-Clause** on page 3-52 for details. |
| *ordering-clause* | specifies the order of selected data records. A where-clause is required if an ordering-clause is specified. See **Ordering-Clause** on page 3-42 for details. |
| *where-clause* | selects specific data records. See **Where-Clause** on page 3-62 for details. |

## Examples

```
PRINT C1, C2:

PR C100 WHERE C1 = 1265:

PRINT/DOUBLE SPACE,INDENT/ C2,C4,C7,C15,C16 WHERE C1 LE 1003:

PRINT /TITLE THIS TEXT DISPLAYED IN MESSAGE FILE/ C1, C2, C3:
```

## Rules

- Abbreviation: PRINT = PR

## QUEUE

Calls the QUEUE processor.

```
QUEUE   :
```

### Example

```
USER, DEMO:
DBN IS EMPLOYEE:
    .
    .  (QUEST commands)
    .
QUEUE:                        (calls QUEUE)
    .
    .
    .
```

### Rules

- For information about QUEUE processor commands, see *Technical Report: S2-110 QUEUE Language for SYSTEM 2000 Software, Release 11.6 under IBM OS and CMS.*

## RELOAD

Reconstructs database files.

```
RELOAD [[ordering-clause] where-clause] :
```

| | |
|---|---|
| *ordering-clause* | specifies the order of selected data records. You can specify only level 0 items. A where-clause is required if an ordering-clause is specified. See **Ordering-Clause** on page 3-42 for details. |
| *where-clause* | selects specific data records. You cannot include the SAME operator. See **Where-Clause** on page 3-62 for details. |

### Examples

```
RELOAD:

RELOAD WHERE EMPLOYEE NUMBER < 5075:

RELOAD ORDERED BY C1 WHERE C1 EXISTS:
```

### Rules

- Only the master password holder or the DBA password holder (with authorization) can issue this command. However, the DBA password holder cannot issue a RELOAD command containing a where-clause.
- You can issue RELOAD from the CONTROL or QUEST processor, but in either case, the CONTROL processor is attached after execution.
- This command requires the same amount of scratch file space as loading the database does.
- If you include a where-clause and the LIMIT command has been issued, only the number of selected data records within the specified limits are loaded.
- This command resets the cycle number to zero at the beginning of the reload process and increments it by one when the reloading is finished. However, if a specified where-clause produces no qualified data records, the cycle remains reset to zero.
- Issuing this command suspends update logging.
- If the RECORD format option is in effect, only level 0 data records are reloaded.
- In a Multi-User environment, you must have exclusive use of the database to issue this command.

## REMOVE

Replaces values with nulls in selected data records.

```
REMOVE  item  full-trace  :
REMOVE  item  [partial-trace]  where-clause  :
REMOVE  record  full-trace  :
REMOVE  record  [partial-trace]  where-clause  :
```

| | |
|---|---|
| *item* | is a schema item name or component number. |
| *full-trace* | is a list of one or more positive integers or 0, each preceded by the system separator (for example, *1*4*0). In a left-to-right scan, each integer designates the position of the child record under its parent at each level. The rightmost integer designates the level 0 position. Zero means last. |
| *partial-trace* | is a trace in which the rightmost integer designates a position other than level 0. |
| *where-clause* | selects specific data records. See **Where-Clause** on page 3-62 for details. |
| *record* | is a schema record name or component number. |

### Examples

```
RE CURRENT DEDUCTION WH C113 EQ 03/01/90 AND C1 EQ 1120:

REMOVE C302 WHERE C1 = 1043:

REMOVE ENTRY WHERE C1 FAILS:
```

## Rules

- Abbreviation: REMOVE = RE
- REMOVE affects only those records that are occurrences of a single schema record.
- REMOVE eliminates either the value of a single item or the values of all items in selected records.
- You must have U-authority for components specified in the action-clause of a REMOVE command.
- You cannot use Collect File items in a REMOVE action-clause.
- If you remove the values for all items in a record, the record remains in the database (as a null record). Only REMOVE TREE removes records.
- You cannot issue REMOVE for a damaged database.

## REMOVE TREE

Removes data trees (including the data) from a database.

```
REMOVE TREE  record  full-trace  :
```

```
REMOVE TREE  record  [partial-trace]  where-clause  :
```

| | |
|---|---|
| *record* | is a schema record name or component number; the target record. |
| *full-trace* | is a list of one or more positive integers or 0, each preceded by the system separator (for example, *1*4*0). In a left-to-right scan, each integer designates the position of the child record under its parent at each level. The rightmost integer designates the level 0 position. Zero means last. |
| *partial-trace* | is a trace in which the rightmost integer designates a position other than level 0. |
| *where-clause* | selects specific data records. See **Where-Clause** on page 3-62 for details. |

### Examples

```
REMOVE TREE ENTRY WHERE C2 = REID:

RT C130 *2 WHERE C1 EQ 1340:
```

### Rules

- Abbreviation: REMOVE TREE = RT
- REMOVE TREE removes the specified record and all descendant records.
- You must have U-authority for components specified in a REMOVE TREE command.
- You cannot use Collect File items in a REMOVE TREE action-clause.
- You cannot issue REMOVE TREE for a damaged database.

# REPORT

Calls the REPORT processor.

```
REPORT   :
```

## Example

```
USER, DEMO:
DBN IS EMPLOYEE:
     .
     .  (QUEST commands)
     .
REPORT:                    (calls REPORT)
     .
     .
     .
```

## Rules

- For information about REPORT processor commands, see *SYSTEM 2000 REPORT Language, Version 12, First Edition.*

## RETRIEVAL-CLAUSE

Specifies a list of components to be retrieved with a LIST, PRINT, UNLOAD, or COLLECT command. Can include records, items, stored strings, stored functions, ad hoc functions, system functions, by-clauses, and Collect File items -- separated by commas. Can be followed by an ordering-clause, a where-clause, or both.

```
units
```

```
unit    |item
        |record
        |* function *
        |* function (parameters)
        |* string *
        |* string (parameters)
        |(adhoc-function)
        |system-function
        |by-clause
        |cf-item
```

| | |
|---|---|
| *item* | is a schema item name or component number. |
| *record* | is a schema record name or component number. The record cannot be specified in the retrieval-clause of a LIST command unless it appears in a by-phrase or with the COUNT system function. |
| *function* | is a stored function name or component number. |
| *string* | is a stored string name or component number. |
| *parameters* | are the values that replace the numbered parameters in the stored function or string. |
| *adhoc-function* | is a mathematical expression enclosed in parentheses. See **Ad Hoc Function** on page 3-6 for details. |
| *system-function* | allows you to obtain simple arithmetic statistics about values stored in a database. System functions include: AVG, COUNT, MAX, MIN, SIGMA, and SUM. See **System Function** on page 3-56 for details. |
| *by-clause* | specifies that only values for certain items are to be displayed in logical order. See **By-Clause** on page 3-13 for details. |
| *cf-item* | is a Collect File item name. The item name must be followed by (CF) if it has the same name as a schema component. |

## Examples

```
PR C1, C2, C3:

PRINT LAST NAME WHERE ACCRUED VACATION EXISTS:

LIST C1, C2, C3, *C2010* WH C1 LE 1010:

PR (C114 + C126) WH C1 EQ 1010:

PR SUM C124 BY C110, SUM C124 BY DATA BASE WH C1 = 1120:

PR MIN C2, MAX C2:

PR C1, (*FTODAY* - C4) WH C1 <= 1004:
```

## Rules

- You can specify up to 59 units (five rows) in a LIST command retrieval-clause.
- If a retrieval command includes a where-clause and if no by-phrases are specified in the retrieval-clause, all retrieval objects in the retrieval-clause must be related to the qualified records.
- If a retrieval command includes a where-clause and if one or more by-phrases are specified in the retrieval-clause, the retrieval objects preceding the first by-phrase, as well as all records specified in the by-phrases, must be in the path of the target record. Also, the records in the by-phrases must be related to the records qualified by the where-clause.
- You must have R-authority for components specified in a retrieval-clause.
- The retrieval-clause can contain a mixture of schema components and Collect File items if there is a where-clause containing a condition with a Collect File item.

## SAME IS

Tells the software which retained list of qualified data records to use when the SAME operator is specified.

```
SAME IS DYNAMIC  :
```

```
SAME IS STATIC  :
```

### Example

```
SA IS STATIC:
```

### Rules

- Abbreviation: SAME = SA
- At the beginning of each session, SAME is set to DYNAMIC.
- The DYNAMIC setting allows you to issue a sequence of commands acting on an evolving collection of data records.
- The STATIC setting allows you to issue a sequence of commands acting on a static collection of data records. For each individual command, the collection can be modified with additional qualification criteria.

## STOP IF/CONTINUE IF

STOP IF specifies a limit of command errors that can occur before the software ends a session. CONTINUE IF reverses the limit set in a STOP IF command.

```
STOP [AFTER SCAN] IF   |[ANY] [UPDATE]   |COMMAND IS    REJECTED  :
                       |[n]              |COMMANDS ARE
```

```
CONTINUE [AFTER SCAN] IF  |COMMAND IS    REJECTED  :
                          |COMMANDS ARE
```

| | |
|---|---|
| ANY | is the default maximum error count, which specifies 1. |
| *n* | is the maximum error count. The number must be greater than 0. |
| UPDATE | If you omit the UPDATE option, all command errors are counted. |

### Examples

```
STOP IF 2 COMMANDS ARE REJECTED:

CONTINUE IF COMMANDS ARE REJECTED:

STOP IF ANY COMMAND IS REJECTED:

STOP IF 2 UPDATE COMMANDS ARE REJECTED:

STOP AFTER SCAN IF 9 COMMANDS ARE REJECTED:
```

### Rules

- Abbreviation: AFTER = AF
- The status of STOP IF remains the same if you change processors.

## SYSTEM FUNCTION

Obtains simple statistics about database values.

```
|AVG    |unit  [by-phrase]
|COUNT
|MAX
|MIN
|SIGMA
|SUM
```

| | |
|---|---|
| *unit* | \|*item*<br>\|*record*<br>\|*stored-function*<br>\|(*adhoc-function*)<br>\|*cf-item*<br>\|(*system-function*) |
| *item* | is a schema item name or component number. |
| *record* | is a schema record name or component number. |
| *stored-function* | is a stored function name or component number with optional parameters. |
| *adhoc-function* | is a mathematical expression enclosed in parentheses. See **Ad Hoc Function** on page 3-6 for details. |
| *cf-item* | is a Collect File item name. |
| *by-phrase* | BY \|*record*<br>\|DATA BASE<br><br>See **By-Clause** on page 3-13 for details. |
| *system-function* | You can have one system function inside another. The inner system function must be enclosed in parentheses. |

### Examples

```
PRINT MIN C2, MAX C2:

PR COUNT C126, MIN C126, MAX C126, SUM C126, AVG C126, SIGMA C126:

PR SUM C124 BY C0 WH C1 LE 1010:

PR AVG (SUM C124 BY C0):

PR (((MIN C126) + (MAX C126))/2):
```

## Rules

- You can use system functions in the action-clause of a LIST, PRINT, or UNLOAD command, in an ordering-clause with a by-phrase, and in update commands if the system functions are enclosed in parentheses.
- AVG produces the average of the values stored for the requested numeric item.
- COUNT produces a count of the number of times a data value exists or a data record occurs. This is the only system function you can use with records.
- MAX produces the maximum value stored for the requested item.
- MIN produces the minimum value stored for the requested item.
- SIGMA produces the standard deviation for the requested numeric item.
- SUM adds all the values for the requested numeric item.
- You can use any system function with a by-phrase, but the by-phrase should contain an ancestor record name or number.
- You can use SUM, AVG, and SIGMA with numeric item types only (including dates).
- You cannot use Collect File items with system functions.
- If you use a system function in an ad hoc function, the system function must be enclosed in parentheses.
- When the ZERO format option is in effect and when the object of a system function is enclosed in parentheses, nulls are treated as zero.

## SYSTEM STRING

Provides the current date in date and function format and the current time according to your computer's internal calendar and clock.

| | |
|---|---|
| `*NOW*` | `current time` |
| `*TODAY*` | `current date` |
| `*FTODAY*` | `current date in function format` |

### Examples

```
PRINT C1, (*FTODAY* - C4) WHERE C1 LE 1004:

LIST/TITLE D(20) *NOW*/ C1, C2, C3 WH C1 <= 1010:
```

### Rules

- Use *FTODAY* in ad hoc and stored functions.
- Use *TODAY* and *NOW* in update and retrieval commands.
- Use *NOW* to time-stamp reports.
- The default formats for time and date are:

| | |
|---|---|
| *NOW* | HH.MM.SS |
| *TODAY* | MM/DD/YYYY |
| *FTODAY* | MM.DD.YYYY |

- You cannot use *TODAY* in the layout specifications of a LIST command because the slashes in this case are treated as delimiters.
- You can use the system strings in a Command File for updates, but they cannot be used in a Data File.

## TALLY

Provides summary information about the values stored for key items in a database.

```
TALLY   |[/EACH/] items   :
        |[/ALL/]
```

*items* is a list of one or more key item names or component numbers, separated by commas. The maximum is 96 items.

### Examples

```
TA /EACH/ C8:

TALLY C7, C8, C9, C102:
```

### Rules

- Abbreviation: TALLY = TA
- At the beginning of each session, TALLY is set to EACH.
- EACH lists each distinct value for the specified items as well as the number of times each value occurs, along with the total number of values and total occurrences.
- ALL lists only the minimum and maximum values for the specified items, along with the total number of distinct values and total occurrences.
- You must have R-authority for items specified in a TALLY command.
- If Security by Entry is in effect, only the master password holder can issue TALLY.
- Issuing the LIMIT command affects TALLY output. Only values whose number of occurrences fall in the range of the limits are displayed.
- SINGLE SPACE/DOUBLE SPACE is the only format option that affects TALLY.

## UNLOAD

Retrieves data in a format suitable for use in a Data File.

```
UNLOAD  :
```

```
UNLOAD [/[options] [,TITLE comment]/] retrieval-clause

     [[,ordering-clause] where-clause]  :
```

```
UNLOAD/COLLECT FILE [,options] [,TITLE comment]/  :
```

*options* is a list of one or more of the following format options, separated by commas. Defaults are listed in the first column.

| | |
|---|---|
| INDENT | BLOCK |
| NULL SUPPRESS | NULL |
| NUMBER | NAME |
| REPEAT | REPEAT SUPPRESS |
| SINGLE SPACE | DOUBLE SPACE |
| STUB | STUB SUPPRESS |
| TREE | RECORD |
| ZERO SUPPRESS | ZERO |
| HEX OFF | HEX ON |

See **FORMOP** on page 3-30 for details on format options.

TITLE allows you to insert a comment within the output. (If you do not specify format options, do not include a comma before the keyword TITLE.)

*comment* can be any string of characters not containing the command terminator or a slash. The comment cannot begin with (, B(, D(, F(, L(, R(, or S(, and it must be less than 130 characters long, although leading, trailing, and extraneous blanks are not counted. The comment is displayed with the echo of the command in the Message File.

*retrieval-clause* specifies a list of retrieval objects, separated by commas. See **Retrieval-Clause** on page 3-52 for details.

*ordering-clause* specifies the order of selected data records. A where-clause is required if an ordering-clause is specified. See **Ordering-Clause** on page 3-42 for details.

*where-clause* selects specific data records. See **Where-Clause** on page 3-62 for details.

## Examples

```
UNLOAD:

UN C1, C2:

UNLOAD C100 WH C1 = 1265:

UNLOAD C1, C2, JOB SKILLS WH C1 EQ 1120:

UNLOAD /RECORD/ C100 WH C1 = 1265:

UN C1, C2, SUM C126, (C124 - C114) WH C1 EQ 1265:

UNLOAD /CF/:
```

## Rules

- Abbreviations:
  UNLOAD = UN
  COLLECT FILE = CF
- Issuing UNLOAD without any syntax options produces only the system separator, which appears in column 1 in the Report File. You can utilize this marker for your own purposes, such as to separate output from different commands.
- Specifying the DOUBLE SPACE format option causes errors because of unwanted blanks between lines.
- Use a comma in front of format options only if you specify COLLECT FILE.
- A Collect File can be unloaded only in its entirety.
- The software checks for the current system separator in any CHARACTER, TEXT, or UNDEFINED value being unloaded. If one occurs, message -290- appears. You can disable this checking with the execution parameter OPT000. You can also request a check for all characters that can be specified as the system separator.

## WHERE-CLAUSE

Selects a subset of a database for retrieval or update commands. A where-clause consists of one or more expressions following the keyword WHERE. Each expression is made up of one or more conditions that an item value must meet.

```
WHERE   expression
```

```
expression    |condition
              |(expression)
              |NOT expression
              |expression AND expression
              |expression OR expression
              |record HAS expression
              |expression AT n
              |SAME

condition     [NON-KEY] item  |unary-operator
                              |binary-operator value
                              |ternary-operator value * value
                              |CONTAINS text
                              |* binary-operator item *
                              |* binary-operator cf-item [(CF)] *
|                             |IN (value, value [, value])
|                             |CTNSIN (value, value [, value])
```

| | |
|---|---|
| **NOT** | finds the complement of specified criteria. |
| **AND** | combines two expressions by finding data records that satisfy both expressions. |
| **OR** | combines two expressions by finding data records that satisfy either expression or both. |
| *record* | is a schema record name or component number. |
| **HAS** | specifies a focal record to qualify data records based on the contents of descendant or ancestor data records. |
| **AT** | specifies a data record by its position under its parent. |
| *n* | is a 0 or a positive integer indicating position. Zero means last position. |
| **SAME** | repeats the where-clause of a previous command with or without additional qualification criteria. |
| *item* | is a schema item name or component number. The item can be key or non-key. |

| | |
|---|---|
| *unary-operator* | EXISTS or FAILS verifies existence or nonexistence of values for an item. |
| *binary-operator* | EQ, NE, GE, GT, LE, or LT |
| *ternary-operator* | EQ, NE, or SPANS compares data values with a range of values. |
| *value* | is a literal value or a system string (*NOW*, *TODAY*, *FTODAY*). Enclose the value with a delimiter of your choice. The default is the slash. Ternary operators require a low value and a high value. |
| **CONTAINS** | searches for characters within an item's value. |
| *text* | uses the following syntax: |

```
[[THE] CHARACTERS]  substring

  [* substring [correlation]]  |[position]
                               |[interval]
```

```
[THE] |WORD    substring  |[position]
      |PREFIX             |[interval]
      |SUFFIX
```

For details on searching for characters using CONTAINS, see *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition.*

| | |
|---|---|
| *cf-item* | is a Collect File item name. The item name may be followed by (CF) to resolve a conflict between a schema item name and a Collect File item name. |
| **IN** | causes the software to create a condition for each specified value. The conditions are connected with the OR operator. Each condition contains the specified item, the EQ operator, and one of the specified values. |
| **CTNSIN** | causes the software to create a condition for each specified value. The conditions are connected with the OR operator. Each condition contains the specified item, the CONTAINS operator, and one of the specified values. The software searches for the characters within the item's value. |

## Examples

```
PRINT LAST NAME WHERE ACCRUED VACATION EXISTS:

PR LAST NAME WH EMPLOYEE STATUS NE FULL TIME OR EMPLOYEE
   STATUS FAILS:

LI C2, C3 WH C13 FAILS AND C4 GE 01/01/91:

PR C2 WH HIRE DATE SPANS 01/01/88*12/31/1990:

PR C2 WH (MAJOR FIELD = MATH OR MINOR FIELD = MATH)
   AND GENDER EQ FEMALE:

PR C2 WH C0 HAS C201 EQ FORTRAN AND C0 HAS C201 EQ COBOL:

PR C101 WH START DATE* = HIRE DATE*:

LIST C102, C103 WHERE C102 EXISTS AT 1:

AS C14 EQ 2311 HANSFORD* WH C411 EQ /COLLEGE OF WILLIAM AND MARY/:

| LI BY ENTRY, C1, C2, C3, C101, C102, C7, C111, OB C2 WH C413
|    GT 06/01/1986 AND (C412 IN (MA, MBA, MS, PHD)).

LI C1, C2, C3 WH LAST NAME CONTAINS MAC:

LI C2, C3, C10, OB C2 WH C10 CONTAINS THE CHARACTERS 100*199 WITH
   MATCH ON 3 CHARACTERS FROM POSITION 7 THRU LAST :

| PR C1, C2 WH LAST NAME IN (SMITH, JONES):

| PR C1, C2 WH LAST NAME CTNSIN (MC, SON, TON):
```

## Rules

- Abbreviations:
  - CHARACTERS = CHARACTER or CHAR
  - CONTAINS = CONT, CONTAIN, or CONTAINING
  - EXISTS = EXIST or EXISTING
  - FAILS = FAIL or FAILING
  - HAS = HAVE or HAVING
  - NON-KEY = NK
  - SAME = SA
  - SPANS = SPAN or SPANNING
  - WHERE = WH
- You must have W-authority for components specified in a where-clause.
- In some forms of the INSERT TREE and MOVE TREE commands, you need the keywords BEFORE or AFTER instead of WHERE.

- You can also use symbols for the following operators:

| Operator | Symbol |
|---|---|
| EQ | = |
| GE | >= or => or ¬< or !< |
| GT | > |
| LE | <= or =< or ¬> or !> |
| LT | < |
| NE | ¬= or != |
| AND | & |
| OR | \| |
| NOT | ¬ or ! |

  Without symbols, the software reads a blank on either side of an operator to separate it from other syntactic units. However, with symbols, blanks are not required for separation. Any blanks encountered following a symbol are considered part of the value. For example, these two where-clauses are equivalent:

```
WHERE C1 EQ ABC AND C2 GT DEF:
WHERE C1=ABC&C2>DEF:
```

  If component names contain any of the symbols, use the component numbers. Also, if you change the system separator to be any of the symbols, syntax errors could occur if you were to specify that symbol in the where-clause.

- Parentheses control the order of processing, and they take precedence over operators.

- Operator precedence is as follows: AT, HAS, AND, OR.

- When all conditions are joined by the same operator, key-conditions are processed first from right to left. Non-key conditions are processed later.

- When a value includes any syntactic unit (such as the command terminator, the system separator, OR, AT, AND, NOT, or a right parenthesis), you must enclose the value with delimiters. The current delimiter must not occur in a value enclosed by a delimiter.

- You can use only items with the unary operators EXISTS and FAILS.

- When comparing two items with a binary operator, both items must belong to related schema records, and both must be either numeric or nonnumeric. The system separator must follow each specified item.

- Use a Collect File item only as the right-hand term of an item-to-item comparison.

- You cannot specify a range of values for the IN and CTNSIN operators. All characters between the commas are treated as a single value. |

- For ternary operators, the low value must be less than or equal to the high value as determined by the collating sequence. See Appendix D.
- If the conditions in a where-clause combined by AND or OR contain components that are disjoint, you must use the HAS operator.
- You can specify more than one AT operator if the operators are separated by parentheses.
- If two AT operators are given, one must be AT 0.
- You must specify the keyword NON-KEY to change a key condition into a non-key condition.
- To specify the SAME operator in a command, you must first issue a QUEST command with a where-clause. You cannot use SAME if it is joined with an unindexed expression by the binary operator OR. See **SAME IS** on page 3-54 for details on the SAME operator.
- Under Security by Entry, the first or only condition in the where-clause must be a key-condition with the EQ operator. If the condition specifying the Entry Key item is not the only condition in the where-clause, it must be joined to the other condition or expression with the AND operator.
- The software guarantees that floating point values are precisely compared. "Fuzzy" logic does not creep into where-clause logic. If you do not know the precise value stored in the database, use the SPANS operator to locate a range of imprecise values.

Chapter 4

# SYSTEM-WIDE COMMANDS

## CHANGING PROCESSORS

The CONTROL, DEFINE, and QUEST commands call the CONTROL, DEFINE, and QUEST processors respectively, so that you can issue commands specific to that processor.

```
CONTROL :

DEFINE :

QUEST : (or ACCESS :)
```

### Example

```
USER, DEMO:
DBN IS EMPLOYEE:                       (automatically calls QUEST)
    .
    .  (QUEST commands)
    .
CONTROL:                                          (calls CONTROL)
    .
    .  (CONTROL commands)
    .
QUEST:                                              (calls QUEST)
    .
    .  (QUEST commands)
    .
EXIT:                          (closes database and ends session)
```

### Rules

- Some commands automatically call a specific processor after execution: DATA BASE NAME IS calls QUEST, RESTORE calls QUEST, NEW DATA BASE IS calls DEFINE, MAP calls QUEST, and GENERATE calls QUEST.
- You cannot issue the commands CONTROL, DEFINE, or QUEST from the REPORT processor or during a QUEUE/TERMINATE session.
- You must be in the QUEST processor to issue the REPORT and QUEUE commands.
- You must be using the master password to call the DEFINE processor.

## COMMAND FILE IS

Immediately switches the software to start reading commands from an alternate Command File.

```
[LOCAL] COMMAND FILE IS  |INPUT    :
                         |fileref
```

| | |
|---|---|
| LOCAL | directs the software to look for the Command File in your TSO region or CMS machine. |
| *fileref* | is the DDname of your alternate Command File. Do not use reserved file names. See Appendix C for the list of reserved file names. |

### Examples

```
COMMAND FILE IS MYCOM:

LOCAL COMMAND FILE IS COM1:
```

### Rules

- The Command File is set to INPUT (which is your terminal or some other input device) at the beginning of each session. You can let SYSTEM 2000 software allocate the default file (S2KCOMD) dynamically instead of specifying it in a CLIST. An S2KCOMD DD statement must be included in batch JCL.
- An alternate file must be described with a DD statement.
- Only one Command File can be open at a time, but you can change to a different one at any time and as often as necessary.
- The Command File and the Data File cannot be set to the same physical file simultaneously. For both to be used, as in loading, one must be an alternate file. You cannot set the Command File to the Message File or the Report File.
- If the software does not find another COMMAND FILE IS or EXIT command at the end of an alternate Command File, it automatically exits the session.
- The Command File must contain fixed-length records, which cannot exceed 250 characters.
- If the alternate Command File is a nonlabeled file, include LABEL=(,NL) and a DCB parameter with LRECL, BLKSIZE, and RECFM in the DD statement for the file.
- Alternate file names must be unique in a Multi-User environment.
- The LOCAL option requires the SYS2KTPI interface to Multi-User software.

## DATA FILE IS

Tells the software to read data from an alternate Data File.

```
[LOCAL] DATA FILE IS  |INPUT    :
                      |fileref
```

| | |
|---|---|
| LOCAL | directs the software to look for the Data File in your TSO region or CMS machine. |
| *fileref* | is the DDname of your alternate Data File. Do not use reserved file names. See Appendix C for the list of reserved file names. |

### Examples

```
DATA FILE IS MYDATA:

LOCAL DATA FILE IS DATA1:
```

### Rules

- The Data File is set to INPUT (which is your terminal or some other input device) at the beginning of each session. You can let SYSTEM 2000 software allocate the default file (S2KCOMD) dynamically instead of specifying it in a CLIST. An S2KCOMD DD statement must be included in batch JCL.
- An alternate file must be described with a DD statement.
- Only one Data File can be open at a time, but you can change to a different one at any time and as often as necessary.
- The Data File and the Command File cannot be set to the same physical file simultaneously. For both to be used, as in loading, one must be an alternate file. You cannot set the Data File to the Message File or the Report File.
- The Data File must contain fixed-length records, which cannot exceed 250 characters.
- If the alternate Data File is a nonlabeled file, include LABEL=(,NL) and a DCB parameter with LRECL, BLKSIZE, and RECFM in the DD statement for the file.
- Alternate file names must be unique in a Multi-User environment.
- The LOCAL option requires the SYS2KTPI interface to Multi-User software.

## DELIMITER IS

Changes the where-clause delimiter. The where-clause delimiter encloses a value in any where-clause condition for QUEST commands, for the DEFINE language MAP command, and for the REPORT language GENERATE command.

```
DELIMITER IS  delimiter  :
```

| | |
|---|---|
| *delimiter* | is a single special character to be used as the where-clause delimiter. The delimiter cannot be used in values, and it cannot be the same as the current system separator or the command terminator. See Appendix A for the list of special characters. |

### Example

```
DELIMITER IS $:
      .
      .
      .
PRINT ENTRY WHERE C303 EQ $CHURCH AND SCHOOL$:
      .
      .
      .
DITTO WHERE C303 GT $SCHOOL/UNIVERSITY$:
```

### Rules

- The default is the slash; it is set at the beginning of each session.
- Once the delimiter is changed, it stays in effect until changed again or until the end of the session.
- You cannot use a where-clause delimiter in a QUEUE language where-clause.
- In a Multi-User environment, avoid conflicting delimiters by establishing rules with all users of a database.

## ECHO

Writes comments or commands to the Report File and directs the software to write input commands to the Message File.

```
ECHO   |ON                                :
       |OFF
       |delimiter  string  delimiter
```

| | |
|---|---|
| *delimiter* | is a single special character to be used as the ECHO string delimiter. The delimiter cannot be the same as the current system separator or the command terminator. See Appendix A for the list of special characters. |
| *string* | defines a string of characters not containing the character used for the delimiter. The maximum is 249 characters. You can issue as many ECHO string commands as needed. |

### Examples

```
ECHO ON:
```
(*echoes commands and results*)

```
ECHO OFF:
```
(*echoes results only*)

```
ECHO/MAP:COMMAND FILE IS INPUT:/:
```
(*writes MAP:COMMAND FILE IS INPUT: on the Report File*)

```
ECHO/THIS IS A COMMENT/:
```
(*writes THIS IS A COMMENT on the Report File*)

### Rules

- The default for a batch job is ECHO ON. The default for an interactive session is ECHO OFF. The defaults are set at the beginning of each session.
- If a string is specified, all characters are written to the Report File. The string is written in 80-character lines.
- If you specify stored strings or functions in an ECHO string, they are not expanded unless they are later read and processed as a Command File. Characters are written to the Report File exactly as encountered.
- If an ECHO string is null (that is, ECHO //:), nothing is written to the Report File.

## ENTRY TERMINATOR IS

Changes the entry terminator word. The entry terminator word signals the end of data for a data tree in a loader stream or in an update command.

```
ENTRY TERMINATOR IS  word  :
```

| | |
|---|---|
| *word* | is a name to be used as the entry terminator word. The name can be from 1 to 4 alphanumeric characters with no blanks. If you use a special character, it must be only one character, and it cannot be the same as the current system separator or the command terminator. See Appendix A for the list of special characters. |

### Example

```
ENTRY TERMINATOR IS FINI:
     .
     .
     .
IT C120*1*1*1 EQ 121*06/30/91*FINI* WH C1 EXISTS:
```

### Rules

- The default entry terminator word is END.
- If the master password holder changes the entry terminator word, it is changed for all users of that database. The current setting is stored in the database.
- If a secondary password holder or the DBA password holder changes the entry terminator word, it is changed only for that database in that session.
- The only reason to change the entry terminator word is for readability. No conflict occurs if END is a value in a value stream.

## EXIT

Closes the database and ends the session.

```
EXIT :
```

### Example

```
EXIT:
```

### Rules

- Issue EXIT as the last command in a session.
- In interactive mode, you must issue EXIT.
- In batch mode, if EXIT is not included or not contained in an alternate Command File, the software automatically issues an EXIT command.
- If you should need to exit a session right after invoking the SYSTEM 2000 software, you can issue EXIT as the first command in a session before issuing the USER command. That is, the first command in a session must be either the USER command or the EXIT command.

## IF...THEN...ELSE

Conditionally executes any sequence of CONTROL, DEFINE, or QUEST commands, and tests system variables.

```
IF  expression  THEN  commands ...

                 [ELSE  commands ...]

                 ENDIF  :
```

```
expression    |condition
              |NOT expression
              |(expression)
              |expression  [AND  expression]
              |expression  [OR  expression]

 condition    |COMPLETE
              |WARNING [OCCURRED]
              |ERROR [OCCURRED]
              |MESSAGE [NUMBER]  binary-operator  message
              |MESSAGE [NUMBER]  ternary-operator  message * message
              |[SELECTED] RECORDS  binary-operator  integer
              |[SELECTED] RECORDS  ternary-operator  integer * integer
              |DATA BASE CYCLE  binary-operator  integer
              |DATA BASE CYCLE  ternary-operator  integer * integer
              |PASSWORD  binary-operator  password
              |PASSWORD  ternary-operator  password * password
              |MASTER PASSWORD
```

| | |
|---|---|
| *binary-operator* | LT, LE, NE, EQ, GT, or GE |
| *ternary-operator* | EQ, SPANS, or NE |
| *message* | is a Self-Contained Facility message number. |
| *integer* | is a positive number. |
| *password* | is a password of 1 to 4 alphanumeric characters with no blanks. |
| *commands* | are Self-Contained Facility commands. Each command must end with a command terminator. |

## Examples

```
IF MSG = 502 THEN RESTORE MYDATA FROM SAS1234:
              ENDIF:

IF W AND MSG EQ 81 THEN ...

IF COMPLETE THEN ...

IF WARNING OCCURRED THEN ...

IF E THEN ...

IF MESSAGE NUMBER EQ 507 THEN ...

IF MSG SPANS 300* 310 THEN ...

IF MSG > 799 THEN ...

IF SELECTED RECORDS =< 100 THEN ...

IF RECS EQ 4 THEN ...

IF MPW THEN ...

IF PASSWORD = ABC THEN ...

IF DB CYCLE GT 375 THEN ...
```

## Rules

- Abbreviations:
  COMPLETE = C
  DATA BASE = DB
  ERROR = E
  MASTER PASSWORD = MPW
  MESSAGE = MSG
  PASSWORD = PW
  RECORDS = RECS
  WARNING = W
- IF...THEN...ELSE statements can be nested up to 32 levels.
- Use the SHOW command to write the system variables to the Message File. See **SHOW** on page 4-18 for details.
- In a Multi-User environment, IF-ENDIF blocks must be completely contained in a single segment.

- You can also use symbols for the following operators:

| Operator | Symbol |
|---|---|
| EQ | = |
| GE | >= or => or ¬< or !< |
| GT | > |
| LE | <= or =< or ¬> or !> |
| LT | < |
| NE | ¬= or != |
| AND | & |
| OR | \| |
| NOT | ¬ or ! |

  Without symbols, the software reads a blank on either side of an operator to separate it from other syntactic units. However, with symbols, blanks are not required for separation. Any blanks encountered following a symbol are considered part of the value.

- If you change the system separator to be any of the symbols, syntax errors could occur if you were to specify that symbol in IF...THEN...ELSE logic.

## INSERTING COMMENTS

A comment can be inserted after any CONTROL, DEFINE, QUEST, QUEUE, or REPORT language command. For example, comments can explain job streams or serve archival purposes.

```
delimiter  comment  delimiter
```

| | |
|---|---|
| *delimiter* | is a single special character to be used as the comment delimiter. The delimiter cannot be the same as the current system separator or the command terminator. Also, the beginning and ending delimiter must be the same. See Appendix A for the list of special characters. |
| *comment* | defines a string of 1 to 250 characters. It cannot contain the delimiter. |

### Examples

```
1*EMPLOYEE NUMBER (INTEGER 9999): &MUST BE FOUR DIGITS&
```

```
PRINT C1, C5:
    .
    .
    .
$NEXT COMMAND COMMENTED OUT FOR TEST RUN$
$MESSAGE FILE IS ABCCMT:$
```

### Rules

- ECHO ON must be in effect to write comments to the Message File.
- Consecutive blanks are counted as one blank toward the 250 character maximum, but single blanks are counted individually.
- You can insert a comment anywhere after a command terminator and before the next command.
- A comment can span line boundaries like a Self-Contained Facility command.

## LOOKUP IS

Provides you some flexibility in changing from one processor to another. If processor lookup is activated and you issue a command that is not valid for the current processor, the software will try to switch you to the appropriate processor.

```
LOOKUP [IS] |OFF :
            |ON
```

### Examples

```
LOOKUP IS ON:

LOOKUP IS OFF:
```

### Rules

- ON activates processor lookup. OFF (the default) disables processor lookup.
- Processor lookup switches to the appropriate processor only if the first keyword in a command is unique to one processor. That is, if you issue PRINT SIZE (a CONTROL processor command) from the QUEST processor, you will remain in QUEST because PRINT is a valid keyword for the QUEST processor.
- Processor lookup functions only while you are using the QUEST or CONTROL processor. The DEFINE, REPORT, and QUEUE processors accumulate commands over the course of a session, so it would not be appropriate to lose any accumulated (but unexecuted) commands because of a change in processors.
- You must explicitly invoke the CONTROL processor in order to issue the RELEASE command. This prevents your releasing a database inadvertently from the QUEST processor.
- Your site can enable or disable processor lookup with the execution option OPT045=YES/NO. The default setting at installation time is OPT045=NO, which disables lookup.

## MESSAGE FILE IS

Tells the software to write messages and echoes of commands to an alternate Message File.

```
[LOCAL] MESSAGE FILE IS  |OUTPUT   :
                         |fileref
```

| | |
|---|---|
| LOCAL | directs the software to look for the Message File in your TSO region or CMS machine. |
| *fileref* | is the DDname of your alternate Message File. Do not use reserved file names. See Appendix C for the list of reserved file names. |

### Examples

```
MESSAGE FILE IS MYMSG:

LOCAL MESSAGE FILE IS S2KMSG:
```

### Rules

- The Message File is set to OUTPUT (which is your terminal or some other output device) at the beginning of each session. You can let SYSTEM 2000 software allocate the default file (S2KMSG) dynamically instead of specifying it in the JCL or CLIST.
- An alternate file must be described with a DD statement.
- If any other SYSTEM 2000 commands appear on the line with this command, the other commands are sent to the current Message File.
- Only one Message File can be open at a time, but you can change to a different one at any time and as often as necessary.
- The Message File cannot be the same file as the Command File or the Data File.
- The Message File must contain fixed-length records, which cannot exceed 140 characters.
- Alternate file names must be unique in a Multi-User environment.
- The LOCAL option requires the SYS2KTPI interface to Multi-User software.

## PROMPT IS

Allows you to choose another system prompt during a session. Instead of the standard three dashes (- - -), the system prompt can be the name of the current processor. Or you can turn the prompt off.

```
PROMPT [IS] |DASH      :
            |PROCESSOR
            |OFF
```

### Example

```
---
USER, DEMO:
---
DBN IS EMPLOYEE:
---
PROMPT IS PROCESSOR:
QUEST>
LIST C1, C2, C3, C4 WHERE C4 GT 01/01/1988:
   (command output)
QUEST>
PROMPT IS OFF:
DESCRIBE:
   (command output)
PROMPT IS ON:
---
```

### Rules

- Abbreviations and Synonyms:
  DASH = DASHES, DASHS, ON, or -
  PROCESSOR = PROC
- PROMPT IS DASH (the default) activates the standard prompt of three dashes. OFF suppresses the system prompt. PROCESSOR means that the prompt is the name of the current processor, for example, CONTROL, DEFINE, REPORT, or QUEST.
- The ECHO command determines whether the prompt is displayed.
- Your site can reset the system prompt with the execution option OPT044=YES/NO. The default setting at installation time is three dashes (OPT044=NO).
- If you have written programs that read SYSTEM 2000 output files and the default prompt at your site was changed to be the processor name, you can change the prompt in your jobs to the standard prompt (- - -) with the PROMPT IS command. This will make SYSTEM 2000 output compatible with your programs.
- For Multi-User batch jobs, the prompt is always three dashes.

## REPORT FILE IS

Immediately switches the software to start writing retrieval command output to an alternate Report File.

```
[LOCAL] REPORT FILE IS  |OUTPUT   :
                        |fileref
```

| | |
|---|---|
| LOCAL | directs the software to look for the Report File in your TSO region or CMS machine. |
| *fileref* | is the DDname of your alternate Report File. Do not use reserved file names. See Appendix C for the list of reserved file names. |

### Examples

```
REPORT FILE IS MYREPORT:

LOCAL REPORT FILE IS RPTOUT:
```

### Rules

- The Report File is set to OUTPUT (which is your terminal or some other output device) at the beginning of each session. You can let SYSTEM 2000 software allocate the default file (S2KMSG) dynamically instead of specifying it in the JCL or CLIST.
- An alternate file must be described with a DD statement.
- If any other SYSTEM 2000 commands appear on the line with this command, output from those commands is sent to the current Report File.
- Only one Report File can be open at a time, but you can change to a different one at any time and as often as necessary.
- The Report File must contain fixed-length records, which cannot exceed 250 characters.
- Alternate file names must be unique in a Multi-User environment.
- The LOCAL option requires the SYS2KTPI interface to Multi-User software.

## SEPARATOR IS

Changes the system separator. The system separator separates labels from values and also marks the end of a value.

```
SEPARATOR IS  symbol  :
```

*symbol* is a single special character to be used as the system separator. The system separator cannot be used in values, and it cannot be the same as the current where-clause delimiter or the command terminator. See Appendix A for the list of special characters.

### Example

```
SEPARATOR IS $:
    .
    .
    .
REMOVE TREE ENTRY $ 4:

PRINT ENTRY WHERE C1 SPANS 1100 $ 1251:
```

### Rules

- The default is the asterisk.
- If the master password holder changes the system separator, it is changed for all users of that database. The current setting is stored in the database.
- If a secondary password holder or the DBA password holder changes the system separator, it is changed only for that database in that session.

## SHOW

Writes system variables to the Message File.

```
SHOW SYSTEM VARIABLES   :
```

### Example

```
SHOW SYS VARS:
```

### Rules

- Abbreviations:
  SYSTEM = SYS
  VARIABLES = VARS
- The system variables are: completion code, message number, database cycle number, and number of selected records.
- You can issue SHOW within or outside an IF...THEN...ELSE statement. See **IF...THEN...ELSE** on page 4-9 for details.

## TIMING

Reports statistics about I/O requests and CPU utilization for commands.

```
TIMING |OFF  :
       |ON
```

### Examples

```
TIMING ON:

TIMING OFF:
```

### Rules

- At the beginning of each session, timing is set to OFF.
- Timing statistics are written to the Message File.

Appendix A

# Special Characters

You can use these special characters in SYSTEM 2000 commands:

| | | |
|---|---|---|
| ¢ | $ | _ |
| . | * | > |
| < | ) | ? |
| ( | ; | # |
| + | ¬ | @ |
| \| | - | ' |
| & | / | = |
| ! | % | " |

Appendix B

# Reserved Words and Characters

You cannot use these reserved words and characters in component names, even when they are preceded and followed by special characters. (For example, -OR- is not acceptable.)

## Words That Cannot Begin a Component Name

| | |
|---|---|
| ALL | MIN |
| ANY | NK |
| AVG | NON-KEY |
| C (as a single letter) | OB |
| C*nnnn* | ORDERED BY |
| COUNT | SA |
| DATA | SAME |
| HIGH | SIGMA |
| LOW | SUM |
| MAX | TREE |

## Words and Characters That Cannot Occur in a Component Name

| | | |
|---|---|---|
| AND | FAILING | OCCURS |
| AT | FAILS | OR |
| BY | FROM | SPAN |
| CONT | GE | SPANNING |
| CONTAIN | GT | SPANS |
| CONTAINING | HAS | TO |
| CONTAINS | HAVE | WH |
| CTNSIN | HAVING | WHERE |
| ELSE | IN | , |
| EQ | LE | ( |
| EXIST | LT | ) |
| EXISTING | MOVE | *system separator* |
| EXISTS | NE | *command terminator* |
| FAIL | NOT | |

In addition, you cannot use the following operator symbols in component names:

| | | |
|---|---|---|
| = | <= | & |
| >= | =< | \| |
| => | ¬> | ¬ |
| ¬< | != | ! |
| !< | | |

You can use the following words in component names, but you must reference the component in a command by its component number.

## Restricted Usage To Begin a Component Name

CNT
OF
RCNT
RESUM

## Restricted Usage Anywhere in a Component Name

FOR
THROUGH
THRU

Appendix C

# Reserved File Names

Following are the reserved file names:

| | | |
|---|---|---|
| S2KSNAP | S2KSYS00 | SF01 |
| S2KMSG | S2KSYS01 | SF02 |
| S2KCOMP | S2KSYS02 | SF03 |
| SYSUDUMP | S2KSYS03 | SF04 |
| SYSABEND | S2KSYS04 | SF05 |
| SYSOUT | S2KSYS05 | SF06 |
| S2KUSERS | S2KSYS06 | |
| TAPES2K | S2KSYS07 | SF11 |
| KEEPFILE | S2KSYS08 | SF12 |
| STEPLIB | | SF13 |
| SYSLIB | S2KSYS10 | . |
| S2KPARMS | S2KSYS11 | . |
| S2KPAD*nn* | S2KSYS12 | . |
| | . | and so forth |
| | . | |
| | . | |
| | and so forth | |

In addition, the following DDnames are reserved:

- the eight DDnames used for the database files to be accessed in the same SYSTEM 2000 session

- the two DDnames used for the Savefile and Keepfile.

A DDname consists of the first seven characters of the database name, plus a suffix. The suffix is a number from 1 through 8 for the database files, a letter S or K for the Savefile or Keepfile. If the database name is less than seven characters long, the DDname uses Xs to fill it out to seven characters. For example, if the database name is AUTO, the DDname for the Savefile is AUTOXXXS.

Appendix D

# Collating Sequence and Character Set

The ascending collating sequence for the EBCDIC (Extended Binary Coded Decimal Interchange Code) character set is provided in this appendix.

| Collating Sequence | Hexadecimal Representation | Symbol | Meaning |
|---|---|---|---|
| ⋮ | | | |
| 64 | 40 | | Space |
| ⋮ | | | |
| 74 | 4A | ¢ | Cent sign |
| 75 | 4B | . | Period, decimal point |
| 76 | 4C | < | Less than sign |
| 77 | 4D | ( | Left parenthesis |
| 78 | 4E | + | Plus sign |
| 79 | 4F | \| | Vertical bar, logical OR |
| 80 | 50 | & | Ampersand |
| ⋮ | | | |
| 90 | 5A | ! | Exclamation point |
| 91 | 5B | $ | Dollar sign |
| 92 | 5C | * | Asterisk |
| 93 | 5D | ) | Right parenthesis |
| 94 | 5E | ; | Semi-colon |
| 95 | 5F | ¬ | Logical NOT |
| 96 | 60 | - | Minus sign, hyphen |
| 97 | 61 | / | Slash |
| ⋮ | | | |
| 107 | 6B | , | Comma |
| 108 | 6C | % | Percent sign |
| 109 | 6D | _ | Underscore |
| 110 | 6E | > | Greater than sign |
| 111 | 6F | ? | Question mark |
| ⋮ | | | |
| 122 | 7A | : | Colon |
| 123 | 7B | # | Number sign |
| 124 | 7C | @ | At sign |

*(continued)*

| Collating Sequence | Hexadecimal Representation | Symbol | Meaning |
|---|---|---|---|
| 125 | 7D | ′ | Apostrophe |
| 126 | 7E | = | Equals sign |
| 127 | 7F | ″ | Quotation marks |
| ⋮ | | | |
| 193 | C1 | A | |
| 194 | C2 | B | |
| 195 | C3 | C | |
| 196 | C4 | D | |
| 197 | C5 | E | |
| 198 | C6 | F | |
| 199 | C7 | G | |
| 200 | C8 | H | |
| 201 | C9 | I | |
| ⋮ | | | |
| 209 | D1 | J | |
| 210 | D2 | K | |
| 211 | D3 | L | |
| 212 | D4 | M | |
| 213 | D5 | N | |
| 214 | D6 | O | |
| 215 | D7 | P | |
| 216 | D8 | Q | |
| 217 | D9 | R | |
| ⋮ | | | |
| 226 | E2 | S | |
| 227 | E3 | T | |
| 228 | E4 | U | |
| 229 | E5 | V | |
| 230 | E6 | W | |
| 231 | E7 | X | |
| 232 | E8 | Y | |
| 233 | E9 | Z | |
| ⋮ | | | |
| 240 | F0 | 0 | |
| 241 | F1 | 1 | |
| 242 | F2 | 2 | |
| 243 | F3 | 3 | |
| 244 | F4 | 4 | |
| 245 | F5 | 5 | |
| 246 | F6 | 6 | |
| 247 | F7 | 7 | |
| 248 | F8 | 8 | |
| 249 | F9 | 9 | |

# Index

# Your Turn

If you have comments or suggestions about SYSTEM 2000 software or *SYSTEM 2000® Quick Reference Guide, Version 12, First Edition*, please send them to us on a photocopy of this page.

Please return the photocopy to the Publications Division (for comments about this book) or the Technical Support Division (for suggestions about the software) at SAS Institute Inc., P.O. Box 200075, Austin, TX 78720-0075.