SYSTEM 2000® CONTROL Language

Version 12
First Edition

The Power to Know.®

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

SYSTEM 2000® CONTROL Language, Version 12, First Edition

ISBN 1-55544-180-7

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

1st printing, August 1991

SAS Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at support.sas.com/pubs or call 1-800-727-3228.

Contents

Using This Book v
Changes and Enhancements xi

1 INTRODUCTION 1-1

SYSTEM 2000 Software Facilities 1-1
Overview of CONTROL Language Commands 1-2

2 CREATING AND ACCESSING DATABASES 2-1

Accessing SYSTEM 2000 Software: USER 2-2
Creating New Databases: NEW DATA BASE IS 2-3
Opening Existing Databases: DATA BASE NAME IS 2-6
Dynamically Allocating Database Files: ALLOC 2-8
Dynamically Deallocating Database Files: FREE 2-13

3 REORGANIZING AND RECONSTRUCTING DATABASES 3-1

Reorganizing Databases: REORGANIZE 3-1
Reconstructing Databases: RELOAD 3-4

4 CREATING AND REMOVING INDEXES 4-1

Introduction 4-1
Changing an Item from Non-Key to Key: CREATE INDEX 4-2
Changing an Item from Key to Non-Key: REMOVE INDEX 4-4
Massive REMOVE TREE Operations 4-6

5 SAVING AND RESTORING DATABASES 5-1

Why Save Your Database? 5-1
Saving a Database: SAVE 5-4
Releasing a Database: RELEASE 5-11
Restoring a Database: RESTORE 5-12
Transferring Updates: KEEP 5-18
Applying Logged Updates: APPLY 5-22
Suspending Update Logging: SUSPEND 5-26
Examples of the Save/Restore Process 5-28

6 RECOVERING DATABASES WITH COORDINATED RECOVERY 6-1

How Does Coordinated Recovery Work? 6-1
When is Coordinated Recovery Activated? 6-4
The Rollback Log 6-5
The Update Log 6-6
Synchpoints and Logical Units of Work 6-7
Enabling the Rollback Log: ENABLE/DISABLE ROLLBACK 6-11
Resetting the Rollback Log: RESET ROLLBACK AFTER 6-13

7 ENSURING DATABASE SECURITY 7-1

Introduction 7-1
Assigning the Master Password: USER 7-3
Assigning Secondary Passwords: VALID PASSWORD IS 7-6
Assigning and Changing Authorities: ASSIGN Authorities 7-7
Assigning Security by Entry: ENTRY KEY IS 7-11
Assigning the DBA Password: DBA PASSWORD IS 7-14
Controlling DBA Password Commands: ENABLE/DISABLE DBA FOR 7-15
Invalidating Passwords: INVALID PASSWORD IS 7-16
Changing Passwords: CHANGE PASSWORD 7-17
Listing Passwords: LIST PASSWORDS 7-18
Listing Secondary Passwords and Authorities: LIST PASSWORDS AND AUTHORITIES 7-19
Listing the DBA Password: LIST DBA 7-20
Listing the DBA Password and Commands: LIST DBA AND COMMANDS 7-21

8 CONTROLLING NON-KEY WHERE-CLAUSE PROCESSING: ALLOW/NO FULL PASSES 8-1

9 OBTAINING DATABASE SIZE STATISTICS: PRINT SIZE 9-1

10 CONTROLLING USER EXITS: ENABLE/DISABLE ROUTINE 10-1

11 CONTROLLING MULTIPLE LOCAL HOLDS: ENABLE/DISABLE MULTIPLE HOLDS 11-1

APPENDIX A: CONTROL LANGUAGE COMMANDS UNDER MULTI-USER SOFTWARE A-1

Index X-1

Using This Book

Purpose

This book provides both reference information and examples for the CONTROL language of SYSTEM 2000 software, Version 12. SYSTEM 2000 software is SAS Institute's data management system for mainframe computers running under MVS and CMS operating systems.

This book is for SYSTEM 2000 users who administer databases, and it assumes you have Version 12 or later of SYSTEM 2000 software.

Organization of This Book

This book has reference chapters and an appendix. The following sections list the chapters.

Reference The Reference chapters describe the syntax of CONTROL commands in detail and provide examples.

Chapter 1, "Introduction"

Chapter 2, "Creating and Accessing Databases"

Chapter 3, "Reorganizing and Reconstructing Databases"

Chapter 4, "Creating and Removing Indexes"

Chapter 5, "Saving and Restoring Databases"

Chapter 6, "Recovering Databases with Coordinated Recovery"

Chapter 7, "Ensuring Database Security"

Chapter 8, "Controlling Non-Key Where-Clause Processing: ALLOW/NO FULL PASSES"

Chapter 9, "Obtaining Database Size Statistics: PRINT SIZE"

Chapter 10, "Controlling User Exists: ENABLE/DISABLE ROUTINE"

Chapter 11, "Controlling Multiple Local Holds: ENABLE/DISABLE MULTIPLE HOLDS"

Appendix The appendix provides a chart showing the effects of CONTROL language commands in a Multi-User environment.

Appendix A, "CONTROL Language Commands under Multi-User™ Software"

Reference Aid

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

Index	identifies pages where specific topics and terms 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. In margins, indicates a technical change in the text for the latest version of the software.
. . . (ellipsis) . .	indicates that the previous syntax can be repeated. In examples, either vertical or horizontal ellipses indicates an omitted portion of output or a command.
b	indicates a significant blank in syntax or output. Generally, spaces indicate blanks, and the *b* 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 Quick Reference Guide, Version 12, First Edition* (order #A55020) contains syntax, usage rules, and examples for all DEFINE, CONTROL, QUEST, and system-wide commands. Commands are organized alphabetically within each language, and the pages are color-coded for easy reference.

- *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 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 made to the CONTROL language of SYSTEM 2000 software for Version 12. This information is intended for users who have previous experience with the CONTROL language.

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.

Dynamically Allocating Database Files

SYSTEM 2000 software dynamically allocates database files if they are not already allocated in the JCL or CLIST. The new ALLOC command allows you to specify dynamically the data set name, disposition, unit, space, and so on within an SCF session. If the existing database files use only defaults, you do not need the ALLOC command. For a new database, SYSTEM 2000 software needs only the name of the new database in the ALLOC command in order to allocate the files with the default settings. You can also deallocate the files with the new FREE command. Dynamic allocation of database files means you no longer have to include the database DD statements in the Multi-User initialization step or in a single-user job.

The new ALLOC, FREE, and PREFIX execution parameters control dynamic database file allocation. These parameters are discussed in the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.

Savefiles and Keepfiles can also be dynamically allocated. The syntax for the SAVE and RESTORE commands is simpler than in earlier releases. The single word SAVE allocates a Savefile and copies the current database. SAVE followed by a slash activates update logging (SAVE/). In this situation, the Keepfile is allocated dynamically when needed if it is not already allocated. The Savefile is now a QSAM file. SAVE and RESTORE syntax contains options for specifying the data set name, volume, unit, and so on if you do not want to use the defaults for dynamic allocation.

The Savefile and Keepfile DDnames are the first seven characters of the database name (with Xs replacing blanks) plus a suffix of S or K, respectively. The new DDnames allow several users to save and restore (and keep updates) several different databases at the same time.

In Version 12, TAPES2K and KEEPFILE are no longer honored as DDnames. Also, all update logging is indirect mode. Direct mode is no longer supported, and the keywords INDIRECT and DIRECT have been dropped from the SAVE and RESTORE command syntax.

Dynamically Allocating User Files

SYSTEM 2000 software dynamically allocates the Command File, Data File, Message File, and Report File if it does not find the S2KMSG or S2KCOMD DDnames in the JCL or CLIST. (See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition*.)

Dynamically Allocating Work Files

SYSTEM 2000 software can now dynamically allocate all work files, that is, scratch pads, sort files, S2KUSERS file, S2KPARMS file, S2KSNAP file, S2KOUTP file, and so on if they are not already allocated in the JCL or CLIST. Eight new execution parameters are available to change the default scratch pad and sort file allocation parameters. Allocation of work files is discussed in the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.

Resetting the Rollback Log

In Version 12, SYSTEM 2000 software resets the Rollback Log if no one is doing updates. To take advantage of this feature, commit your updates before you obtain another local hold. If a reset is requested, the software can grant the reset without having to suspend users.

Maximum Number of Components in a Database

The maximum number of components is 10,000. This number includes stored strings and functions, as well as schema items and schema records. The default maximum in the NEW DATA BASE IS command is still 430.

Maximum Database Cycle Number

The maximum database cycle number is 2,147,483,647.

Chapter 1

Introduction

SYSTEM 2000 SOFTWARE FACILITIES 1-1

OVERVIEW OF CONTROL LANGUAGE COMMANDS 1-2

SYSTEM 2000 SOFTWARE FACILITIES

SYSTEM 2000 software is a general purpose database management system. SYSTEM 2000 software has two major facilities: the Self-Contained Facility (SCF) and the Programming Facility (PLEX).

The Self-Contained Facility offers five basic languages for defining, accessing, and controlling databases. You can call any of these languages during a SYSTEM 2000 session:

DEFINE	The DEFINE language allows you to define and redefine a SYSTEM 2000 database.
CONTROL	The CONTROL language is for administrative functions, such as saving and restoring databases, assigning passwords, and maintaining an Update Log or a Rollback Log.
QUEST	The QUEST language allows you to access a database for retrievals and updates.
QUEUE	The QUEUE language allows you to group updates and retrievals into batches of commands to process large runs more efficiently.
REPORT	The REPORT language allows you to produce reports with running subtotals, special formats, and logical pages within the physical page.

The PLEX (Programming Language Extension) Facility allows you to access and update SYSTEM 2000 databases within a COBOL, FORTRAN, PL/I, or Assembler program.

SYSTEM 2000 software also features QueX and Genius, which can be used to access databases. QueX provides menus for viewing, modifying, or entering data in a SYSTEM 2000 database. Genius is an interactive report writing system that prompts you for report specifications and then displays database information in columnar format.

In addition to these facilities, SYSTEM 2000 software offers many system support functions and tools for fine-tuning the execution configuration at your site. These options include multiple user access, coordinated recovery of multiple databases, accounting information, performance statistics, and user-tailored configurations.

OVERVIEW OF CONTROL LANGUAGE COMMANDS

CONTROL language commands perform a variety of administrative functions for SYSTEM 2000 databases. CONTROL language commands can only be issued from the CONTROL processor, which you can call with the system-wide command CONTROL.

Here is a general overview of the CONTROL language commands.

ALLOC	allocates database files dynamically within an SCF session.
ALLOW/NO FULL PASSES	allows or prohibits complete passes through the Data Table.
APPLY	applies updates recorded on the Keepfile to the restored database.
ASSIGN authorities	allows the master password holder to assign, remove, and change access authorities for secondary password holders.
CHANGE PASSWORD	allows the master password holder to change the master password, the secondary passwords, and the DBA password.
CREATE INDEX	changes a non-key item to a key item and creates an index of the values.
DATA BASE NAME IS	opens an active database.
DBA PASSWORD IS	allows the master password holder to assign a DBA password.
ENABLE/DISABLE DBA FOR	allows the master password holder to authorize or remove the use of certain commands for the DBA password holder.
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/DISABLE ROLLBACK	enables or disables the Rollback Log for a database, which allows or disables Coordinated Recovery for that database.
ENABLE/DISABLE ROUTINE	specifies which user exits are to be enabled or disabled.
ENTRY KEY IS	allows the master password holder to restrict the access of secondary password holders.
FREE	deallocates database files that were dynamically allocated.
INVALID PASSWORD IS	allows the master password holder to remove secondary passwords and the DBA password.
KEEP	transfers updates from the Update Log to the Keepfile.
LIST DBA	displays the DBA password.

LIST DBA AND COMMANDS	displays the DBA password and a subset of authorized commands that the DBA password holder can issue.
LIST PASSWORDS	displays valid secondary passwords.
LIST PASSWORDS AND AUTHORITIES	displays valid secondary passwords and their authorities.
NEW DATA BASE IS	assigns a name for a new database.
PRINT SIZE	provides statistical information about the size of a database.
RELEASE	makes the disk files of an active database unavailable until the database is restored.
RELOAD	reconstructs the database files.
REMOVE INDEX	changes a key item to a non-key item and removes the index of the values.
REORGANIZE	compacts and reorders the Index.
RESET ROLLBACK AFTER	sets the amount of data to be written to the Rollback Log for Coordinated Recovery.
RESTORE	copies a saved database onto the active database disk files.
SAVE	copies an active database from the disk files onto a Savefile.
SUSPEND	stops the recording of processed updates on the Update Log.
USER	accesses SYSTEM 2000 software and sets the current password for the database to be created or accessed.
VALID PASSWORD IS	allows the master password holder to assign secondary passwords.

Each command consists of the command syntax followed by a command terminator (usually a colon). You can enter multiple commands on one input line. For example:

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

The following chapters describe each CONTROL language command.

Chapter 2

Creating and Accessing Databases

ACCESSING SYSTEM 2000 SOFTWARE: USER 2-2

CREATING NEW DATABASES: NEW DATA BASE IS 2-3
Allocating New Database Files 2-3
Maximum Number of Components 2-4

OPENING EXISTING DATABASES: DATA BASE NAME IS 2-6

DYNAMICALLY ALLOCATING DATABASE FILES: ALLOC 2-8
Existing Database Files 2-10
New Database Files 2-11
Examples 2-11

DYNAMICALLY DEALLOCATING DATABASE FILES: FREE 2-13
Example 2-13

This chapter describes how to access SYSTEM 2000 software, how to name and create a new database, and how to open existing databases. It also describes how you can dynamically allocate and deallocate database files within an SCF session using the ALLOC and FREE commands, instead of allocating them with JCL or a CLIST.

Dynamic allocation is available in single-user jobs and in the Multi-User environment. With this feature, you can even let SYSTEM 2000 software allocate database files automatically with default sizes. Dynamic allocation is especially significant in the Multi-User environment, because you do not have to predetermine which databases to allocate when Multi-User is initialized.

Earlier versions of SYSTEM 2000 software required allocations in the Multi-User initialization job for all databases that might possibly be used in the Multi-User session. With Version 12 and later releases, SYSTEM 2000 software will honor the JCL or CLIST allocations if they exist, but they are not required.

For more information about dynamic database file allocation, see **Dynamically Allocating Database Files: ALLOC** on page 2-8.

You can also let SYSTEM 2000 software dynamically allocate the Command File, Data File, Message File, and Report File, without your having to allocate S2KCOMD and S2KMSG in the JCL or CLIST, except for non-TSO jobs. For details, see *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition.*

In addition, you do not have to allocate scratch pads, sort files, and other files such as the S2KPARMS file. You can let SYSTEM 2000 allocate them dynamically. For details about dynamic allocation of work files, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition.*

ACCESSING SYSTEM 2000 SOFTWARE: USER

The USER command accesses SYSTEM 2000 software and sets the password for the database to be created or accessed.

Format

```
USER, password :
```

password is a password using 1 to 4 alphanumeric characters with no blanks. The password can also be a single special character or lowercase letter. See the SEPARATOR IS command in *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for the list of special characters.

To access SYSTEM 2000 software, issue the USER command. When defining a new database, the password you specify becomes the master password. See **Assigning the Master Password: USER** on page 7-3 for details about setting the master password. For an active database, the password you specify must be the master password, a secondary password, or the DBA password.

The USER command permits access to only one database at a time; however, if the password you specify with the USER command is valid for another database, you can access the other database by issuing another DATA BASE NAME IS command. The open database is closed before the next database is opened.

To access a different database that does not share the current password or to use a different password for the same database, issue the CONTROL command to call the CONTROL processor and then reissue the USER command.

Examples

Example 1

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

Example 2

```
USER, +:
| ALLOC OIL SPILLS,DISP=NEW;
NEW DATA BASE IS OIL SPILLS:
```

CREATING NEW DATABASES: NEW DATA BASE IS

The NEW DATA BASE IS command assigns a name to the new database that you are defining. Use the optional *n* syntax to specify the maximum number of components that can be defined for a database. To allocate the new database files dynamically, issue the ALLOC command before you issue the NEW DATA BASE IS command.

Format

```
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 an MVS or CMS environment.

n is an optional number from 1 to 10,000 specifying the maximum number of components (including C0 component, strings, and functions) that can be defined. The default is 430.

The NEW DATA BASE IS command assigns a name to a new database that you are defining. The database name is stored in the Master Record of the named database. After you issue the NEW DATA BASE IS command, the database can be referenced only by using the assigned name. This command causes exclusive use of the database, and SYSTEM 2000 software assigns the DEFINE processor.

The password you specify with the USER command issued prior to a NEW DATA BASE IS command becomes the master password for the new database. The master password is stored in the Master Record of the named database.

You can use any character in the standard character set in a database name, except for those listed below:

Symbol	Hex
/	61
:	7A
=	7E

To change the name of a database, see **Restoring a Database: RESTORE** on page 5-12.

Allocating New Database Files

You can allocate new database files dynamically within your SCF session with the ALLOC command. The ALLOC command avoids having to allocate the files in the JCL or CLIST, and, optionally, lets SYSTEM 2000 software assign default parameter values. In a Multi-User environment, you can use the ALLOC command within your job instead of having to

Abbreviation: NEW DATA BASE = NDB

| remember to allocate the files when Multi-User is initialized. If SYSTEM 2000 software finds
| they are already allocated, it will use those allocations. For details about allocating and
| freeing database files dynamically, see **Dynamically Allocating Database Files: ALLOC** on
| page 2-8 and **Dynamically Deallocating Database Files: FREE** on page 2-13.

The software uses the first seven characters of the database name as part of the DDname for the database files. Any restrictions imposed by the operating system on DDnames also apply to the characters used in the database name.

The database name is stored in the Master Record for the database. Because the first seven characters in a database name are used to construct the DDname, they must be unique for each database. For example, for a database named PERSONNEL, the DDnames are //PERSONN*n*, where *n* equals an integer from 1 to 8 representing database Files 1 through 8. Therefore, you could not access a PERSONNAL database, because the Master Record already contains PERSONNEL. In addition, you could not define a PERSONNAL database, because DD statements necessary for the //PERSONN*n* database files would already exist in the operating system.

Database names do not need to be compatible with the naming conventions of any procedural language that will use the database in PLEX programs. You can resolve any conflicts that arise in the PLEX program by using an alias name compatible with the procedural language naming conventions. The only exception is that the characters PARM, PARAMETER, or PARA cannot be used in a database name that will be used in a PL/I PLEX program.

For details about these naming conventions, see the appropriate IBM programming language manual. For details regarding the use of an alias name, see the *SYSTEM 2000 PLEX Manual, Version 12, First Edition*.

Maximum Number of Components

The optional *n* syntax for the NEW DATA BASE IS command sets the maximum number of components for a database. This limit is stored in the database Master Record and reflects the expected definition growth over the life of the database.

| The maximum number of components allowed for a database is 10,000. Any limit you specify (or the default) includes the number of strings and functions that can be defined and the C0 (ENTRY) component. If you do not use the *n* option, the default limit is 430 components.

Using a smaller number of components allows the database definition to be assigned to a smaller segment of main memory when the database is in use, provided the limitation is coordinated with the SDBS, SDBSIZE, LDBS, and LDBSIZE execution parameters.

For small databases, the SDBS parameter specifies the initial number of database definition blocks to be acquired at run time, and the SDBSIZE parameter specifies the maximum number of components that can be contained. For large databases, the LDBS parameter specifies the initial number of database definition blocks to be acquired at run time, and the LDBSIZE parameter specifies the maximum number of components that can be contained. For details about these execution parameters, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.

The limit set for the number of definable components, including the default, cannot be changed unless you recreate the database. To recreate a database, you must follow these steps:

1. Unload the database using the UNLOAD command.
2. Release the database using the RELEASE command.
3. Issue the NEW DATA BASE IS command with the new maximum.
4. Redefine the database.
5. Load the previously unloaded data into the newly defined database using the QUEST or PLEX LOAD commands.

When you are determining the maximum number of definable components, consider your long-term needs, especially when trying to anticipate the number of strings and functions that will be stored.

Examples

```
NEW DATA BASE IS EMPLOYEE:    (uses the default of 430 components)

NDB IS EMPLOYEE/1200:

NEW DATA BASE IS FREIGHT CAR:

NDB IS FREIGHT-CAR:

NEW DATA BASE IS QA RECORDS/700:

NDB IS CROSSREF/125:
```

OPENING EXISTING DATABASES: DATA BASE NAME IS

The DATA BASE NAME IS command opens an existing database.

Format

```
[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.

In order for a database to be accessed with the DATA BASE NAME IS command, it must have been previously named using the NEW DATA BASE IS command; however, it is not necessary for the database to be defined or populated. The password you specify with the USER command issued prior to the DATA BASE NAME IS command must be the master password, a valid secondary password, or the DBA password for the database.

When you open a database with the DATA BASE NAME IS command, SYSTEM 2000 software provides the QUEST processor. The EXCLUSIVE option provides exclusive use of a database; no other user is permitted to use the database until you exit the database. If the database is unavailable or the password is not valid, SYSTEM 2000 software issues an error message and the database is not opened.

You can open only one database at a time in an SCF session. To access a different database, issue another DATA BASE NAME IS command. The DATA BASE NAME IS command closes the open database and opens the database specified, providing that the current password is valid for the database. If you need to specify a different password, issue the CONTROL command to call the CONTROL processor, and then reissue the USER command with the different password.

| You can let SYSTEM 2000 software allocate the files dynamically, or you can allocate the
| files within your SCF session with the ALLOC command. Using the ALLOC command avoids
| having to allocate them with JCL or a CLIST. In a Multi-User environment, dynamic
| allocation avoids having to allocate the files when Multi-User is initialized. If SYSTEM 2000
| software finds the files are already allocated, it will use those allocations. For details about
| allocating and freeing database files, see **Dynamically Allocating Database Files: ALLOC** on page 2-8 and **Dynamically Deallocating Database Files: FREE** on page 2-13.

Abbreviations:
EXCLUSIVE = EXCL
DATA BASE NAME = DBN

Examples

```
DBN IS EMPLOYEE:

EXCLUSIVE DATA BASE NAME IS BILL OF MATERIAL:

EXCL DBN IS STATUS:
```

DYNAMICALLY ALLOCATING DATABASE FILES: ALLOC

The ALLOC command allocates new or existing database files dynamically within an SCF session. Allocations for the database files can be omitted from your JCL or CLISTs.

Format

```
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 any valid IBM data set name. Use quotes around the data set name if you do not want the PREFIX execution parameter value or the TSO prefix concatenated to the beginning of the data set name. The default data set name is the prefix, followed by the first seven characters of the database name (with embedded blanks replaced by Xs) and a suffix of 1 through 8 that represents the database file number. For single-user TSO sessions, the TSO user-prefix is used if the PREFIX execution parameter is not specified. For details about the PREFIX execution parameter, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.

DISP is 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, where *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

(continued on next page)

(continued)

Option	Description	Default
SPACE=	SMALL or LARGE	SMALL
SPC*n*=	(TRK | CYL,*prim*,*sec*)	the SPACE values

The FILES option determines which database files will be allocated for the new database.

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))

You can allocate database files dynamically when you issue the NEW DATA BASE IS command or the DATA BASE NAME IS command, and the software will search for existing files or generate the new files. If the files were varied offline, dynamic allocation cannot occur unless you issue the VARY ONLINE console command.

You can also set up a database allocation table (the S2KDBCNT file) to control access to existing database files and to provide the data set names and disposition. All files will have DISP=OLD when dynamically allocated unless you specify SHR. The S2KDBCNT file is especially useful in a Multi-User environment. In your SCF session, you specify only the name of the existing database, and the rest of the information comes from the S2KDBCNT table. In addition, the VARY operator console command is available to flag any database in the S2KDBCNT file as being offline.

For more details about the format of the S2KDBCNT file, the VARY console command, and the ALLOC execution parameter, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.

If you have not allocated the files in your JCL or CLIST, you must use the ALLOC command in the following situations:

- when you want to allocate new database files.
- when you want to use existing database files and the data set names do not conform to DSN=*prefix.database-name*.
- when you want to assign DISP=SHR.
- when the ALLOC execution parameter is set to YES and you want to override the data set name or the file disposition that is specified in the S2KDBCNT table. For information about setting the ALLOC execution parameter, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.

Note: All files in the S2KDBCNT table have DISP=OLD when they are dynamically allocated unless SHR is specified.

Savefiles and Keepfiles are not allocated at this time, but they use the same data set name with a suffix of S or K, unless you use the DSN option on the SAVE command. For more details, see **Saving a Database: SAVE** on page 5-4 and **Restoring a Database: RESTORE** on page 5-12.

Considerations

- You can issue one or more ALLOC commands at any time before opening a database. That is, specify your allocation information before you issue the NEW DATA BASE IS or DATA BASE NAME IS command.

- The ALLOC execution parameter controls dynamic allocation of database files. If ALLOC=YES, you can issue the ALLOC command at any time, or you can use the S2KDBCNT table to allocate existing files. If ALLOC=TBL, dynamic allocation is restricted to only those databases that exist in the S2KDBCNT table. If ALLOC=NO, you cannot use the ALLOC command or the S2KDBCNT table; that is, you cannot dynamically allocate database files. For more information on setting up the ALLOC execution parameter, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.

Existing Database Files

For dynamic allocation of existing database files, the data set names must fit the default naming convention. If the data set names fit the default convention and you want DISP=OLD, you do not need an ALLOC command; the software will look for the existing files automatically when you issue the DATA BASE NAME IS command. If you want to open the files in shared mode, you must specify DISP=SHR.

The default data set names and disposition are as follows:

DSN=*prefix.database*,**DISP=OLD**

where *database* is the first seven characters of the database name (with embedded blanks replaced by Xs), followed by an integer from 1 through 8.

SYSTEM 2000 software tries to locate existing database files with the steps given below. Files 1 through 6 are required, although the software searches for File 7 and 8 also.

- Sees whether the files were allocated in the JCL or CLIST; if so, uses those allocations.

- Gives user exit EXIT03 a chance to allocate the files dynamically.

- Looks for permanent data sets, using the default data set names or the DSN specified in the ALLOC command.

At this point, if the files are still not allocated, an error message appears: -766-DATA BASE OPEN FAILED FOR *database*. You will also receive an error message if the files are already allocated and you try to give an ALLOC command.

New Database Files

If you want to create new database files with the ALLOC command, use DISP=NEW. Also, if the files are cataloged but empty, you can issue the ALLOC command.

Here is an example of the minimum syntax required for new database files:

```
USER,NEW:
ALLOC NEWDB,DISP=NEW:
NDB IS NEWDB:
```

These commands produce the six database files with the following characteristics:

DSN	BLKSIZE	SPACE	UNIT	DSORG
prefix.NEWDBXX1	6216	3,5 TRK	SYSDA	DA
prefix.NEWDBXX2	6216	1,5 CYL	SYSDA	DA
prefix.NEWDBXX3	6216	3,5 TRK	SYSDA	DA
prefix.NEWDBXX4	6216	3,5 TRK	SYSDA	DA
prefix.NEWDBXX5	6216	1,5 CYL	SYSDA	DA
prefix.NEWDBXX6	6216	1,5 CYL	SYSDA	DA

To change any of the default attributes, specify the appropriate options in the ALLOC command before you issue the NEW DATA BASE IS command. Notice that by default, the Update Log (File 7) and the Rollback Log (File 8) are not dynamically allocated. You must explicitly specify them in the ALLOC command or set 'FILES=ALL'.

The software first checks to see whether the database files are already allocated in the single-user job or Multi-User initialization job. If so, you will receive an error message if you attempt to issue an ALLOC command.

Examples

The following examples illustrate using the ALLOC command to dynamically allocate both new and existing database files.

Example 1

This ALLOC command shows that the only information needed to dynamically allocate existing database files is the database data set names and the disposition. OLD is the default. This example uses DISP=SHR to indicate shared mode.

Note: This information for existing databases can come from the S2KDBCNT table, which SYSTEM 2000 software searches if you do not issue an ALLOC command.

```
ALLOC EMPLOYEE,DSN='S2K.EMPLOYE',DISP=SHR:
```

Example 2

This example illustrates the simplest ALLOC command for allocating new database files. If you do not allocate the new files with JCL or a CLIST, you must give the new database name and a disposition of NEW. This example allocates database Files 1 through 6 for the new EMPLOYEE database, using the defaults for the new database options.

```
ALLOC EMPLOYEE,DISP=NEW:
```

Example 3

This example shows allocating a new database EMPLOYEE with a data set name that is different than the default data set name. It also requests the specific disk pack SAS834.

```
ALLOC EMPLOYEE,DSN='S2K.EMPLOYE',VOL=SAS834,DISP=NEW:
```

Example 4

This example requests dynamic allocation of all eight database files for the new database EMPLOYEE. It also requests a specific disk pack, SAS834.

```
ALLOC EMPLOYEE,FILES=ALL,VOL=SAS834,DISP=NEW:
```

Example 5

This example specifies a 23464 block size for Files 1 through 6 of the new database EMPLOYEE.

```
ALLOC EMPLOYEE,BLK=23464,DISP=NEW:
```

Example 6

This example allocates new database Files 1 through 6 for database EMPLOYEE. By default, the block size for Files 1 through 4 is 6216. This ALLOC command specifically requests a block size of 23464 for Files 5 and 6.

```
ALLOC EMPLOYEE,BLK5=23464,BLK6=23464,DISP=NEW:
```

Example 7

This example allocates the new EMPLOYEE database files (1 through 6) with the space parameter set to LARGE.

```
ALLOC EMPLOYEE,SPACE=LARGE,DISP=NEW:
```

Example 8

This example is similar to Example 7 in that it allocates new EMPLOYEE database Files 1 through 6 and specifies SPACE=LARGE. However, this ALLOC command also requests an even larger space for File 6, with the specific option SPC6=(CYL,100,10).

```
ALLOC EMPLOYEE,SPACE=LARGE,SPC6=(CYL,100,10),DISP=NEW:
```

DYNAMICALLY DEALLOCATING DATABASE FILES: FREE

The FREE command closes the specified database and deallocates Files 1 through 8.

Format

```
FREE database :
```

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

The FREE command allows you to explicitly deallocate the files before the SCF session ends. After a FREE command, message -815- COMMAND INVALID, NO DATA BASE OPENED- appears if you issue an SCF command for the specified database.

Note: Savefiles and Keepfiles that were dynamically allocated are deallocated automatically when a SAVE, RESTORE, APPLY, or KEEP command completes.

Considerations

- FREE deallocates only database files that were previously allocated dynamically. In a Multi-User environment, you must have exclusive use of the database or issue the FREE command before the DATA BASE NAME IS command when no one is using the database.

- 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. Multi-User jobs cannot access the offline database until the console operator issues a VARY ONLINE command to bring the database online again. Making a database offline gives single-user jobs a chance to access the database, for example, for long save or load operations. For more details about the VARY console command, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.

- The FREE execution parameter can prevent SYSTEM 2000 software from deallocating database files that were dynamically allocated. If FREE=YES, the software deallocates those database files at the end of the SCF session. Also, the software deallocates the files if you switch databases in an SCF session. If FREE=NO, the files are not deallocated in either of those situations. The FREE execution parameter does not affect your explicitly issuing the FREE command for a database. For more details about the FREE execution parameter, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.

Example

```
FREE EMPLOYEE:
```

Chapter 3

Reorganizing and Reconstructing Databases

REORGANIZING DATABASES: REORGANIZE 3-1

RECONSTRUCTING DATABASES: RELOAD 3-4

If a large database has experienced massive updates or incremental loads of new data, you might want to use the REORGANIZE or RELOAD commands. REORGANIZE compacts the Index tables. RELOAD unloads the database and rebuilds the database files, which eliminates unwanted space due to deletions and creates the files as though they were a newly created database. Reloading a database automatically rebuilds the Index tables, so you do not need to reorganize the newly created files.

REORGANIZING DATABASES: REORGANIZE

The REORGANIZE command compacts and reorders the Index. Use this command when entries for a database become scattered due to incremental loads or a high volume of updates.

Only the master password holder or a DBA password holder with authorization for this command can issue the REORGANIZE command.

Format

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

When database tables are constructed during the first loading of data, the entries in the Index are ordered in their most optimized state. If the database is relatively static (few or no updates or additions for key item values), the database remains in this organized state. However, if the values for key items are dynamic (many additions, removals, and changes), the I/O activity for a database can increase. Increased I/O activity usually indicates that entries have become scattered; it may be time to reorganize the Distinct Values Table (DVT) or the Multiple Occurrence Table (MOT) or both. For information about the tables, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition.*

Abbreviation: REORGANIZE = REORG

When you issue the **REORGANIZE DVT** command, pages in the Distinct Values Table are physically rearranged on the disk, and upper level indexes are rebuilt. The only efficiency gained from reorganizing the Distinct Values Table may be that of contiguous disk read access in a single-user session. You might issue the REORGANIZE DVT command if your database contains a lot of key item values that are distinct or occur infrequently, for example, employee numbers, gender, or last names.

If REORGANIZE DVT has no processing to do, the file sizes do not change. However, if REORGANIZE DVT does some reorganization, the next available Distinct Values Table page is temporarily used as a work space. After the reorganization process completes, the extra page is included in the "page-in-use" count for the Distinct Values Table, as displayed in the PRINT SIZE command statistics. The page is available for future use when needed. As a result of this processing, the "pages-in-use" count increases even though many removals may have been done previously. However, the "pages-in-use" counter cannot decrease for REORGANIZE DVT. When the MOT option is used, the counter can increase or decrease.

When you issue the **REORGANIZE MOT** command, the Multiple Occurrence Table is reorganized. The scattered blocks of pointers for each distinct value are consolidated into one or several contiguous blocks of maximum size. Reorganization of the Multiple Occurrence Table is effective if updates to the database involve items for which a single value occurs many times (for example, gender, city, state).

When you issue the **REORGANIZE INDEX** (or **ALL**) command, both the Distinct Values Table and the Multiple Occurrence Table are reorganized as discussed for the REORGANIZE DVT and the REORGANIZE MOT commands. The REORGANIZE INDEX command is more efficient than issuing two separate REORGANIZE commands (with DVT and MOT), because some of the processing is common to both processes.

When a REORGANIZE command completes successfully, SYSTEM 2000 software provides the CONTROL processor. Rejection of a REORGANIZE command via a batch fatal error ends a batch run. If a REORGANIZE command is rejected in interactive mode, SYSTEM 2000 software issues an error message and provides the CONTROL processor for subsequent commands.

Considerations

- When you issue a REORGANIZE command, the Update Log status and the database cycle number are unchanged.
- If you issue a REORGANIZE command in a Multi-User environment, a hold is placed on the database until the reorganization process completes so that no other user can access the database.
- You cannot issue a REORGANIZE command for a damaged database.
- If there is an error in the Index of a database (for example, a bad pointer), reordering the Index with the REORGANIZE command may fix the problem.
- If you have enabled rollback for the database, the Rollback Log (File 8) is not suspended by a REORGANIZE command. Therefore, you must ensure that File 8 is large enough for the database files being reorganized. (Refer to previous page to see which files are affected by the REORGANIZE command.) Or you can disable rollback during the reorganization processing and enable rollback again after the reorganization is complete.

Examples

```
REORGANIZE DVT:

REORG MOT:

REORG INDEX:
```

RECONSTRUCTING DATABASES: RELOAD

The RELOAD command reconstructs the files of a database.

Only the master password holder or a DBA password holder with authorization for this command can issue the RELOAD command.

Format

```
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 *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for details.

where-clause selects specific data records. The SAME operator cannot be included. See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for details.

The RELOAD command does an internal unload of existing data and then recreates the database tables. The reload process compacts the database tables and eliminates reusable space. RELOAD requires a considerable amount of scratch file space, equivalent to the amount required to do a LOAD of the database.

If you specify a where-clause, the selected logical entries are unloaded, and the reconstructed database size is reduced accordingly. If you do not specify a where-clause, the entire database is reloaded. If you specify an ordering-clause, the selected logical entries are ordered as specified before reloading occurs. The RELOAD command processes entire logical entries only; that is, the RELOAD command cannot reload subtrees (partial logical entries). However, a level 0 item may be a complex item involving lower-level items. For example, RELOAD ORDERED BY COUNT C100 BY ENTRY WHERE *where-clause* is acceptable even though C100 is not at level 0.

If the RELOAD command contains a where-clause and a LIMIT command is in effect, SYSTEM 2000 software reloads only if the number of level 0 records is within the specified limit. See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for a discussion of the LIMIT command.

The cycle number for the database is reset to zero at the beginning of the reload process and is incremented by one when the reloading completes. The only exception is if a where-clause produces no qualified logical entries; then the cycle remains reset to zero, because the database contains no entries.

Abbreviations:
ORDERED BY = OB
WHERE = WH

If a RELOAD command does result in the message saying that no records were selected, the resulting database consists of a database definition but no entries. If a saved copy of the database does not exist, the database has to be recreated from scratch by rerunning the jobs that created it. See Chapter 5, "Saving and Restoring Databases," for information about a saved database.

If update logging has been activated for the database, the RELOAD command suspends it. Update logging remains suspended until a SAVE or a RESTORE command activates it for the reloaded database. See Chapter 5, "Saving and Restoring Databases," for information about update logging.

You can issue the RELOAD command in either the CONTROL processor or the QUEST processor. If a batch fatal error occurs, the batch run ends with a batch fatal error message. If a RELOAD command is rejected in interactive mode, the job remains in the processor in which the command was issued, and SYSTEM 2000 software issues an error message. A successful RELOAD command results in either the database being reloaded or a message informing you that no records were selected by the specified where-clause. In either case, SYSTEM 2000 software provides the CONTROL processor for subsequent commands.

In a Multi-User environment, the database must be under exclusive use (that is, opened with the EXCLUSIVE DATA BASE NAME IS command) for a RELOAD command. If the database is not under exclusive use, SYSTEM 2000 software issues a batch fatal message. However, in interactive mode, the job remains in the processor in which the command was issued.

The RELOAD command is also discussed in *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition.*

Considerations

- If the RECORD format option is in effect, only level 0 data records are reloaded by a RELOAD command. See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for a discussion of the RECORD format option.

- If you modify a database schema such that restructuring occurs, issuing the reload command is superfluous, because the reload is done automatically.

- You can issue a RELOAD command for a damaged database, but the results may not be desirable. You can exclude the records with the errors, but you may not know which records are in error and you may not want to lose the data. To protect your database, see your SYSTEM 2000 Software Representative.

  If you get SYSTEM 2000 Error Code 247 or System 2000 Error Code 812, do not reload the database. These System 2000 Error Codes are issued if the errors are in the Hierarchical Table or the Data Table. They are complex errors and require seeing your SYSTEM 2000 Software Representative.

  If you do reload the database and the reload completes successfully, the resulting database is not flagged as damaged. (Errors in the Index Tables are always corrected by a reload.)

- If many logical entries (that is, more than half) need to be removed from a database, it may be more efficient to reload the entries to be kept than to issue the REMOVE TREE command. For example, if the past five years of accounting data are to be removed, it may be much faster to issue RELOAD WHERE *current year*.

  Also, see the method for massive REMOVE TREE operations shown in the REMOVE INDEX examples in **Changing an Item from Key to Non-Key: REMOVE INDEX** on page 4-4.

- A DBA password holder who has authorization to issue a RELOAD command still cannot issue a RELOAD command with a where-clause.

Examples

```
RELOAD:

RELOAD WHERE EMPLOYEE NUMBER LT 5075:

RELOAD ORDERED BY C1 WHERE C1 EXISTS:
```

Chapter 4

Creating and Removing Indexes

INTRODUCTION 4-1

CHANGING AN ITEM FROM NON-KEY TO KEY: CREATE INDEX 4-2

CHANGING AN ITEM FROM KEY TO NON-KEY: REMOVE INDEX 4-4

MASSIVE REMOVE TREE OPERATIONS 4-6

INTRODUCTION

When you first define a database, any item can be defined as key or non-key. An item is key by default, and the values for each key item are indexed for quick access. During the life of a database, you may change any item from key to non-key or non-key to key.

When planning a database, decide which items should be key items. Weighting a database with key items increases the size of the Index and often encumbers updating. The availability of the CONTROL language INDEX commands and the capability of specifying non-key items in the where-clause allows the initial definition to favor non-key items, because building the indexes for specific items with the CREATE INDEX command is relatively inexpensive.

There are two ways to change the key status of an item:

- If you want to change the key status of items and you do not want to restructure the database, issue the CREATE INDEX or REMOVE INDEX command. The INDEX commands do not restructure the database; they affect only the specified items by creating or removing indexes for those items.

- If you want to change the key status of items and you also need to restructure the database in a DEFINE session, issue the DEFINE language CHANGE command to change the key status of items.

The CREATE INDEX and REMOVE INDEX commands provide significant time savings when processing massive REMOVE TREE or LOAD operations. A method for accomplishing this is provided in the REMOVE INDEX examples in **Changing an Item from Key to Non-Key: REMOVE INDEX** on page 4-4.

CHANGING AN ITEM FROM NON-KEY TO KEY: CREATE INDEX

The CREATE INDEX command changes a non-key item to a key item and creates an index of the item's values without restructuring the database.

Only the master password holder can issue the CREATE INDEX command.

Format

```
[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.

When you issue the CREATE INDEX command, the non-key/key status for the specified item(s) is changed in the Definition Table. SYSTEM 2000 software passes through the Data Table once to gather the non-key item values for items specified in the CREATE INDEX command and then creates appropriate entries in the Index.

In a Multi-User environment, the CREATE INDEX command does not require exclusive use. While processing the command, SYSTEM 2000 software places a global hold on the database to prevent access.

Each time a CREATE INDEX command is processed, the definition version number is incremented by one. If the command contains errors, no processing is done and the definition version number is not incremented. The CREATE INDEX command does not increment the database cycle number, because the data are not changed.

The status of update logging is not changed during processing of the CREATE INDEX command. The database is flagged as damaged before the index is actually created. When the command completes successfully, the damage flag is cleared. However, you should save a new copy of the database, and update logging should be reactivated each time the definition number changes.

Considerations

- A CREATE INDEX command cannot be processed for a damaged database.
- If there is an error in any of the item names or numbers given in a CREATE INDEX command, the entire command is rejected unprocessed.
- If you specify an item name or number more than once in a CREATE INDEX command, a batch fatal error results. In interactive mode, you remain in the CONTROL processor. In any case, SYSTEM 2000 software issues an error message and rejects the entire command.

- As a time-saver, if more than one item is to be converted, specify the items in one consolidated list with a single CREATE INDEX command. Each CREATE INDEX command makes one pass of the Data Table to gather pertinent values.

  See also the timesaving method for massive REMOVE TREE operations using the CREATE INDEX and REMOVE INDEX commands in Example 3 in **Changing an Item from Key to Non-Key: REMOVE INDEX** on page 4-4.

- The Rollback Log (if enabled) is used during CREATE INDEX processing to record before-images. See Chapter 6, "Recovering Databases with Coordinated Recovery," for information about the Rollback Log.

Examples

```
INDEX C106, PROFICIENCY, MAJOR FIELD:
```

CHANGING AN ITEM FROM KEY TO NON-KEY: REMOVE INDEX

The REMOVE INDEX command changes an item from key to non-key and removes the index(es) of the values for the specified item(s) without causing a restructure of the database.

Only the master password holder can issue the REMOVE INDEX command.

Format

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

LOSE specifies that removed Multiple Occurrence Table space is lost until restructuring, reorganizing, or reloading occurs. The default is LOSE.

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.

When you issue the REMOVE INDEX command, the key or non-key status for the specified item(s) is changed in the Definition Table, and the appropriate index(es) are removed. Removed entries in the Distinct Values Table are chained to the appropriate reusable space lists for use at a later time. If a value exceeds the picture for the item, each of its appearances in the Data Table points to a new entry in the Extended Field Table; each occurrence of overflow from non-key values has an entry in the Extended Field Table.

The LOSE and REUSE options pertain to the Multiple Occurrence Table entries being removed. LOSE (the default) specifies that the removed Multiple Occurrence Table space is lost until restructuring, reorganizing, or reloading occurs. REUSE specifies that the removed Multiple Occurrence Table space is placed on the reusable space lists.

The LOSE option takes less processing time than the REUSE option, because removed Multiple Occurrence Table entries need not be cleared or chained to reusable space lists. For the LOSE option, the software chains through the Multiple Occurrence Table blocks only if a value exceeds the item's picture.

In a Multi-User environment, the REMOVE INDEX command does not require exclusive use. While processing the command, a global hold is placed on the database to prevent access.

Each time a REMOVE INDEX command is processed, the software increments the definition number by one. If the command contains errors, no processing is done and the definition version number is not incremented. The REMOVE INDEX command does not increment the database cycle number, because the data are not changed.

Abbreviation: REMOVE = RE

The Update Log status is not changed during processing of the REMOVE INDEX command. The database is flagged as damaged before the index is actually removed. When the command completes successfully, the damage flag is cleared. However, you should save a new copy of the database, and update logging should be reactivated each time the definition number changes. If the database is not properly saved after removing the index, the database tables will not reflect the removal. See Chapter 5, "Saving and Restoring Databases," for information about update logging.

Considerations

- You cannot issue a REMOVE INDEX command for a damaged database.
- If there is any error in any of the item names or numbers given in a REMOVE INDEX command, the entire command is rejected unprocessed.
- Specifying an item name or number more than once in a REMOVE INDEX command is a batch fatal error. In interactive mode, you are given the CONTROL processor to continue entering commands. In any case, SYSTEM 2000 software issues an error message and rejects the entire command.
- Grouping several items in one REMOVE INDEX command does not decrease processing time, because each item is treated separately.
- The Rollback Log (if enabled) records before-images during the REMOVE INDEX processing. See Chapter 6, "Recovering Databases with Coordinated Recovery," for information about the Rollback Log.
- If you specify the LOSE option, reorganize the Multiple Occurrence Table to regain lost space. (See the REORGANIZE command, **Reorganizing Databases: REORGANIZE** on page 3-1.) However, if the item(s) being changed to non-key represents a small percent of the data in the database, the REUSE option is cheaper than using the LOSE option followed by REORGANIZE.
- If massive REMOVE TREE or LOAD operations are necessary for the database, the CREATE INDEX and REMOVE INDEX commands can be significant time-savers.

Examples

```
REMOVE INDEX C413:

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

MASSIVE REMOVE TREE OPERATIONS

Notice the following sequence of commands for large volume REMOVE TREE operations:

```
.
.
.

CONTROL:
REMOVE INDEX/LOSE/ key items in the schema records to be removed:
QUEST:
REMOVE TREE SR WHERE. . .
CONTROL:
CREATE INDEX key items in the specified schema records:
REORGANIZE ALL:

.
.
.
```

In this example, the REMOVE INDEX command deletes the key item indexes in the data trees to be removed. There are only a few I/Os to the Index and no I/Os to the Hierarchical Table or the Data Table except to handle key value overflow, if any. As a result, the REMOVE TREE command(s) accesses only the Hierarchical Table to accomplish the removals, because no key items exist in the data trees being removed.

The CREATE INDEX command then changes the Definition Table to reflect that the items are now changed back to key from non-key and makes a pass through the Hierarchical Table to see if any records exist for the items mentioned. If they were removed, no other action is taken. If some still remain, the indexes are created for the specified items. The process in this example involves fewer I/Os than a restructure resulting from the use of the DEFINE processor to accomplish the same task. The REORGANIZE ALL (or INDEX) command reclaims in bulk the Index space freed by the REMOVE INDEX command.

In one case, the technique in this example provided a 3-to-1 reduction in CPU time and 15% reduction in I/Os. Also, the I/Os are done primarily on the database files, not on the scratch/sort files. The scratch/sort file requirements can be reduced by a factor of approximately 10. The amount of optimization achieved depends on the percentage of schema records removed versus those still existing in the database. If records for a schema record (or several) are removed, the reduction in I/O and CPU time is very significant. However, this process can be expensive if there are many key overflow values.

Also, see how to avoid massive REMOVE TREE operations in the RELOAD examples in **Reconstructing Databases: RELOAD** on page 3-4.

Chapter 5

Saving and Restoring Databases

WHY SAVE YOUR DATABASE? 5-1
Saving a Database Without Activating Update Logging 5-2
Saving a Database and Activating Update Logging 5-3

SAVING A DATABASE: SAVE 5-4
Dynamically Allocating the Savefile for SAVE 5-5
Examples: SAVE commands for dynamic Savefile allocation 5-6
Specifying the Savefile, Keepfile, or Update Log with JCL 5-7

RELEASING A DATABASE: RELEASE 5-11

RESTORING A DATABASE: RESTORE 5-12
Dynamically Allocating the Savefile for RESTORE 5-15
Specifying JCL for the RESTORE Command 5-15

TRANSFERRING UPDATES: KEEP 5-18
Dynamically Allocating the Keepfile 5-18
Specifying the Keepfile with JCL 5-19

APPLYING LOGGED UPDATES: APPLY 5-22
Format 5-22
Specifying JCL for the APPLY Command 5-23

SUSPENDING UPDATE LOGGING: SUSPEND 5-26
Other Operations That Suspend Update Logging 5-26

EXAMPLES OF THE SAVE/RESTORE PROCESS 5-28

WHY SAVE YOUR DATABASE?

There are two main reasons to save a database:

- You should save a database in case it becomes damaged and you need to restore it. You should always save a database after massive updates, definition modifications, or incremental loads.

  A database can become damaged if the disk cannot be read because of a head crash or a power failure. A database can also become damaged if an update is interrupted or if erroneous updates are made to the database. If the update operation is interrupted and does not complete, the database is left in the damaged state, and no further updates are allowed.

- You should save a database if the database is needed only weekly or monthly. Rather than maintain the active database files, you can save the database, release the active files, and then restore the database when needed.

SYSTEM 2000 software offers various methods to ensure the integrity of your database:

- You can save a database every time you update it. If the active database becomes damaged, you can restore the saved copy. However, any updates made since the last save operation will not be restored.

- You can save a base copy of a database and activate update logging, which records update transactions. You can restore the base copy and then apply the update transactions. This ensures that the database is totally restored.

- You can also enable Coordinated Recovery for automatic recovery of databases in a Multi-User environment.

Coordinated Recovery is explained in Chapter 6, "Recovering Databases with Coordinated Recovery." Saving a database and update logging are discussed in this chapter.

Saving a Database Without Activating Update Logging

You can save a database as often as needed. For example, you can save a database each night or after any period of processing. However, if the database is relatively large, the save process could take a long time and consume resources. Also, any update transactions since the last time you saved the database are not recorded unless you specifically activated update logging.

The SAVE command copies a database to a Savefile, a QSAM file which can be tape or disk. The Savefile holds the entire saved database, and you can use the saved copy to restore the database, if needed. The active database files remain accessible on disk.

You can save a database without activating update logging with any of the following methods:

- Issue a 'SAVE:' command with no options and no DD statement in the JCL. SYSTEM 2000 software will dynamically allocate the Savefile using the defaults.

- Dynamically allocate the Savefile in the SCF session using one or more SAVE command options and no DD statement.

- Include a DD statement for the Savefile in the JCL and issue a 'SAVE:' command.

If the database becomes damaged later, follow these steps to restore the saved copy:

1. Issue the RELEASE command for the damaged database. The damaged database disk files are made unusable.

2. Issue the RESTORE command to copy the database from the Savefile to the active database disk files.

Saving a Database and Activating Update Logging

Rather than saving your database every time you update it, you can save a **base** copy of a database and activate update logging.

Update logging involves two files: the Update Log and the Keepfile. The Update Log is database File 7 (optional), a temporary disk file where SYSTEM 2000 software writes update transactions performed since the database was saved. As the Update Log fills, users issue KEEP commands to transfer the contents of the Update Log to the Keepfile. The Keepfile is a permanent file that can be used to apply the updates if the database becomes damaged. Together, the saved base copy of the database and the Keepfile ensure that a damaged database can be restored to the point at which it was damaged.

The SAVE command and the RESTORE command have syntax (the slash) to activate update logging for a database. When the cycle number for the database is incremented, any updates processed are recorded.

To save a database and activate update logging with the SAVE command:

1. Explicitly or dynamically allocate the Savefile and the Update Log. (The Keepfile is allocated when you issue the KEEP or APPLY command.)

2. Issue the SAVE command including the slash to activate update logging for the database to be saved.

SYSTEM 2000 software copies the database to the Savefile; the active database remains in the active database disk files, and update logging is activated for the database. (Note that the Update Log is machine-readable only by SYSTEM 2000 software. The Update Log is an internal SYSTEM 2000 audit trail, and it cannot be meaningfully displayed.)

Recovery with the Keepfile involves the following steps:

1. Issue the KEEP command to move the recorded transactions from the Update Log to the Keepfile.

2. If the database becomes damaged, issue a final KEEP, then issue the RELEASE command for the damaged database. The active (damaged) database disk files are made unavailable.

3. Issue the RESTORE command to copy the database from the Savefile to the active database disk files.

4. Issue the APPLY command to apply the updates from the Keepfile to the restored copy of the database.

The database is now restored to the point at which it was damaged.

See specific information on each command in the following sections.

SAVING A DATABASE: SAVE

The SAVE command copies a specified database from active disk files onto a Savefile, which can be tape or disk. This copy becomes a saved database. The Savefile is a QSAM file; the Keepfile is a BDAM file. SAVE can also activate or suspend update logging for a database with the SAVE command.

Only the master password holder or a DBA password holder with authorization for this command can issue the SAVE command.

Format

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 data set 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.

When you issue a SAVE command, SYSTEM 2000 software copies the disk files for the database onto a Savefile. The Savefile is a tape or disk QSAM file that holds the saved database. On the Savefile, the individual files of the database constitute one file. (An active database consists of six files.) Before the save process starts, any database pages in the buffer that have been updated are "cleared," that is, written to the active database disk files.

SAVE copies the database files to the specified volumes, if any. It also requests additional volumes after filling those specified.

SAVE suspends update logging unless you specifically activate it with the slash, for example,

```
SAVE/:
```

If update logging was activated by a previous SAVE or RESTORE command and there have been updates since the previous KEEP command, you should issue a KEEP command before you issue a SAVE command. Otherwise, these updates will not be transferred to the Keepfile, because the SAVE command does not save the Update Log disk file.

In a Multi-User environment, the SAVE command does not require exclusive use. SYSTEM 2000 software automatically puts a temporary hold on the database when saving the Master Record and then places a "retrieval only" mode on the database for the rest of the save operation. Therefore, you do not need to wait for exclusive use to save a database, and retrievals can continue while the database is being saved. (SYSTEM 2000 software protects the database from updates during the save processing.)

The DDBUFN execution parameter enables you to specify the number of QSAM buffers that will be used for database files during SAVE and RESTORE processing. For more details, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*. To obtain more than five buffers for the Savefile, you could also specify DCB=BUFNO=*n* in the JCL for *xxxxxxx*S.

Dynamically Allocating the Savefile for SAVE

SYSTEM 2000 software dynamically allocates the Savefile if it is not already allocated in the JCL or CLIST. The Keepfile is dynamically allocated when you issue a KEEP or APPLY command. If the software dynamically allocates the Savefile or Keepfile, it dynamically deallocates either one after SAVE, RESTORE, KEEP, or APPLY processing completes.

Note: If the data set is already cataloged, the data set information given in the SAVE command (for example, volume, UNIT, or DS) is ignored.

SYSTEM 2000 software performs the following steps before opening the Savefile:

1. Sees whether the DDname is specified in the JCL or CLIST, and, if so, uses that file.
2. Gives user exit EXIT09 a chance to dynamically allocate the file.
3. Looks for a cataloged file, using the default data set name or the DSN specified in the SAVE command.
4. Allocates a new Savefile on disk or tape as specified in the SAVE command. Disk is the default.

The default data set name for the Savefile is:

```
DSN=prefix.databaseS
```

If you activate update logging, SYSTEM 2000 software dynamically allocates database File 7 if it is not already allocated in the JCL or CLIST or with the ALLOC command. For information about dynamically allocating the Keepfile, see **Dynamically Allocating the Keepfile** on page 5-18.

For a SAVE command, if the software does not find an existing Savefile data set, it creates the Savefile data set dynamically using the following attributes:

```
DSN=prefix.databaseS          (or the DSN specified in the SAVE command)
VOL=SER=volume-list           (or the default for SYSDA)
UNIT=SYSDA
RECFM=VB
LRECL= largest block size + 4
BLKSIZE= depends on type of disk
                              (The 3380 default is 23472.)
SPACE= primary space large enough to save the database
                              (SYSTEM 2000 software knows
                               the total pages in use.)
```

If you specify TAPE in the SAVE command, the software uses the following attributes:

```
DSN=prefix.databaseS          (or the DSN specified in the SAVE command)
VOL=SER=volume-list           (or the default for TAPE)
UNIT=TAPE
RECFM=VB
LRECL=32756
BLKSIZE=32760
```

Note: The UNIT=TAPE option dynamically allocates a tape drive and saves the database on tape. DSN=*dsn* allows you to specify an alternate disk data set name, or, for TAPE, an alternate tape data set name.

Examples: SAVE commands for dynamic Savefile allocation

```
SAVE:
SAVE/:                        (activates update logging)
SAVE DATA BASE:
SAVE DATA BASE/:
SAVE ON DSN='NEW.DISK.SAVE':
SAVE ON volume-list UNIT=TAPE DSN= . . . / UNIT=TAPE volume:
```

Specifying the Savefile, Keepfile, or Update Log with JCL

You can allocate the Savefile or Keepfile, or you can let SYSTEM 2000 software dynamically allocate them.

If you allocate the Savefile or Keepfile with JCL or a CLIST, you can code all the information on the DDname *xxxxxxx*S and simply issue the 'SAVE:' command with no additional syntax. Any information specified in the SAVE command syntax overrides that information in the DD statement. The only exception is saving to an existing cataloged data set. Information from the catalog takes precedence over everything else.

JCL for the Savefile:

All of these three examples are valid:

```
//xxxxxxxS  DD  DSN= . . .,DISP=OLD
                        or
              DSN= . . .,DISP=(NEW,CATLG),
              UNIT=SYSDA,VOL=SER=STOR01,SPACE=(CYL,(10,1))
                        or
              DSN= . . .,DISP=(NEW,CATLG),
              UNIT=TAPE,VOL=SER=TAPE01
```

JCL for the Keepfile: JCL for the Keepfile is the same as for the Savefile except the suffix in the DDname is K.

JCL for the Update Log:

```
//xxxxxxx7 DD DISP=OLD,DSN=dsn
```

Note: The Update Log is the seventh database file. Therefore, *xxxxxxx* is the first seven characters of the database name with Xs replacing blanks. The example above assumes that File 7 has been preallocated. DCB=BLKSIZE=*block size* can be coded the same as database Files 1-6 for a new allocation, as shown in this example:

```
//xxxxxxx7 DD DSN=dsn,DISP=(NEW,CATLG),
//         SPACE=space,VOL=SER=volser, UNIT=unit
```

The following notes give specific information about the format of the Savefile:

- DCB=(RECFM=VB,BLKSIZE=*x*)

  where *x* equals 32760 for tape and *x* is as stated below for disk.

  ```
  27980  for  3390
  23472  for  3380
  17592  for  3375
  19069  for  3350
   8368  for  3340
  13030  for  3330
  ```

- The file created by the SAVE command is a QSAM file.

- The first physical record on the Savefile is 80 characters long. The first six full words contain the length (in hex) of the page size for each database file. That is, the first full word in this record is the page size (in hex) of database File 1, the second full word is for database File 2, and so forth.

See the following notes on using the DSN and VOL=SER parameters.

Case 1: SAVE: or SAVE/:

- If you have JCL, specify all information in the JCL to identify the new or existing Savefile.
- With no JCL, let SYSTEM 2000 software dynamically allocate the new or existing Savefile, using all the defaults.
- Since the data set name for the Savefile is derived from the database name (plus a suffix of S), several users can save different databases at the same time in a Multi-User environment.
- Multi-User does not require exclusive use of the database for a SAVE command, but a global hold status is automatically obtained, which locks out other users attempting a hold or update of the database. Retrievals are allowed after File 1 has been saved.

Case 2: Specifying the Data Set Name

- If you supply the data set name in the command and also in the JCL, SYSTEM 2000 software ignores the JCL.
- If VOL=SER is not coded in the JCL and the data set name is not cataloged, a scratch tape is used if UNIT=TAPE; for disk, the file is allocated to UNIT=SYSDA. The data set name is cataloged when the SAVE command completes. You can also specify a volume in the SAVE command.
- If VOL=SER is coded in the JCL and the data set name is not cataloged, the VOL=SER in the JCL is used.
- If the data set name is cataloged, the VOL=SER from the catalog is used, even if it is specified in the JCL.
- A global hold is obtained automatically on the database, as described under Case 1.

Note: With no JCL for the Savefile, SYSTEM 2000 software dynamically allocates and uses an existing cataloged file or creates a new disk file on a volume designated with UNIT=SYSDA.

Case 3: Generation Data Groups

Generation Data Groups (GDG) are supported in the JCL and in the SAVE command. Once you have established a GDG, you can specify the relative data set name with either method.

The following example specifies the data set name in the SAVE command syntax:

```
SAVE ON DSN='S2K.EMPLOYES(+1)':
```

The following statement specifies the appropriate data set name in the JCL:

```
//EMPLOYES DD  DSN=S2K.EMPLOYES(+1),DISP=OLD
```

Considerations

- You cannot issue a SAVE command during a QUEUE/TERMINATE session.
- The DDname for the Savefile must be the first seven characters of the database name with Xs replacing blanks, followed by a suffix of S. (TAPES2K is no longer honored as the DDname.) In the SAVE command, you must give the volume serial number that you specified when you allocated the Savefile.
- You can issue a SAVE command for a damaged database, but the save process does not affect the damaged condition.
- If the Rollback Log is enabled, the SAVE command creates a user synchpoint for Coordinated Recovery.
- If a SAVE command specifies more than ten volumes, SYSTEM 2000 software issues an error message. However, for saving a database that exceeds ten tape volumes, up to the maximum number of volumes allowed by the operating system, use the following procedure:

  Specify the volumes in the Savefile DD statement.

  Execute a simple SAVE command with no options.

  For example,

  ```
  //xxxxxxxS DD DSN=dsn,UNIT=(TAPE,,DEFER),
  // VOL=SER=(111111,222222,333333,444444,555555,666666, . . .),
  // DISP=NEW
            .
            .
            .
  SAVE:
  ```

- The DDBUFN execution parameter allows you to set the number of QSAM buffers to be used for database files during SAVE processing. The default is five buffers. For more information about setting execution parameters, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*. To obtain more than five buffers for the Savefile, you can specify DCB=BUFNO=*n* in the JCL for *xxxxxxx*S.

- You can suspend update logging by issuing a SAVE command that does not activate update logging, by reloading the database with the RELOAD command, by issuing the SUSPEND command, by restructuring the database in a DEFINE session, or by doing a load in a PLEX program.

Examples

Example 1

This example saves the current database (EMPLOYEE) and does not activate update logging:

```
USER, DEMO:              (master password)
DBN IS EMPLOYEE:
     .
     .
     .
(updates)
     .
     .
     .
CONTROL:
SAVE:                    (saves EMPLOYEE database)
     .
     .
     .
EXIT:
```

Example 2

This example saves the BOOKS database and activates update logging:

```
USER, ABC:               (master password)
DBN IS BOOKS:
     .
     .
     .
(incremental load)
     .
     .
     .
CONTROL:
SAVE/:
     .
     .
     .
EXIT:
```

RELEASING A DATABASE: RELEASE

The RELEASE command makes the active database disk files unavailable. You might want to release a database because it has become damaged and needs to be restored, because you are not going to use the database for a long period of time and wish to make the disk space available, or for other reasons.

Only the master password holder or the DBA password holder with authorization for this command can issue the RELEASE command. You must have exclusive use of the database to issue the RELEASE command.

Format

```
RELEASE :
```

Before you issue the RELEASE command, be sure the database has been saved. If no saved copy of the database exists, you will have to recreate the database from scratch. See **Saving a Database: SAVE** on page 5-4.

You must issue the RELEASE command before you issue the RESTORE command. The RELEASE command reformats database File 1, which renders the database unusable. That is, Files 2 through 6 remain allocated until you specifically free them, but they can no longer be accessed.

The RELEASE command suspends update logging. If update logging was activated when the last SAVE command was issued, it is reactivated when you issue the RESTORE command. Or, if update logging was not activated with the last SAVE command, you can activate it with the RESTORE command.

Consideration

Execution option OPT009 forces SYSTEM 2000 software to clear all the database files of a released database to binary zeros. Under normal operation, only File 1 is cleared to binary zeros. (See the *SYSTEM 2000 Product Support Manual, Version 12, First Edition* for more information on SYSTEM 2000 execution options.)

Example

```
USER, DEMO:
EXCLUSIVE DATA BASE NAME IS EMPLOYEE:
     .
     .
     .
CONTROL:
RELEASE:
EXIT:
```

RESTORING A DATABASE: RESTORE

The RESTORE command copies a specified database from a Savefile onto the active database disk files. You can also activate update logging for the specified database or change its name using RESTORE. However, you cannot suspend update logging with RESTORE.

Only the master password holder or a DBA password holder with authorization for this command can issue RESTORE.

Format

```
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.

You can restore a backup copy (Savefile) of a database to look at historical data. Or you can use a restored copy if the current database becomes damaged. Also, if you have activated update logging, the restored database becomes the base version for applying updates (from the Keepfile) in the recovery process.

Before you issue the RESTORE command:

- You need to have saved the database. See **Saving a Database: SAVE** on page 5-4.
- You probably need to have issued the RELEASE command, which marks the existing database files unusable. The exception is this situation:

  If the disk space to be used for the restore operation is newly allocated, that is, contains no database (definition or data), a RELEASE command is not necessary. In fact, RELEASE would result in an error message.

You can activate update logging with the RESTORE command. This may not be necessary; if update logging was activated in a previous SAVE command, it remains active.

Additionally, you can change the name of a database with the RESTORE command. However, if you specify a new name for the database, update logging cannot be activated. This is because there would be no saved copy of the renamed database. Once the newly named database is saved, update logging can be activated. The old saved copy of the database is not altered by changing the database name.

Changing the database name suspends update logging for the new restored database. As the database is copied to disk, SYSTEM 2000 software changes the identification of the disk database to the new database name. The saved database copy still retains the old database name. Therefore, you must issue a SAVE command to create a Savefile that contains the new name. The new saved copy provides a beginning or recovery point (known cycle and date) for activating update logging for the renamed database.

When changing a database name, the = character cannot be part of the database name, because this character is part of the command syntax.

The following rules pertain to changing a database name with the RESTORE command:

Update Logging Status for *old-database-name*	Update Logging Specified in RESTORE *old=new*	Software's Action
activated	yes	rejects command with a batch fatal error message
not activated	yes	rejects command with a batch fatal error message
activated	no	accepts the name change but suspends update logging
not activated	no	accepts the name change

When you issue RESTORE, the database files are copied from the Savefile. A data set name can be supplied in the syntax DSN = *data-set-name*.

If DSN is specified for the Keepfile, SYSTEM 2000 software stores the specified data set name or other information for later use. When a subsequent KEEP command is issued for that Keepfile, SYSTEM 2000 software uses that information or the default Keepfile name.

SYSTEM 2000 software provides the QUEST processor after RESTORE processing completes.

Dynamically Allocating the Savefile for RESTORE

SYSTEM 2000 software dynamically allocates the Savefile if it is not already allocated in the JCL or CLIST. The Keepfile is dynamically allocated when you issue a KEEP or APPLY command. If the software dynamically allocates the Savefile or Keepfile, it dynamically deallocates it after the SAVE, RESTORE, KEEP, or APPLY command completes.

Note: If the data set is already cataloged, the data set information given in the RESTORE command (for example, volume, UNIT, or DS) is ignored.

SYSTEM 2000 software performs the following steps before opening the Savefile:

1. Sees whether the DDname is specified in the JCL or CLIST, and, if so, uses that file.
2. Gives user exit EXIT09 a chance to dynamically allocate the file.
3. Allocates the file using the default data set name or the DSN specified in the RESTORE command. The data set must exist and must be cataloged so that SYSTEM 2000 software can allocate the Savefile.

If you activate update logging, SYSTEM 2000 software dynamically allocates database File 7 if it is not already allocated in the JCL or CLIST or with the ALLOC command.

Specifying JCL for the RESTORE Command

The rules for JCL and SYSTEM 2000 command syntax for restoring a database are the same as those for saving a database (discussed in the SAVE command). The only difference is that Multi-User automatically gives you exclusive use of the database when you issue the RESTORE command. Exclusive use is relinquished only when you exit or respecify opening of the database without exclusive use.

Here is the general syntax for specifying the Savefile in the JCL:

```
//xxxxxxxS  DD  DSN= . . .,DISP=OLD
```

Considerations

- You cannot issue a RESTORE command in a QUEUE/TERMINATE session.
- The DDname for the Savefile is the first seven characters of the database name plus a suffix of S.
- The DDBUFN execution parameter allows you to set the number of QSAM buffers to be used for database files during SAVE processing. The default is five buffers. For more information about setting execution parameters, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*. To obtain more than five buffers for the Savefile, you can specify DCB=BUFNO=*n* in the JCL for *xxxxxxx*S.

- There is one case in which a database cannot be restored from a saved copy. If any of the disk database files are split across more than one disk volume and the corresponding DD statement specifies a disposition of NEW, then the command sequence 'RELEASE: RESTORE . . .' causes an abend with a B37 MVS completion code. This problem can be resolved by preallocating the database files with IEFBR14 and specifying a disposition of OLD in DD statements that refer to the database disk files.

Examples

Example 1

This example restores the EMPLOYEE database from the EMPLOYES Savefile.

```
RESTORE EMPLOYEE:
```

Example 2

This example restores the BOOKS database from a Savefile on SAS310.

```
RESTORE BOOKS FROM SAS310:
```

Example 3

This command yields an informative message. The database name is changed to B, and update logging is suspended regardless of its previous status.

```
RESTORE A = B:
```

Example 4

This sequence of commands releases existing disk files and restores the Savefile for the EMPLOYEE database.

```
USER, ABLE:
DBN IS EMPLOYEE:
(A batch-fatal error occurs if the database is not on disk.)
  CONTROL:
  RELEASE:
  RESTORE EMPLOYEE:
```

Example 5

Generation Data Groups (GDG) are supported in the JCL and in the SAVE command. Once you have established a GDG, you can specify the appropriate data set name with either method.

The following example specifies the data set name in the RESTORE command syntax:

```
RESTORE EMPLOYEE FROM DSN='S2K.EMPLOYES(+0)':
```

The following statement specifies the appropriate data set name in the JCL:

```
//EMPLOYES DD  DSN=S2K.EMPLOYES(+0),DISP=OLD
```

TRANSFERRING UPDATES: KEEP

The KEEP command transfers processed updates that are recorded on the Update Log disk file to the Keepfile.

Format

```
KEEP :
```

If you issue KEEP when update logging was not enabled in a SAVE or a RESTORE command, or when SUSPEND was issued without a subsequent SAVE or RESTORE command to enable update logging, SYSTEM 2000 software issues an error message.

If you issue KEEP when the correct device for a Keepfile is not available, or if some other error occurs relating to that device, SYSTEM 2000 software issues an error message. If the Keepfile is a disk file, it is considered a scratch file and must be on a like device type with other disk scratch files.

SYSTEM 2000 software dynamically allocates the Keepfile if it is not already allocated. The default data set name is *prefix.database*K.

If DSN was specified for a Keepfile in the most recent SAVE or RESTORE command, a KEEP command uses that information. The volume specified in the command overrides the first VOL=SER in the JCL.

During the processing of a KEEP command, SYSTEM 2000 software automatically puts the database into hold status to prevent updating.

When a KEEP command completes processing, the software displays the database definition number, cycle, data, and time.

Dynamically Allocating the Keepfile

The DDname for the Keepfile is the first seven characters of the database name plus a suffix of K. (The DDname KEEPFILE used in earlier releases is no longer valid.) SYSTEM 2000 software takes the following steps when opening the Keepfile:

1. Sees whether the Keepfile DDname is given in the JCL or CLIST, and, if so, uses that allocation.

2. Gives user exit EXIT10 a chance to dynamically allocate the Keepfile.

3. For the first KEEP command, allocates a new file on disk or tape as specified in the SAVE or RESTORE command. Disk is the default.

4. Allocates an existing Keepfile with the default data set name or the DSN specified in the SAVE or RESTORE command. If the command is APPLY or a subsequent KEEP (after the first KEEP), the file must exist and must be cataloged in order for SYSTEM 2000 software to allocate the Keepfile.

For example, if you issue the following commands, the default data set name is *prefix.database*K:

```
SAVE /:  KEEP:
```

If the KEEP command is the first KEEP and an existing data set is not found, SYSTEM 2000 software creates a Keepfile with the following attributes:

```
DSN=prefix.databaseK    (or the DSN in the SAVE or RESTORE command)
VOL=SER=volume          (or the default for SYSDA)
UNIT=SYSDA
RECFM=F
LRECL=same as database File 7
BLKSIZE=same as database File 7
SPACE=enough space to do the KEEP in primary space
  and secondary space of the same size to allow subsequent KEEPs.
```

If you specify TAPE for the Keepfile, the software uses the following attributes:

```
DSN=prefix.databaseK    (or the DSN in the SAVE or RESTORE command)
VOL=SER=volume          (or the default for TAPE)
UNIT=TAPE
RECFM=F
LRECL=same as database File 7
BLKSIZE=same as database File 7
```

Note: The UNIT=TAPE option dynamically allocates a tape drive and saves the database on tape. DSN=*dsn* allows you to specify an alternate disk data set name, or, for TAPE, an alternate tape data set name.

Specifying the Keepfile with JCL

You can allocate the Keepfile, or you can let SYSTEM 2000 software allocate it dynamically.

If you allocate the Keepfile with JCL or a CLIST, you can code all the information on the DDname *xxxxxxx*K and simply issue the 'KEEP:' command. Any information specified in the SAVE command syntax overrides that information in the DD statement. The only exception is keeping to an existing cataloged data set. Information from the catalog takes precedence over everything else. See **Specifying the Savefile, Keepfile, or Update Log with JCL** on page 5-7 for sample JCL.

Considerations

- You cannot issue a KEEP command in a QUEUE/TERMINATE session.

- If the Update Log is empty when a KEEP command is issued, SYSTEM 2000 software issues an error message. If the Keepfile is also empty and is a tape, the tape is mounted and a label and logical end-of-file mark are written on it. The tape is then demounted.

- Changes due to DEFINE language commands, the CREATE/REMOVE INDEX commands, or the LOAD command in PLEX are not recorded on the Update Log. If a PLEX LOAD command is successful, update logging is suspended. DEFINE language command changes and CREATE/REMOVE INDEX command changes make the Update Log incompatible for later APPLY commands. Such changes should be preserved by using the SAVE command. The LOAD process uses the Update Log.

- If the QUEST language CLEAR UPDATE LOG command (system default) is in effect, the updates are not written to the Update Log until the memory buffer is full. If the CLEAR UPDATE LOG AUTOMATICALLY command is issued, every update is written to the Update Log as soon as it has been processed. These commands cause clearing of the Update Log independent of database files.

- The Keepfile can be tape or disk, limited to one volume. However, if you issue a KEEP command for a series of updates and the end of the Keepfile is reached, another tape is automatically requested to complete the KEEP operation. If KEEP causes the use of an additional volume, no more KEEP commands can be issued. You must immediately save the database and reactivate update logging. The two-volume Keepfile can be used in a RESTORE/APPLY operation.

- If you issue KEEP for a damaged database, the update cycle number is the cycle number for the last complete update that was kept on the Update Log. This cycle number is always equal to or greater than the cycle number prior to the last KEEP command. The KEEP command processing retains as many complete update cycles as possible regardless of the last cycle that was reflected in the Master Record.

- If the Rollback Log is enabled, the KEEP command causes a database synchpoint for Coordinated Recovery.

Example

Example 1

You must issue the KEEP command periodically after a sequence of update commands in order to transfer batches of update recordings from the Update Log disk file to the Keepfile.

```
USER, DEMO:
DBN IS EMPLOYEE:
      .
      .
      .
(update commands)          (written on the Update Log disk file)
      .
      .
      .
CONTROL:
KEEP:                      (transfers Update Log to Keepfile)
QUEST:
(update commands)
      .
      .
      .
CONTROL:
KEEP:
QUEST:
      .
      .
      .
```

APPLYING LOGGED UPDATES: APPLY

The APPLY command takes updates recorded on a Keepfile and applies them to a restored database.

Only the master password holder or a DBA password holder with authorization for this command can issue the APPLY command. You must have exclusive use of the database for APPLY operations.

Format

```
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.

You must issue the RESTORE command before you issue the first APPLY command. SYSTEM 2000 software dynamically allocates the Savefile and the Keepfile if they are not already allocated. For details, see **Dynamically Allocating the Keepfile** on page 5-18.

An APPLY command with the ALL option means that all update cycles in the Keepfile should be applied to the restored database. An APPLY command can also specify that only a range of consecutive update cycles be applied, starting with the first cycle in the Keepfile. For instance, if a Keepfile holds update cycles 20 through 35 for a database, an APPLY command can indicate that only cycles 20 through 29 are to be applied. Note, however, that update cycles cannot be skipped; for example, you cannot apply cycles 20 through 29 and then apply cycles 32 through 35.

After you issue APPLY, the database is current as of the last update cycle that was applied. Updated database pages in memory have been cleared to disk. Now you can do new updates to the database, and they are recorded in the Keepfile. If you do an update or issue KEEP following RESTORE, you cannot issue APPLY, because SYSTEM 2000 software checks the cycle number on the Keepfile to ensure consecutive updates.

Only one Keepfile can be associated with an APPLY command for one restored database; this Keepfile is the one that was associated in the most recent SAVE or RESTORE command. SYSTEM 2000 software marks the beginning of a Keepfile with information, including the name and cycle number of the associated Savefile. If the name and cycle number of the restored database do not match the Keepfile information when you issue an APPLY command, SYSTEM 2000 software issues an error message.

You can issue APPLY only if the number of the cycle in the Keepfile label matches the current cycle number for the restored database. Consequently, you cannot update a database between a RESTORE and an APPLY command. If you try to update the database after you issue a RESTORE command and then you issue APPLY, SYSTEM 2000 software issues an error message. An error message also appears if you specify a previously applied cycle number in an APPLY command.

Abbreviation: THROUGH=THRU

The highest possible database cycle number is 2,147,483,647. If another cycle of updating occurs after the maximum cycle number, the cycle number is reset to 1.

Specifying JCL for the APPLY Command

You can allocate the Keepfile, or you can let SYSTEM 2000 software allocate it dynamically.

If you allocate the Keepfile with JCL or a CLIST, you can code all the information on the DDname *xxxxxxx*K and simply issue the 'APPLY:' command. Any information specified in the RESTORE command syntax overrides that information in the DD statement. The only exception is applying from an existing cataloged data set. Information from the catalog takes precedence over everything else. See **Specifying the Savefile, Keepfile, or Update Log with JCL** on page 5-7 for sample JCL.

Considerations

- You cannot issue an APPLY command in a QUEUE/TERMINATE session.
- You cannot issue an APPLY command for a damaged database.
- You cannot issue an APPLY command after updates have been made to the database.
- If update logging is not activated or if the database is not open and you issue an APPLY command, SYSTEM 2000 software issues an error message.
- The APPLY command can successfully use a two-volume Keepfile if you issue a KEEP command and all the updates do not fit on the Keepfile. SYSTEM 2000 software requests another tape to complete the KEEP operation.
- Updates to a database are done in cycles. An update cycle represents the database after an update operation is done. Cycles are numbered consecutively. Depending upon the execution mode, the update cycle number is increased one or more times as a sequence of update commands is processed. For QUEST language updates and for PLEX immediate mode updates, the cycle number is incremented by one after each successful update command. For a QUEUE/TERMINATE session, a PLEX queue mode session, or a Self-Contained Facility or PLEX loading session, the cycle number is increased by one after you issue the TERMINATE command and the processing is completed.

  If you use the Self-Contained Facility with update logging, SYSTEM 2000 software issues the following message after an update is complete:

  ```
  UPDATE CYCLE = yyy
  ```

  where *yyy* is the database cycle number. This message does not appear for an APPLY operation, but the normal "UPDATED. . ." message appears after all cycles have been applied for one APPLY command.

The cycles of updates in the Keepfile correspond to the update operations done on the database after the database is saved. The cycles of updates in the Keepfile are ready to be applied, without any scan of the syntax or update values.

Examples

In these examples, SYSTEM 2000 software dynamically allocates the Savefile and the Keepfile if they are not already allocated.

Example 1

This example releases the current active database, restores the Savefile, and applies all recorded updates.

```
USER, DEMO:                    (master password)
DBN IS EMPLOYEE:
CONTROL:
RELEASE:                       (releases the active database)
RESTORE EMPLOYEE:              (restores from the saved copy)
CONTROL:
APPLY ALL:                     (applies all updates recorded
                                on the Keepfile)
QUEST:
   .
   .
   .
```

Example 2

This sequence of Self-Contained Facility commands illustrates restoring a database from an unrecoverable error using the APPLY command. The KEEP command writes all successful updates to the Keepfile before releasing the database.

```
CONTROL:
KEEP:
   .
   .                           (master password in effect)
   .
RELEASE:
RESTORE EMPLOYEE:
CONTROL:
APPLY ALL:
```

Example 3

These Self-Contained Facility commands retain update recordings through cycle 58, and then apply only the updates through cycle 55. This example assumes that cycles 56 through 58 contain erroneous updates:

```
   USER, DEMO:                      (master password)
   DBN IS EMPLOYEE:
        .
        .
        .
     (update commands)
        .
        .
        .
   CONTROL:                         (invokes the CONTROL processor)
   KEEP:                            (keeps through cycle 58)
   RELEASE:                         (releases the active database)
   RESTORE EMPLOYEE:                (restores from the saved copy)
   CONTROL:
   APPLY THRU CYCLE 55:             (applies through cycle 55)
        .
        .
        .
(resubmit corrected updates)
        .
        .
        .
   CONTROL:
   KEEP:
   QUEST:
        .
        .
        .
```

SUSPENDING UPDATE LOGGING: SUSPEND

The SUSPEND command stops the recording of processed database updates on the Update Log.

Only the master password holder or a DBA password holder with authorization for this command can issue the SUSPEND command. You cannot issue a SUSPEND command in a QUEUE/TERMINATE session.

Format

```
SUSPEND :
```

The SUSPEND command has a twofold result: it stops recording updates on the Update Log disk file and makes the Update Log unavailable. The Keepfile, however, remains unaffected; therefore, if you want to keep any updates, issue a KEEP command before suspending update logging.

If you suspend update logging for a database, and then you issue a SAVE or RESTORE command for the same database specifying the same Update Log, the recording of the processed updates starts over. You cannot apply two sets of Update Log data that have an intervening Update Log suspension. After the SAVE or RESTORE command, the processed updates can be applied from the Keepfile. Subsequent updates are recorded on the Keepfile following the applied cycles.

After you issue a SUSPEND command, the updates recorded in the Keepfile are unavailable. You can issue RELEASE and RESTORE commands for the saved database, implying the Keepfile. Then you can issue an APPLY command specifying the update cycles in the Keepfile that are to be applied to the saved database. New updates can then be executed and are recorded on the Update Log.

Other Operations That Suspend Update Logging

The following operations also suspend update logging:

- issuing a SAVE command that does not specify a Keepfile
- reloading the database with the RELOAD command
- reconstructing the database in a DEFINE session
- using load mode in a PLEX program.

After you suspend update logging, the only way you can activate it again is by issuing another SAVE or RESTORE command that activates update logging.

Note: If you activate update logging in a SAVE command, and a subsequent RESTORE command does not activate it, update logging remains activated. The RESTORE command can only activate update logging; it cannot suspend update logging.

Example

This example illustrates using a SUSPEND command in a Self-Contained Facility Multi-User environment.

```
USER, DEMO:                 (master password)
DBN IS EMPLOYEE:
CONTROL:
SAVE/:                      (specifies update logging)
     .
     .
     .
(update commands)           (The software records processed updates
                             on the Update Log disk file.)
     .
     .
     .
CONTROL:
KEEP:                       (transfers update transactions
QUEST:                       from the Update Log to the Keepfile)
     .
     .
(update commands)           (The software records processed updates
                             on the Update Log disk file.)
     .
     .
     .
CONTROL:
KEEP:                       (transfers updates to the Keepfile)
SUSPEND:                    (stops recording of the updates)
QUEST:
```

EXAMPLES OF THE SAVE/RESTORE PROCESS

The following examples illustrate using the SAVE, RESTORE, KEEP, APPLY, and RELEASE commands.

Example 1

This example shows that when update logging is activated in a SAVE command, it remains activated for subsequent RESTORE commands. Also, update logging does not have to be reactivated after an APPLY command.

```
USER, DEMO:
DBN IS EMPLOYEE:
CONTROL:
SAVE/:
QUEST:
   (update commands)
      .
      .
      .
CONTROL:  KEEP:    (allocates and deallocates the Keepfile)
RELEASE:           (makes disk database files unusable)
RESTORE EMPLOYEE: (allocates Savefile and copies database to disk)
CONTROL:
APPLY ALL:         (allocates and deallocates the Keepfile)
QUEST:
   (update commands)
      .
      .
      .
CONTROL:  KEEP:    (allocates and deallocates the Keepfile)
```

Example 2

This example shows copying new cycles 5 and 6 over old cycles 5 and 6 if they were on the Keepfile. Then the new cycle 7 is appended to the Keepfile. The APPLY command allocates and mounts the Keepfile, then applies updates through cycle 4.

```
USER, DEMO:
RESTORE EMPLOYEE:
CONTROL:
APPLY THRU CYCLE 4:  (allocates and deallocates the Keepfile)
QUEST:
IT CO*O EQ 1*1476*END*:
UPDATE CYCLE =          5
IT CO*O EQ 1*2890*END*:
UPDATE CYCLE =          6
IT CO*O EQ 1*3658*END*:
UPDATE CYCLE =          7
CONTROL:
KEEP:                (allocates and deallocates the Keepfile again)
```

Example 3

This example shows the effects of trying to activate update logging in a RESTORE command.

Case 1: Assume that you saved the database with the Keepfile specified as ML0065. The RESTORE command does not require the specification of ML0065. However, if you specify a Keepfile other than ML0065 in the RESTORE command, the new Keepfile is available for use, but you cannot issue an APPLY command.

```
USER, DEMO:
DBN IS EMPLOYEE:
CONTROL:
SAVE ON ML0064/ML0065:
QUEST:
  (update commands)
CONTROL:  KEEP:  RELEASE:
RESTORE EMPLOYEE FROM ML0064/ML0015:  CONTROL:
APPLY ALL:
-518- 'APPLY' WAS NOT PERFORMED - (batch fatal)
```

Case 2: Here the SAVE command does not activate update logging. The RESTORE command in the second job activates update logging. With this procedure, you must activate update logging every time you restore the database if updates are to be recorded.

First Job

```
USER, DEMO:
DBN IS EMPLOYEE:
CONTROL:
SAVE:
RELEASE:
EXIT:
```

Second Job

```
USER, DEMO:
RESTORE EMPLOYEE/:                    (activates update logging)
(update commands)
CONTROL: KEEP:
   .
   .
   .
```

Chapter 6

Recovering Databases with Coordinated Recovery

HOW DOES COORDINATED RECOVERY WORK? 6-1

WHEN IS COORDINATED RECOVERY ACTIVATED? 6-4

THE ROLLBACK LOG 6-5

THE UPDATE LOG 6-6

SYNCHPOINTS AND LOGICAL UNITS OF WORK 6-7
- *User Synchpoints 6-7*
- *Database Synchpoints 6-8*
- *Committing a Logical Unit of Work 6-9*
- *Multiple Concurrent Uncommitted Logical Units of Work 6-10*

ENABLING THE ROLLBACK LOG: ENABLE/DISABLE ROLLBACK 6-11
- *Alternate Methods of Suspending the Rollback Log 6-12*

RESETTING THE ROLLBACK LOG: RESET ROLLBACK AFTER 6-13
- *Database Damage Flag 6-15*

HOW DOES COORDINATED RECOVERY WORK?

With Coordinated Recovery, SYSTEM 2000 software automatically ensures the integrity of one or more databases during Multi-User update sessions. Through Coordinated Recovery, committed updates for all databases are automatically recovered in the event of a system or transaction failure.

Coordinated Recovery depends on the following conditions:

- **Update logging must be enabled.**
  The Update Log contains a recording of processed updates and information about committed updates. SYSTEM 2000 software uses the information recorded on the Update Log to recover the database.

- **The Rollback Log must be enabled.**
  The Rollback Log contains before-images of modified database pages, that is, images of pages before they were modified by an update.

- **Updates must be committed.**
  Committed updates are logical units of work delimited by synchpoints. Synchpoints can be user-specified or set automatically by SYSTEM 2000 software after certain processing has occurred. The series of updates issued between two commands that set synchpoints is a logical unit of work. When the second synchpoint is issued, the previous updates are committed. Therefore, those updates can be recovered in case the database is damaged.

If a database becomes damaged due to a transaction failure or a system failure, Coordinated Recovery takes over automatically, ensuring that all committed updates are recovered and reinstated to the database(s) involved.

The next few pages give a quick reference to terms used in the subsequent sections. These sections explain Coordinated Recovery in detail.

before-image	A record written to the Rollback Log and used when Coordinated Recovery is required; a page of a database table before the page is modified by an update.
Rollback Log	A log containing before-images of modified database pages. SYSTEM 2000 software uses the Rollback Log (along with the Update Log) during Coordinated Recovery.
Coordinated Recovery	SYSTEM 2000 processing that automatically ensures integrity of all databases that have the Rollback Log enabled when a transaction failure or a system failure occurs.
synchpoint	A beginning and ending boundary for update commands in a logical unit of work. If both a beginning and ending synchpoint exist, the updates are committed to the database.
user synchpoint	The boundary of a logical unit of work. Each user synchpoint ends a previous logical unit of work and marks the start of the next logical unit of work.
database synchpoint	A boundary for all user synchpoints on a database. A database synchpoint is also a user synchpoint.
logical unit of work	One or more updates grouped together between two synchpoints. A logical unit of work (LUW) can be either committed or uncommitted. By default in a Self-Contained Facility session, a logical unit of work is a single update command or a QUEUE/TERMINATE session, unless you issue the FRAME and END FRAME commands. By default in a PLEX program, a logical unit of work is the span of commands between the START S2K and STOP S2K commands. You can also issue the PLEX COMMIT command to commit a group of updates as a logical unit of work.
committed logical unit of work	A logical unit of work that is bounded by synchpoints, ensuring that all updates in that logical unit of work are incorporated in the database(s).

uncommitted logical unit of work

A logical unit of work that began with a synchpoint but is still in progress; that is, no closing synchpoint has occurred yet. Updates in an uncommitted logical unit of work are not committed to the database(s) affected.

primary database

The database that you are attempting to open after a system failure and that is being recovered through Coordinated Recovery. See also secondary database.

secondary database

A database involved in Coordinated Recovery after a system failure, but not the primary database. If the CORECOV parameter is set to YES, secondary databases are recovered. If the CORECOV execution parameter is set to NO, secondary databases are not recovered. See the *SYSTEM 2000 Product Support Manual, Version 12, First Edition* for more information on the CORECOV execution parameter.

system failure

A failure in the operating system, hardware, or SYSTEM 2000 software that causes normal processing to halt. A system failure triggers Coordinated Recovery when you attempt to open an affected database.

transaction failure

The failure of one update (transaction) within an uncommitted logical unit of work. A transaction failure triggers Coordinated Recovery of the database(s) involved; other processing continues during the recovery. See also system failure.

Update Log

A log containing a recording of processed updates and information about commitment of logical units of work and affected databases. Used in producing a Keepfile for user-specified restoration of a database and also for Coordinated Recovery by SYSTEM 2000 software.

WHEN IS COORDINATED RECOVERY ACTIVATED?

Coordinated Recovery is activated when SYSTEM 2000 software detects damage to a database (when you open the database). A database becomes damaged from the following kinds of failures:

- **transaction failure**
  Transaction failure is the failure of an update (transaction) within an uncommitted logical unit of work. Transaction failure causes SYSTEM 2000 software to dynamically undo one or more logical units of work during normal processing. Coordinated Recovery takes place when you attempt to open one (or more) of the databases that was being updated.

- **system failure**
  System failure is a failure in the operating system, hardware, or SYSTEM 2000 software that causes normal processing to halt. Following a system failure, Coordinated Recovery takes place when you attempt to open the damaged database.

The recovery process is outlined below.

1. You try to open a damaged database.

2. SYSTEM 2000 software places a hold on the database you are trying to open. (This is the primary database.)

   SYSTEM 2000 software then issues informative messages to the Message File and to the operator console (WTOs). These messages give the name of the damaged database, the cycle number of the Rollback Log, and the job name, step name, program name, or terminal ID that initiated the recovery. Informative messages also display the job name, step name, program name or terminal ID of the user whose updates are being skipped in the recovery process.

3. SYSTEM 2000 software analyzes the Update Log of the primary database. This analysis determines which logical units of work must be removed from the database and identifies any other damaged databases.

4. If other damaged databases are found, steps 2 and 3 are repeated for each database. If the execution parameter CORECOV equals NO, secondary databases are not recovered. SYSTEM 2000 software proceeds with the next step.

5. For each database that has one or more uncommitted logical units of work, SYSTEM 2000 software

   a) rolls the database back using before-images from the Rollback Log.

   b) rolls the database forward, applying transactions from the Update Log. Updates from uncommitted logical units of work are not applied.

6. SYSTEM 2000 software then drops all holds and issues informative messages that display the name of the recovered database and its cycle number after the recovery.

You can then continue with normal processing.

THE ROLLBACK LOG

The Rollback Log (File 8) logs the before-images of modified database pages. The Rollback Log can contain before-images for more than one logical unit of work. The logical units of work can belong to one or more users and can be overlapping and concurrent.

The Rollback Log for each database must be enabled before SYSTEM 2000 software can perform Coordinated Recovery. The Rollback Log must be formatted for BDAM, and it can be a multi-volume file. The Rollback Log must have a block size that is at least 12 bytes greater than the largest block size for database File 1 through File 6. If SYSTEM 2000 software cannot format the file with this block size for BDAM, a message appears.

Whenever a new logical unit of work is begun on a database, the Rollback Log is not reset to its beginning. Rather, the before-images of the new logical unit of work are written after the before-images of the previous logical unit of work. Thus, the Rollback Log becomes a continuous log of before-images; that is, it contains before-images of multiple logical units of work.

A discussion of how to control the number of logical units of work that are allowed to stack up on the Rollback Log appears in **Resetting the Rollback Log: RESET ROLLBACK AFTER** on page 6-13. This stacking of before-images on the Rollback Log allows SYSTEM 2000 software to remember before-images taken in earlier logical units of work. Because the Rollback Log is not reset for each logical unit of work, before-images of database pages caused by earlier logical units of work are used by subsequent database pages. This means that a database page has only one before-image taken while it remains in the buffer into which it was read. Therefore, fewer I/Os may be required for before-image logging if multiple logical units of work update the same page. This can be significant for users who have default logical units of work for one database cycle. For example, if many data trees of the same structure are inserted over and over, and the updates modify the same database pages, only one before-image of each page is taken as long as that page's buffer stays in memory.

THE UPDATE LOG

The Update Log (File 7) must be active if the Rollback Log is enabled. However, if you allocate File 7 and you do not activate update logging in a SAVE or RESTORE command, SYSTEM 2000 software activates it internally. It is in use for recovery information. The Update Log must be formatted for BDAM, and it can be a multi-volume file. It contains the following types of data:

- processed update records that are used during recovery and APPLY command processing
- header records for logical units of work
- link records for logical units of work; used only when more than one database is updated within a logical unit of work
- trailer records for logical units of work
- verify records for logical units of work; used only when more than one database is updated within a logical unit of work.

Each record type written to the Update Log has an eight-byte user identification field at the beginning of the record. This field uniquely identifies a logical unit of work. Records generated in the same logical unit of work have the same user identification. This identification is critical in the recovery of a database.

SYNCHPOINTS AND LOGICAL UNITS OF WORK

Each synchpoint in a SYSTEM 2000 session causes updates in the preceding logical unit of work to be committed to databases having the Rollback Log enabled. A synchpoint signals the beginning of a logical unit of work and commits the previous logical unit of work. There are two general types of synchpoints: user synchpoints and database synchpoints. A user synchpoint pertains to one user's job; a database synchpoint affects the entire database and all users who are accessing that database.

User Synchpoints

The commands listed below cause a user synchpoint. In addition, any command that causes a database synchpoint also causes a user synchpoint.

```
CLOSE  (PLEX only)
COMMIT  (PLEX only)
DROP HOLD  (PLEX only)
END FRAME  (Self-Contained Facility and PLEX)
FRAME  (Self-Contained Facility and PLEX)
QUEUE  (Self-Contained Facility and PLEX)
ROLLBACK (PLEX only)
START S2K  (PLEX only)
STOP S2K  (PLEX only)
TERMINATE  (Self-Contained Facility and PLEX)
```

You can issue the COMMIT and ROLLBACK commands in PLEX programs to set user synchpoints. The COMMIT command causes the current uncommitted logical unit of work to be committed.

The CLOSE command causes a user synchpoint, because it causes the database to be detached from the user's job, thereby erasing all of the user's local holds. The PLEX commands START S2K and STOP S2K cause a user synchpoint, because they are the boundaries of a PLEX program's communication with SYSTEM 2000 software. The DROP HOLD command in PLEX causes a user synchpoint, because the user's holds on the database are dropped.

The FRAME, END FRAME, and QUEUE commands cause a user synchpoint because they drop all of the user's local holds. Therefore, the logical unit of work must be committed before the local holds are dropped. The END FRAME command causes a user synchpoint, because it commits the updates done since the previous FRAME command. When the TERMINATE command processing completes successfully, a user synchpoint occurs, because it commits the updates done in the QUEUE/TERMINATE session. The Rollback Log is suspended for PLEX load mode. However, after TERMINATE processing completes, the Rollback Log starts again at a new synchpoint. For information on the FRAME and END FRAME command, see *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition.*

Database Synchpoints

A database synchpoint lasts for the duration of the command (or process) causing the synchpoint, and it causes a user synchpoint to occur. The following commands and processes cause a database synchpoint to occur:

- KEEP (Self-Contained Facility and PLEX)
- SAVE (Self-Contained Facility and PLEX)
- PLEX LOAD (PLEX only)
- ENABLE/DISABLE ROLLBACK (Self-Contained Facility and PLEX)
- any command that causes a global hold
- a reset of the Rollback Log.

The KEEP command causes a database synchpoint, because there can be no uncommitted logical units of work for the database when writing the Update Log records to the Keepfile. Coordinated Recovery relies on the Update Log to apply updates; issuing a KEEP command erases these data and makes it impossible to apply the updates necessary for Coordinated Recovery. Therefore, a database synchpoint is required to ensure that no data on the Update Log are needed for Coordinated Recovery.

The rationale for the SAVE command causing a database synchpoint is the same as for KEEP. A SAVE command causes the Update Log to be discarded. Therefore, a database synchpoint is required to ensure that there are no data required (for Coordinated Recovery) on the Update Log.

The PLEX LOAD command (load mode) causes a database synchpoint also. However, this is not significant because exclusive use is required for PLEX load mode and other users do not have access to the database. The PLEX LOAD command is included on the list of database synchpoints mainly because it causes a reset of the Rollback Log. (See also the section in this chapter on Resetting the Rollback Log.)

Any command that obtains a global hold on a database causes a database synchpoint. This includes all Self-Contained Facility update commands, as well as PLEX queue mode holds and the FRAME command in both PLEX and the Self-Contained Facility. For a Self-Contained Facility session, only one database is open at a time. Each QUEST update command causes a global hold on the database. When the update completes, the global hold is dropped and a synchpoint occurs, committing the update. If the update fails, the database is recovered. Other Self-Contained Facility users can be updating the database (interleaving their global holds). PLEX users can retrieve from the database, but they cannot update it. In other words, unless FRAME/END FRAME is issued, a logical unit of work is one update command or one QUEUE/TERMINATE session.

The ENABLE/DISABLE ROLLBACK command causes a database synchpoint and a reset of the Rollback log (if enabled). It also commits any updates performed by the user since the last user synchpoint in the affected database(s).

Committing a Logical Unit of Work

For each synchpoint, SYSTEM 2000 software assigns a unique ID. This ID is added to all records written to the Update Log for that user's logical unit of work.

For example, in a Self-Contained Facility session, the following situation might occur:

1. You issue the FRAME command, causing a synchpoint and assigning a unique synchpoint ID. If a previous logical unit of work exists, the updates are committed.

2. You issue the first update, and a header record is written to the Update Log.

3. The Update Log buffers are cleared, ensuring that the header is on disk.

4. The standard Update Log data are transferred to the Update Log buffer.

5. SYSTEM 2000 software processes the update command against the database.

6. If database pages are modified, SYSTEM 2000 software writes their before-images to the Rollback Log.

7. SYSTEM 2000 software processes subsequent updates to the database in a similar manner. The only difference is that the header record is not written again and the Update Log buffers are not explicitly cleared. (They are cleared as they become full or when no buffer is available.)

   If another database is included in the logical unit of work, a link record is written to the Update Log of all databases currently in the logical unit of work. This record indicates the presence of the new database in the logical unit of work. Also, the writing of header, link, trailer, and verify records occurs for each database modified in the logical unit of work.

8. You issue the END FRAME command, and the logical unit of work is committed.

   A trailer record is written to the Update Log of each database that was updated in the logical unit of work.

9. The Update Log buffers for each affected database are cleared. (When the Rollback Log is enabled, clearing is directed by SYSTEM 2000 software. The CLEAR commands only set the mode of clearing and no actual clearing is done by the CLEAR commands.)

10. A verify record is written to the Update Log of each database involved. Verify records are used during Coordinated Recovery only when more than one database is in the logical unit of work. No explicit clearing is done after the verify records are written.

The steps described above are repeated for each user synchpoint.

Multiple Concurrent Uncommitted Logical Units of Work

Any number of users can have uncommitted logical units of work on a database at the same time. SYSTEM 2000 software ensures that any record held or updated in an uncommitted logical unit of work cannot be held or updated by another user until the first user's logical unit of work is committed. This commitment occurs at the first user's next synchpoint.

After the updates are committed, the holds on those records are dropped, and the records are then available to other users. By ensuring that no two uncommitted logical units of work can hold or update the same record, SYSTEM 2000 software can skip an uncommitted logical unit of work during recovery without affecting any other committed or uncommitted logical unit of work.

The records that are held in a logical unit of work are

- records explicitly held via the PLEX/HOLD option
- records whose data have been modified or removed
- records being inserted
- records being removed.

Notice that User A can retrieve a record held or updated by User B if User A does not use the /HOLD option on his retrievals. Additionally, where-clause processing ignores whether a record is held by another user. Therefore, it is possible to qualify records updated by another user's uncommitted logical unit of work. If that logical unit of work is later removed from the database for some reason, an identical where-clause may yield different results.

ENABLING THE ROLLBACK LOG: ENABLE/DISABLE ROLLBACK

The ENABLE ROLLBACK command enables the Rollback Log for a specified database and allows Coordinated Recovery for the database. The DISABLE ROLLBACK command disables the Rollback Log and prevents Coordinated Recovery.

Only the master password holder or a DBA password holder with authorization for this command can issue the ENABLE/DISABLE ROLLBACK command. You must also have exclusive use of the database.

Format

```
ENABLE ROLLBACK  :
DISABLE ROLLBACK  :
```

The ENABLE ROLLBACK command enables the Rollback Log for a specific database so that SYSTEM 2000 software can record before-images of modified database table pages. However, the ENABLE ROLLBACK command does not by itself cause Coordinated Recovery to occur. See **How Does Coordinated Recovery Work?** on page 6-1.

The effect of an ENABLE ROLLBACK or DISABLE ROLLBACK command lasts for all subsequent sessions accessing the database until the opposite command is issued.

Each database has a separate Rollback Log (database File 8). File 8 is a BDAM file, and it can be a multi-volume file. Users of a given database share the same Rollback Log. A DD statement (or an ALLOC command) must be provided for the Rollback Log for each database having the Rollback Log enabled. Tapes are not allowed for the Rollback Log.

The Rollback Log can be enabled or disabled in the Self-Contained Facility or in PLEX. For details about the PLEX commands, as well as other PLEX commands related to Coordinated Recovery, see the *SYSTEM 2000 PLEX Manual, Version 12, First Edition.*

If the Rollback Log is enabled, SYSTEM 2000 software automatically places a local hold on every record updated by a PLEX program unless the user has a global hold or exclusive use. For a REMOVE TREE operation, a hold is placed on every record to be removed plus the node of attachment before the actual removal process begins. For a MOVE TREE operation with the Rollback Log enabled, a hold is automatically placed on the top node of the tree being moved, on the node of detachment, and on the node of attachment.

Considerations

- If recovery of a database is impossible (for example, due to a permanent I/O error), issue DISABLE ROLLBACK **before** opening the database. This technique is known as a "deferred disable." It defers any disabling until the next attempt to open a database. This allows opening the database with the Rollback Log enabled and issuing a KEEP command. Without deferred disable there is no method of requesting any action on the database if recovery is impossible.

- You cannot issue the ENABLE ROLLBACK or DISABLE ROLLBACK commands for a damaged database.

- Buffer pool specifications need not be changed for the Rollback Log, because normal operations do not require any extra buffers. Writing before-images to the Rollback Log uses the database buffers directly. During Coordinated Recovery, the software uses the pool with the largest buffer size for database file usage.

- Rollback Log I/O is considered database I/O.

- If the Rollback Log is enabled for a database, the PLEX MULTIPLE HOLDS option is automatically set to allow multiple local holds. If the Rollback Log is subsequently disabled, the MULTIPLE HOLDS option reverts to its previous setting.

- For a database with the Rollback Log enabled, database pages are cleared only when the Rollback Log is reset.

- If the Rollback Log is enabled for a database, the Update Log buffers are cleared whenever a synchpoint occurs, that is, when a logical unit of work completes.

- If the Rollback Log is enabled for a database, LHOLD=YES is ignored. See the *SYSTEM 2000 Product Support Manual, Version 12, First Edition* for details about the LHOLD execution parameter.

Alternate Methods of Suspending the Rollback Log

Suspending the Rollback Log means that before-images are not written to the log. In addition to issuing the DISABLE ROLLBACK command, the Rollback Log is suspended in the following cases:

- during RELOAD processing. The Rollback Log is reset at the beginning and end of a RELOAD operation.

- during a DEFINE session that causes restructuring. The Rollback Log is reset at the beginning and end of the restructuring operation.

- during PLEX load mode. The Rollback Log is suspended prior to the TERMINATE command. After TERMINATE is issued by the PLEX program, SYSTEM 2000 software resumes its writing of before-images.

Note: CREATE INDEX, REMOVE INDEX, and REORGANIZE do not suspend the Rollback Log. Before-images of modified pages are recorded. These commands cause a reset of the Rollback Log at the beginning and end of processing, because update records for these commands are not written to the Update Log. Therefore, it is impossible to roll forward during a recovery. SYSTEM 2000 software resets the Rollback Log at the beginning and end of the command, so the Update Log is not needed.

RESETTING THE ROLLBACK LOG: RESET ROLLBACK AFTER

The RESET ROLLBACK AFTER command controls the amount of primary space used or the number of user synchpoints reached before the Rollback Log is reset. When the threshold is reached, SYSTEM 2000 software allows all logical units of work (LUWs) in progress to continue; new LUWs will be temporarily suspended until the Rollback Log is reset. This command determines the amount of data to be written to the Rollback Log and, therefore, the amount of recovery that occurs in the event of a transaction failure or a system failure.

Only the master password holder or a DBA password holder with authorization for this command can issue the RESET ROLLBACK AFTER command. You must also have exclusive use of the database.

Format

```
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 the primary allocation is used. The default is 50 percent.

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.

You can issue the RESET ROLLBACK AFTER command when the Rollback Log is either enabled or disabled (see the ENABLE/DISABLE ROLLBACK command). The values specified in PERCENT and SYNCHPOINT are stored in the Master Record of the specified database and become effective only when the Rollback Log is enabled. If one or both thresholds are not specified, the default value replaces the value stored in the Master Record. New thresholds become immediately effective if the Rollback Log is enabled when the RESET ROLLBACK AFTER command is issued.

If you specify a PERCENT threshold of zero, the Rollback Log is reset at the start of each logical unit of work. If the PERCENT threshold is 100, the Rollback Log is reset at the start of the logical unit of work after the log's primary allocation has been filled. The first user synchpoint that causes a logical unit of work to occur after the SYNCHPOINT threshold has been reached causes a reset of the Rollback Log.

Abbreviations:

PERCENT = PCT
ROLLBACK = RB
SYNCHPOINTS = SYNCHPOINT, SPTS, or SPT

Resetting the Rollback Log begins after a threshold is reached or after a command or operation that causes a reset of the Rollback Log. The following events take place at the occurrence of a logical unit of work inside a new user synchpoint:

1. A database synchpoint is created. No new logical units of work are permitted to start until after the Rollback Log is reset. The reset can occur without requiring everyone to relinquish the database. If the current active logical units of work (LUWs) have local holds but no updates, the reset is allowed. If an LUW has updates, the reset request is dropped, and the software will attempt the reset again when the status on that database changes.

2. The database is cleared, which writes any updated database pages currently in the buffers to disk and turns off the database damage flag.

3. SYSTEM 2000 software resets the Rollback Log to its beginning and writes the database Master Record to the Rollback Log.

4. The database synchpoint ends and either the logical unit of work (with its synchpoint) or the operation that triggered the reset continues.

A global hold causes both a database and a user synchpoint to occur. Global holds are caused by commands, such as

- Self-Contained Facility update commands
- FRAME command in both the Self-Contained Facility and PLEX
- queue mode holds in PLEX
- SAVE and KEEP commands.

In addition to causing both a database and a user synchpoint to occur, the following commands or operations also cause a reset of the Rollback Log:

- exclusive open of a database
- physical close of a database
- KEEP command
- SAVE command
- PLEX LOAD command
- TERMINATE command for a PLEX LOAD command
- at both the beginning and end of processing for
  - a CREATE INDEX command
  - a REMOVE INDEX command
  - a REORGANIZE command
  - a RELOAD command
  - a DEFINE session with restructuring.

Database Damage Flag

Database pages are cleared only when the Rollback Log is reset or the database is physically closed. Notice that beginning or ending a logical unit of work does not clear the database pages. The first time a logical unit of work header is written to the Update Log after the physical opening of the database or resetting of the Rollback Log, the Master Record is written to disk with the damage flag set. The damage flag remains set for a database until the Rollback Log is reset. Resetting the Rollback Log clears the database pages and clears the damage flag to disk. If the damage flag in memory shows no damage to the database, then the damage flag on disk also shows no damage after the clear occurs.

For example, suppose that you open a database that has the Rollback Log enabled. The first update to the database causes the damage flag to be set on disk. No further clears are done for the database until the Rollback Log is reset, regardless of the updates, synchpoints, or logical units of work done on the database. When the Rollback Log is reset or you physically close the database (which resets the Rollback Log), the database pages are cleared. The damage flag currently residing in memory is written to disk. This causes the disk damage flag to be turned off, assuming that the database is logically intact at the time the database is cleared.

The clearing procedure described above means that the overhead of having logical units of work is limited to an I/O on the Update Log at the beginning and end of the logical unit of work. However, there is also the overhead for writing the before-images to the Rollback Log.

Example

The following example causes a reset of the Rollback Log via the PERCENT threshold:

```
EXCLUSIVE DBN IS EMPLOYEE:          (opens the EMPLOYEE database; the
        .                            Rollback Log is enabled and is 49 percent
        .                            full; the PERCENT threshold is at 50.)
        .
FRAME:
        .
        .
        .
(updates to EMPLOYEE)               (EMPLOYEE updates fill the Rollback Log
        .                            to 60 percent of its primary space
        .                            allocation.)
        .
        .                           (commits the logical unit of work
FRAME:                              consisting of the updates since the
        .                           previous synchpoint; establishes a new
        .                           synchpoint.)
        .
        .
        .
(updates to EMPLOYEE)
        .
        .
        .
```

Note: The first update after the FRAME command starts a logical unit of work. The logical unit of work triggers a reset of the Rollback Log, because the PERCENT threshold was exceeded in the previous logical unit of work. The Rollback Log is reset, the Master Record is written to the log and then the update is done, causing the necessary before-image to be recorded. The header and update cycle are written to the Update Log.

Chapter 7

Ensuring Database Security

INTRODUCTION 7-1

ASSIGNING THE MASTER PASSWORD: USER 7-3
Master Password Commands Regarding Security 7-4
The Master Password and Damaged Databases 7-5
Summary of Commands Requiring the Master Password 7-5

ASSIGNING SECONDARY PASSWORDS: VALID PASSWORD IS 7-6

ASSIGNING AND CHANGING AUTHORITIES: ASSIGN AUTHORITIES 7-7

ASSIGNING SECURITY BY ENTRY: ENTRY KEY IS 7-11

ASSIGNING THE DBA PASSWORD: DBA PASSWORD IS 7-14

CONTROLLING DBA PASSWORD COMMANDS: ENABLE/DISABLE DBA FOR 7-15

INVALIDATING PASSWORDS: INVALID PASSWORD IS 7-16

CHANGING PASSWORDS: CHANGE PASSWORD 7-17

LISTING PASSWORDS: LIST PASSWORDS 7-18

LISTING SECONDARY PASSWORDS AND AUTHORITIES: LIST PASSWORDS AND AUTHORITIES 7-19

LISTING THE DBA PASSWORD: LIST DBA 7-20

LISTING THE DBA PASSWORD AND COMMANDS: LIST DBA AND COMMANDS 7-21

INTRODUCTION

Database security is an important part of SYSTEM 2000 software. You implement security by assigning database passwords. They limit and control access of stored data and knowledge of the database structure by restricting update and retrieval capabilities.

Every time you open a database, you must give a password. There are three types of passwords.

- **a master password**
  The master password holder has access to all commands and to all components in a database.

- **secondary passwords**
  Secondary password holders can issue only a subset of SYSTEM 2000 commands and can be further restricted on a component-by-component basis for retrieval authority only, update authority only, qualification criteria authority only, no access, or any combination of these authorities.

- **a DBA (Database Administrator) password**
  A DBA password holder can automatically issue system-wide commands and the CONTROL language KEEP command. A DBA password holder can be authorized to issue other CONTROL language commands from a set of available commands. A DBA password holder cannot issue any commands that access data or any DEFINE language commands.

There is only one master password for each database. You assign the master password with the USER command followed by the NEW DATA BASE IS command. The master password holder then has the option of assigning secondary passwords and a DBA password.

ASSIGNING THE MASTER PASSWORD: USER

The USER command in combination with the NEW DATA BASE IS command assigns the master password. The USER command also accesses SYSTEM 2000 software.

Format

```
USER, password :
```

password consists of 1 to 4 alphanumeric characters with no blanks. The password can also be a single special character or lowercase letter. See the SEPARATOR IS command in *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for the list of special characters.

The USER command, followed by the NEW DATA BASE IS command, assigns the master password for that database. The master password holder controls the database.

When you issue the DATA BASE NAME IS command, SYSTEM 2000 software checks to ensure that the password specified in the USER command is authorized to access the database. For more details see **Accessing SYSTEM 2000 Software: USER** on page 2-2.

After a database is defined, the master password holder automatically receives the R, W, and U authorities for all components. R-authority allows **retrieval** operations. U-authority allows **update** operations. W-authority allows **if-clause** and **where-clause** operations. N-authority is considered **no-access** authority and is used by the master password holder to remove authorities for secondary passwords. N-authority cannot be used to remove authorities from the master password. In addition, the master password holder has unrestricted use of all SYSTEM 2000 commands.

There is no method of listing the master password. If you misplace or forget the master password, see your SYSTEM 2000 Software Representative, who can contact the Institute for help in deciphering your master password.

Examples

Example 1

```
USER,AB:
NEW DATA BASE IS CAR POOL:
```

Example 2

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

Example 3

```
USER,499C:
NEW DATA BASE IS PERSONNEL:
```

Master Password Commands Regarding Security

The master password holder controls access to the database by both secondary password holders and the DBA password holder.

The master password holder controls other password holders with the following commands. No one else can issue these commands.

ASSIGN authorities	assigns or changes retrieval, update, and where-clause authorities for the secondary passwords. The ASSIGN authorities command can also remove all previously assigned authorities for secondary passwords.
CHANGE PASSWORD	changes any password to a new password.
DBA PASSWORD IS	assigns a DBA password.
DISABLE DBA FOR	makes one or all of the authorized commands unavailable to the DBA password holder.
ENABLE DBA FOR	authorizes the DBA password holder to use one or all of the available commands.
ENTRY KEY IS	restricts the access of secondary password holders to only specific logical entries.
INVALID PASSWORD IS	removes a secondary or DBA password from the database.
LIST DBA	lists the DBA password.
LIST DBA AND COMMANDS	lists the DBA password and its authorized commands.
LIST PASSWORDS	displays all secondary passwords.
LIST PASSWORDS AND AUTHORITIES	displays all secondary passwords and the authorities assigned to each component.
VALID PASSWORD IS	assigns a new secondary password.

The Master Password and Damaged Databases

No user, not even the master password holder, can update a damaged database. However, the master password holder and the DBA password holder can release and reload a damaged database using the RELEASE and the RELOAD commands. Also, any user except a DBA password holder can retrieve against a damaged database.

The master password holder can make changes to database security for a damaged database by issuing the commands listed in the preceding topic.

Summary of Commands Requiring the Master Password

This list shows the Self-Contained Facility commands that require the master password:

```
APPLY
ASSIGN authorities
CHANGE PASSWORD
CREATE/REMOVE INDEX
DBA PASSWORD IS
DEFINE
ENABLE/DISABLE DBA FOR
ENABLE/DISABLE MULTIPLE HOLDS
ENABLE/DISABLE ROLLBACK
ENABLE/DISABLE VALUES PADDING
ENTRY KEY IS
FULL PASSES
LIST DBA
LIST DBA AND COMMANDS
LIST PASSWORDS
LIST PASSWORDS AND AUTHORITIES
PRINT SIZE
RELEASE
RELOAD
REORGANIZE
RESET ROLLBACK AFTER
RESTORE
SAVE
SUSPEND
TALLY with Security by Entry
VALID/INVALID PASSWORD IS
```

ASSIGNING SECONDARY PASSWORDS: VALID PASSWORD IS

The VALID PASSWORD IS command assigns a secondary password for a database.

Only the master password holder can issue the VALID PASSWORD IS command.

Format

```
VALID PASSWORD IS  password  :
```

password consists of 1 to 4 alphanumeric characters with no blanks. The password can also be a single special character or lowercase letter. See the SEPARATOR IS command in *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for the list of special characters.

You assign secondary passwords for retrieving, updating, or doing if-clause or where-clause operations on a component-by-component basis. Secondary passwords are restricted to a subset of available SYSTEM 2000 commands.

You assign only one secondary password in a VALID PASSWORD IS command at a time, and each secondary password must be unique for a database. The maximum number of secondary passwords per database is 200.

Assigning a secondary password does not automatically assign any access authority to the password. You assign access authority to specific components in the database with the ASSIGN authorities command. See **Assigning and Changing Authorities: ASSIGN Authorities** on page 7-7.

Secondary password holders cannot issue any of the commands listed on the previous page.

Examples

```
VALID PASSWORD IS ABLE:

VALID PASSWORD IS +:

VALID PASSWORD IS CAM:

VALID PASSWORD IS X9:
```

ASSIGNING AND CHANGING AUTHORITIES: ASSIGN AUTHORITIES

The ASSIGN authorities command assigns, changes, or removes access authorities for secondary passwords.

Only the master password holder can issue the ASSIGN authorities command for a database.

Format

```
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.

R-authority allows retrieval of specific components.

U-authority allows updating of specific items or records.

W-authority allows specific items or records to be given in where-clauses and if-clauses.

N-authority prevents knowledge of specific components and removes all previously assigned authorities.

component is a component name or component number.

passwords is a list of one or more secondary passwords, separated by commas.

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

The authorities you assign to a secondary password depend on the secondary password holder's operational responsibilities. To allow a secondary password holder to reference a component (an item, record, string, or function) by name or number, you assign an authority for that user to use that component. Which authority you assign depends on how the secondary password holder will use a component. (See the four types of authorities on the previous page.)

For example, if a secondary password holder, ABLE, wants to retrieve the values for items C1 and C2 and qualify that retrieval using C12, the command is

```
PRINT C1, C2 WHERE C12 EXISTS:
```

You must first assign authorities to these components using the following commands:

```
ASSIGN R TO C1, C2 FOR ABLE:
ASSIGN W TO C12 FOR ABLE:
```

If the secondary password holder, ABLE, attempts to issue the next command, SYSTEM 2000 software issues an error message, because C2 was not assigned update authority.

```
REMOVE C2 WHERE C12 EXISTS:
```

If a secondary password holder does not have authority for all components, C0 cannot be used in update or retrieval operations. However, you can assign authority individually to C0 in the same way it is assigned to other components.

Assigning an authority to a schema record does not automatically give that same authority to all of its items. A request to print a data record prints only those items for which the requesting password has been assigned R-authority. If you assign U-authority to a record, its associated items and descendant data records can be removed without U-authority assignments for each item. However, to insert a new data record, U-authority must have been assigned to each schema item individually.

If you want a secondary password holder to have all authorities for all components, the following command would be appropriate:

```
ASSIGN R,U,W TO ALL COMPONENTS FOR ABLE:
```

You can assign authorities for an inclusive range of components with one ASSIGN authorities command. The following command assigns retrieval, where-clause, and if-clause authorities for the database components C2 through C10, inclusively in DESCRIBE order, for password ABLE.

```
ASSIGN R, W TO C2 THRU C10 FOR ABLE:
```

The DESCRIBE command, which displays the database definition, lists only those components that have been assigned one or more authorities for the secondary password in use. Other components are transparent to that password.

You assign authorities for component numbers or names that reference strings and functions in the same manner as items and records. Strings and functions usually refer to other components within their definitions. For example,

```
1000* ABC (STRING (PRINT C1, C2 WHERE C12 EXISTS:))
```

String ABC (or C1000) refers to three items in its definition. Simply accessing a string or function is possible with any authority. However, expansion of the string or function still depends on authorizations for the individual components. You could assign access to string ABC in the example above for password ABLE, as follows:

```
ASSIGN R TO C1000 FOR ABLE:
```

However, unless secondary password ABLE has been assigned the proper authorities for the components C1, C2, and C12, as well as the string, ABLE could not use the string in the QUEST language.

Any or all of the allowable 200 secondary passwords can be assigned authorities for as many components in the definition as necessary.

If a secondary password holder tries to access a component without having the necessary authority, SYSTEM 2000 software issues an error message.

Any statement of authorities for a secondary password totally replaces any existing authorities. Therefore, when you are changing authorities for a secondary password, the new authorities must include any previous authorities that are to remain in effect.

For example, if you want to add W-authority to an existing R-authority, both R and W must be specified. You can specify R, U, or W in any order and in any combination. The N-authority, however, cannot appear with other authorities in the authorities list. If you specify the N-authority with other authorities, SYSTEM 2000 software issues an error message.

Rules

The following considerations apply to the THROUGH option:

- You cannot combine the THROUGH option with a list of components.
- You can specify the THROUGH option only once in a given command.
- THROUGH is not a reserved word, so it can be part of a component name. However, it is a keyword; it indicates the THROUGH option to the software. Therefore, you should not specify in an ASSIGN command any component name that includes the word THROUGH; use the component number instead.
- The two components you specify in the THROUGH option provide an inclusive range. They must appear in DESCRIBE order.

Examples

```
ASSIGN R TO C101 FOR ABLE:

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

ASSIGN R TO C201 THRU C206 FOR BAKE:

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

ASSIGN N TO C142,C145 FOR +:

ASSIGN N TO C10,C11 FOR 994,CHAR:
```

ASSIGNING SECURITY BY ENTRY: ENTRY KEY IS

The ENTRY KEY IS command restricts the access of secondary password holders to specific logical entries, based on the unique values of a declared level 0 Entry Key item.

Only the master password holder can issue the ENTRY KEY IS command.

Format

```
ENTRY KEY IS item  :
```

```
ENTRY KEY IS |ENTRY                    :
             |C0
             |user-defined-C0-name
```

item	is a level 0 key item name or component number designated as the Entry Key.
ENTRY	cancels Security by Entry.
C0	cancels Security by Entry.
user-defined-C0-name	is an ENTRY record that has been renamed; cancels Security by Entry.

The item specified as the Entry Key must be a level 0 key item; the Entry Key cannot be a schema record or C0. Each value for the Entry Key item must be unique for each logical entry, like social security number or an employee ID number. The secondary password holder must know the unique value for the specific item in a logical entry in order to gain access to other data in that logical entry.

After a secondary password holder gains access to a logical entry, he can access only that one logical entry. However, the master password holder has access to the entire database. After the Entry Key item has been specified, Security by Entry is in effect for the database for all jobs.

The only exception to Security by Entry, besides the master password, is a secondary password that begins with a numeral (0-9). That is, a secondary password, such as 2AA, can access all logical entries, subject to the specific R-, U-, and W-authorities assigned to password 2AA.

To cancel Security by Entry for a database, issue ENTRY KEY IS ENTRY, C0, or *user-defined C0-name*.

To access a logical entry, the secondary password holder must specify the Entry Key item in the first condition of a QUEST where-clause the first time the database or entry is accessed. The first condition in the QUEST where-clause must consist of the level 0 Entry Key item and a unique value with the EQ operator. If more than one level 0 record satisfies this condition (or none), Security by Entry processing does not honor the request for that command or subsequent commands.

After a logical entry is identified, the Entry Key condition does not need to be specified again. Subsequent commands are processed like any other command, except for one significant difference: the database has now been reduced to one specific logical entry. If the secondary password holder wants to change logical entries, he must give the Entry Key condition again with a new value, as the first condition of a new where-clause.

The Entry Key condition cannot be specified within a QUEUE/TERMINATE session. It is honored by the QUEUE processor, but the logical entry must be identified by a QUEST where-clause before you call the QUEUE processor.

Any R-, U-, W-, or N-authorities in effect for the secondary password are tested for the secondary password before the Entry Key security is tested.

Do not assign U-authority to the Entry Key item for any secondary password, in order that no secondary password holder can update the values for the Entry Key.

Only the master password holder (or the holder of a secondary password beginning with a numeral) can issue the TALLY command when Security by Entry is in effect for a database.

The MOVE TREE command allows a data tree to be moved from one entry to another even though Security by Entry is in effect. As long as the appropriate values for the Entry Key are used, the MOVE TREE command is processed. The logical entry available for access after processing is the entry qualified by the "to" where-clause.

With Security by Entry in effect for a database, a where-clause requiring a full pass of the database is allowed only if ALLOW FULL PASSES is in effect. However, the range of the qualified records is limited to one specific entry. For example, Case A and Case B below qualify the same records:

Case A: partial pass (allowed)

```
...WHERE Entry-Key EQ value AND non-key-expression:
```

Case B: full pass not done for second where-clause

```
...WHERE Entry-Key EQ value:
...WHERE non-key-expression:
```

Examples

Example 1

The following example establishes an Entry Key and accesses the database with a secondary password. Master password holder ABLE declares C1 as the Entry Key, which causes Security by Entry to be in effect for the EMPLOYEE database. If C1 is EMPLOYEE NUMBER, then each value of C1 must be unique. The master password holder determines which secondary password holders can have access to specific employees. If secondary password holder USRA can access employee number 1265, then the Entry Key condition C1 EQ 1265 in line (1) establishes the logical entry for that employee as the user "area" for commands shown on lines (2) through (4). Line (5) establishes a new logical entry (user "area") for employee number 1120.

```
    USER, ABLE:                          (master password)
    DBN IS EMPLOYEE:
    CONTROL:
    ENTRY KEY IS C1:                     (establishes C1 as the Entry Key)
    VALID PASSWORD IS USRA:              (creates secondary password USRA)
    ASSIGN R, W TO ALL COMPONENTS FOR USRA:
    ASSIGN U TO C4, C105, C106 FOR USRA:
    EXIT:
                              * * * *


    USER, USRA:                          (secondary password)
    DBN IS EMPLOYEE:                     (Security by Entry in effect)
(1) PRINT C2, C3, C4 WH C1 EQ 1265:      (sets "area" to EMPLOYEE
                                          NUMBER 1265)
(2) PRINT C100:                          (for EMPLOYEE NUMBER 1265 only)
(3) PRINT C120 WHERE C121 EXISTS AT 1:   (for EMPLOYEE NUMBER 1265 only)
(4) PRINT C110 WHERE SAME:
(5) PRINT C0 WHERE C1 EQ 1120:           (changes "area" to EMPLOYEE
                                          NUMBER 1120)
    EXIT:
```

Example 2

This example cancels the Security by Entry capability for the EMPLOYEE database. ABLE is the master password.

```
    USER, ABLE:
    DBN IS EMPLOYEE:
    CONTROL:
    ENTRY KEY IS C0:
    EXIT:
```

ASSIGNING THE DBA PASSWORD: DBA PASSWORD IS

The DBA PASSWORD IS command assigns a DBA password so that the DBA password holder can perform database administrator functions.

Only the master password holder can issue the DBA PASSWORD IS command. A database can have only one DBA password, and it must be unique.

Format

```
DBA PASSWORD IS  password  :
```

password consists of 1 to 4 alphanumeric characters with no blanks. The password can also be a single special character or lowercase letter. See the SEPARATOR IS command in *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for the list of special characters.

When the master password holder first assigns a DBA password, its holder can immediately issue any system-wide command and the CONTROL language KEEP command. The master password holder can then authorize the DBA password holder to issue other CONTROL language commands.

A DBA password holder cannot issue any commands that access data or any DEFINE language commands. Because the DBA password holder cannot access any data, he cannot issue a RELOAD command with a where-clause, and he is not affected by Security by Entry.

To authorize the DBA password holder to issue certain CONTROL language commands, the master password holder issues the ENABLE DBA FOR command. See **Controlling DBA Password Commands: ENABLE/DISABLE DBA FOR** on page 7-15.

Examples

```
DBA PASSWORD IS CAT:

DBA PASSWORD IS ?:

DBA PASSWORD IS FINC:

DBA PASSWORD IS A5:
```

CONTROLLING DBA PASSWORD COMMANDS: ENABLE/DISABLE DBA FOR

The ENABLE/DISABLE DBA FOR command controls the DBA password holder's authorization for specific CONTROL language commands. The DBA password holder has automatic authority to issue all system-wide commands and the KEEP command, and authority for these cannot be disabled. However, none of the administrative commands listed below can be issued by the DBA password holder unless they have been specifically enabled.

Only the master password holder can issue the ENABLE/DISABLE DBA FOR command.

Format

```
|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
```

The ENABLE DBA FOR command authorizes the DBA password holder to issue the commands, and the DISABLE DBA FOR command removes that authorization. The ALL COMMANDS option (with ENABLE) gives the DBA password holder the authority to issue all the commands listed above. When the same syntax is used with the DISABLE DBA FOR command, all previously assigned authority is removed.

If the master password holder issues the ENABLE/DISABLE DBA FOR command and a DBA password does not exist, SYSTEM 2000 software issues an error message.

Examples

```
ENABLE DBA FOR ALL COMMANDS:

DISABLE DBA FOR VALUES PADDING:

DISABLE DBA FOR FULL PASSES:
```

INVALIDATING PASSWORDS: INVALID PASSWORD IS

The INVALID PASSWORD IS command removes a secondary password or the DBA password from the database.

Only the master password holder can issue the INVALID PASSWORD IS command.

Format

```
INVALID PASSWORD IS password :
```

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

When you issue the INVALID PASSWORD IS command to remove a secondary password, the command also removes all authorities assigned to the password. Also, when you remove a DBA password, the commands that have been enabled are automatically disabled. Only one password can be invalidated at a time.

The INVALID PASSWORD IS command cannot invalidate the master password.

Examples

```
INVALID PASSWORD IS ABLE:

INVALID PASSWORD IS +:

INVALID PASSWORD IS 994:
```

CHANGING PASSWORDS: CHANGE PASSWORD

The CHANGE PASSWORD command changes the master password, secondary passwords, and the DBA password.

Only the master password holder can issue the CHANGE PASSWORD command.

Format

```
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 the SEPARATOR IS command in *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for the list of special characters.

Any password for a specific database can be changed. All authorities associated with an old secondary password are transferred to the new secondary password. All commands that are authorized for an old DBA password are transferred to the new DBA password.

The new password cannot be the same as the current master password, DBA password, or any secondary password associated with the database. You can change only one password at a time.

Examples

```
CHANGE PASSWORD ABLE TO BAK:

CHANGE PASSWORD CHAR TO +:

CHANGE PASSWORD $ TO 994:
```

LISTING PASSWORDS: LIST PASSWORDS

The LIST PASSWORDS command displays all valid secondary passwords.

Only the master password holder can issue the LIST PASSWORDS command.

Format

```
LIST PASSWORDS  :
```

Secondary passwords appear in the order they were assigned, and they are listed in a left-aligned column. The output from the LIST PASSWORDS command is written on the Message File. The master password and the DBA password are not listed.

Example

```
LIST PASSWORDS:
        CHAR
        CAT
        +
        ABLE
        BAK
        DOG
        994
```

Abbreviations:

LIST = LI

PASSWORDS = PASSWORD

LISTING SECONDARY PASSWORDS AND AUTHORITIES: LIST PASSWORDS AND AUTHORITIES

The LIST PASSWORDS AND AUTHORITIES command displays all secondary passwords and their authorities for each component.

Only the master password holder can issue the LIST PASSWORDS AND AUTHORITIES command.

Format

```
LIST PASSWORDS AND AUTHORITIES  :
```

The output from this command is written on the Message File. The list of secondary passwords and authorities is tabular with passwords as column headings and components as row headings. The master password and the DBA password are not listed.

The password columns appear left-to-right in the order in which the passwords were assigned. The component rows are in DESCRIBE order. A three-character authority symbol appears at the intersection of each row and column. R, U, and W represent authorities assigned to that password; a period represents authorities not assigned. When periods are present for all authorities, N is the assigned authority.

Example

```
LIST PASSWORDS AND AUTHORITIES:
       CHAR      CAT       +         ABLE
    0* R..       ..W       R..       RUW
    1* R..       .U.       R..       RUW
    2* RU.       ...       R..       RUW
    3* RUW       RUW       RUW       RUW
```

Abbreviations:

AUTHORITIES = AUTH
LIST = LI
PASSWORDS = PASSWORD

LISTING THE DBA PASSWORD: LIST DBA

The LIST DBA command displays the DBA password.

Only the master password holder can issue the LIST DBA command.

Format

```
LIST DBA  :
```

The output from the LIST DBA command is written to the Message File.

If the master password holder issues the LIST DBA command and a DBA password does not exist, SYSTEM 2000 software issues an error message.

Example

```
LIST DBA:
        CAT
```

Abbreviation: LIST = LI

LISTING THE DBA PASSWORD AND COMMANDS: LIST DBA AND COMMANDS

The LIST DBA AND COMMANDS command lists the DBA password and the commands the DBA password has been authorized to issue.

Only the master password holder can issue the LIST DBA AND COMMANDS command.

Format

```
LIST DBA AND COMMANDS  :
```

The output from LIST DBA AND COMMANDS is written to the Message File. It includes the DBA password and all commands the DBA password holder has been specifically authorized to issue by the master password holder. The output does not include the system-wide commands or the KEEP command.

Some commands are abbreviated on the output, as shown below.

Command	Abbreviation
ALLOW/NO FULL PASSES	FULL PASSES
ENABLE/DISABLE MULTIPLE HOLDS	MULTIPLE HOLDS
ENABLE/DISABLE ROLLBACK	ROLLBACK
ENABLE/DISABLE VALUES PADDING	VALUES PADDING
RESET ROLLBACK AFTER	RESET ROLLBACK
SAVE DATA BASE	SAVE

Example

```
LIST DBA AND COMMANDS:
        CAT
PRINT SIZE
RELOAD
REORGANIZE
SAVE
RELEASE
RESTORE
APPLY
SUSPEND
ROLLBACK
RESET ROLLBACK
```

Abbreviation: LIST = LI

Chapter 8

Controlling Non-Key Where-Clause Processing: ALLOW/NO FULL PASSES

The ALLOW/NO FULL PASSES command allows or prohibits complete passes through the Data Table for specified databases.

Only the master password holder or a DBA password holder with authorization for this command can issue the ALLOW/NO FULL PASSES command.

Format

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

SYSTEM 2000 where-clauses require a complete pass through the Data Table when non-key conditions are specified. This processing can be very expensive, especially for large databases. This command allows or prohibits the execution of commands containing where-clauses that require a complete pass through the Data Table.

When you specify whether full passes are allowed or prohibited, the specification is stored in the database. It remains in effect until another ALLOW/NO FULL PASSES command is issued for that database. Because the full passes specification is stored in the database, it affects all users within a Multi-User environment.

See Chapter 7, "Ensuring Database Security" for the effect of Security by Entry on the ALLOW FULL PASSES command.

If NO FULL PASSES is in effect, the software rejects any command requiring full passes and issues an appropriate message.

Example

```
NO FULL PASSES:
```

Chapter 9

Obtaining DataBase Size Statistics: PRINT SIZE

The PRINT SIZE command obtains statistical information about the size of a database.

Only the master password holder or a DBA password holder with authorization for this command can issue the PRINT SIZE command.

Format

```
PRINT SIZE  :
```

The PRINT SIZE command enables you to determine whether the disk space allocated for the six database files is adequate for future increases in the amount of data to be stored. PRINT SIZE output provides statistical information about the following database characteristics:

- database name
- definition number
- database cycle number
- date and time of the most recent update (or modification) of the database
- database pages in use and unused for each of the six required database files and the Update Log and Rollback Log (if enabled)
- page size for each of the required database files and the Update Log and Rollback Log (if enabled)
- volume serial number and number of extents used for each required file and the Update Log and Rollback Log (if enabled)
- total number of characters in the six required database files.

By using the size statistics in conjunction with the growth rate of the various database files, you can increase the disk allocations periodically. Also, if the Update Log (File 7) and the Rollback Log (File 8) are active, the size statistics are given for these files.

Abbreviation: PRINT = PR

If the Update Log is not enabled but the Rollback Log is, the File 7 display line is present. If the Rollback Log is disabled, no display is given for File 8. If the Rollback Log is enabled, then PAGES IN USE and PAGES UNUSED are tracks in use and unused, because variable length records are written to the Rollback Log. PAGES IN USE is zero if the file is empty. The two count numbers are followed by TRKS as a reminder that the count reflects tracks. VARIABLE is printed in the PAGE SIZE column where a number would be meaningless.

Compare the database size statistics to the present disk space allocations for the database. You can then adjust space allocations as needed to accommodate future updates to the database.

Example

```
CONTROL:  PRINT SIZE:
                EMPLOYEE    3    1   01/03/1991   15:08:52
             PAGES      PAGES       PAGE
           IN USE     UNUSED       SIZE
EMPLOYE1   -  8          2          2492
  VOL SER -  STOR04  EXTENTS- 1
EMPLOYE2   -  25         0          2492
  VOL SER -  STOR04  EXTENTS- 1
EMPLOYE3   -  2          3          2492
  VOL SER -  STOR04  EXTENTS- 1
EMPLOYE4   -  5          0          2492
  VOL SER -  STOR04  EXTENTS- 1
EMPLOYE5   -  5          0          2492
  VOL SER -  STOR04  EXTENTS- 1
EMPLOYE6   -  10         10         2492
  VOL SER -  STOR04  EXTENTS- 1
EMPLOYE7   -  1          59         6440
  VOL SER -  STOR08  EXTENTS- 1
EMPLOYE8   -  0          210        2504
  VOL SER -  STOR06  EXTENTS- 1
TOTAL NUMBER OF CHARACTERS = 137060
```

Chapter 10

Controlling User Exits: ENABLE/DISABLE ROUTINE

The ENABLE/DISABLE ROUTINE command specifies which user exits are to be enabled or disabled.

Format

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

names is a list of 1 to 10 routine names, separated by commas. Each name consists of 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. (A single-digit integer can be specified 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. (A single-digit integer can be specified with or without the leading 0.)

If the user-exit feature is installed, any available user exit can be enabled or disabled, except for these:

- EXIT00 must always remain enabled.
- EXIT02 occurs only when Multi-User is initialized.
- EXIT01 must always remain enabled.
- EXIT50 through EXIT63 can be enabled or disabled, but the request is not honored.

The installation systems programmer is responsible for supplying and supporting the code for any user exit to be enabled.

The user-supplied code for an enabled user exit examines the command parameters and determines if the specified request will be done. For more details about user exits, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition.*

Examples

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

DISABLE ROUTINE ALL FOR EXITS 17 THRU 20:

ENABLE ROUTINE DEMO, JTT FOR EXITS 8, 22:
```

Chapter 11

Controlling Multiple Local Holds: ENABLE/DISABLE MULTIPLE HOLDS

The ENABLE/DISABLE MULTIPLE HOLDS command allows PLEX users to acquire multiple local holds simultaneously on a database in a Multi-User environment.

Only the master password holder or a DBA password holder with authorization for this command can issue the ENABLE/DISABLE MULTIPLE HOLDS command. You must also have exclusive use of the database to issue the ENABLE/DISABLE MULTIPLE HOLDS command.

Format

```
ENABLE MULTIPLE HOLDS  :
DISABLE MULTIPLE HOLDS  :
```

The ENABLE MULTIPLE HOLDS command sets a flag in database File 1 that allows PLEX users to acquire multiple local holds on a database simultaneously. The DISABLE MULTIPLE HOLDS command turns off the flag in database File 1 and releases the Multiple Local Holds buffer. Then the user can acquire only one hold at a time; any previous holds are dropped, and the latest one acquired is retained.

Multiple holds are automatically enabled if the Rollback Log is enabled. If you disable the Rollback Log, the MULTIPLE HOLDS option reverts to its previous setting.

Considerations

- If a PLEX user requests a local hold and no space is available in the Multiple Local Holds (MLH) buffer, SYSTEM 2000 software suspends that user unless a deadlock occurs. A potential for deadlock exists if either of the following situations occur:
  - A user requests a local hold and no space is left in the buffer.
  - A user has a hold on a record and requests a local hold on a record currently held by another user. At the same time, a third user is waiting to place a hold on the record the first user is holding.

A non-zero return code is issued if a deadlock condition occurs, and all local holds are dropped for the user requesting the local hold.

- Each database uses only one buffer to track local holds. Therefore, application programs running under Multi-User with multiple holds enabled might receive return code 110 if too many local holds are required on a database at any one time. The location and size of the Multiple Local Holds buffer depends on which operating system you are running under. For details, see the *SYSTEM 2000 Product Support Manual, Version 12, First Edition*.

  If the Rollback Log is enabled, it automatically enables multiple holds. This may be a problem for large REMOVE TREE operations with the Rollback Log enabled, because every record removed in the tree requires a local hold. If the Rollback Log is enabled, User B will attempt to hold the level 1 record and will wait until User A drops the hold. To avoid return code 110 in this case or similar cases, run these high-volume programs with exclusive use or with a global hold (QUEUE/TERMINATE or FRAME/END FRAME).

Examples

```
ENABLE MULTIPLE HOLDS:
```

```
DISABLE MULTIPLE HOLDS:
```

Appendix A

CONTROL Language Commands Under Multi-User™ Software

This appendix shows how various CONTROL language commands affect or are affected by Multi-User software. Some commands require exclusive use of the database. Others cause local or global holds for a short time when the database is under non-exclusive use. The chart also shows how long the holds last and what happens if a hold cannot be honored.

PLEX Command	Requirements for Honoring Command	Duration	Status If Command Not Honored
OPEN/LOCK *dbn*. Requests exclusive use of the data base by a single PLEX program. Also, /EXCLUSIVE and /EXCL are synonymous with /LOCK.	1. No other SCF or PLEX user has the database open, and 2. No other databases are currently open in the PLEX program.	Exclusive use until: 1. CLOSE *dbn* 2. End of PLEX program 3. Job abort	PLEX program goes into a WAIT state until database has been relinquished by all other users. Other users can open the database both before and after the OPEN/LOCK was issued (but not yet honored). See also Summary of PLEX Return Codes.
OPENR/LOCK *dbn*. Same as OPEN/LOCK *dbn* or OPEN/EXCLUSIVE *dbn*.	Same as for OPEN/LOCK *dbn* or OPEN/EXCLUSIVE *dbn*.	Same as for OPEN/LOCK *dbn* or OPEN/EXCLUSIVE *dbn*.	Non-zero return code is sent back to the PLEX program. See Summary of PLEX Return Codes.
RESTORE *dbn* FROM tape. Loads the database from tape. SYSTEM 2000 software automatically causes it to be under exclusive use.	No other databases are currently open in the PLEX program. A specific CLOSE command must be used to close each of the other databases.	Same as for OPEN/LOCK *dbn* or OPEN/EXCLUSIVE *dbn*.	Return code if any other database is open in the PLEX program. (RESTORE verb is synonymous for SCF and PLEX.)
APPLY or PLEX LOAD *dbn*	Database must be opened for exclusive use with /LOCK or /EXCLUSIVE or created or restored from tape in some run unit.	Same as for OPEN/LOCK *dbn* or OPEN/EXCLUSIVE *dbn*.	Return code if database is not under exclusive use.
RELEASE	Same as for APPLY.	If RELEASE command completes processing, the database has been released.	Same as for APPLY.
OPEN *dbn*. Requests non-exclusive use of the database. Allows multiple SCF or PLEX users to retrieve or update with automatic inter-leaving of requests by Multi-User Scheduler.	Database is not under exclusive use by any SCF or PLEX user. Also, if the requesting PLEX program had the database open with exclusive use, a CLOSE command must be given before the OPEN *dbn* command.	Non-exclusive use until: 1. CLOSE *dbn* 2. End of PLEX program 3. Job abort 4. If *dbn* has multiple users, all must have relinquished access to the database.	PLEX program goes into a WAIT state until database has been relinquished from exclusive use. See also Summary of Return Codes.

(*continued on next page*)

(*continued*)

PLEX Command	Requirements for Honoring Command	Duration	Status If Command Not Honored
OPENR *dbn* (same as OPEN *dbn*)	Same as for OPEN *dbn*	Same as for OPEN *dbn*	Non-zero return code is sent back to the PLEX program. See Summary of Return Codes.
GET1/HOLD GET/HOLD GETA/HOLD GETD/HOLD GETP/HOLD (prevents database from being updated, but allows retrievals)	Database is not under the influence of any update processing or a PLEX HOLD. The HOLD option is meaningless under exclusive use.	1. End of program 2. Job abort 3. PLEX IMMEDIATE mode: The HOLD option lasts until a synchpoint occurs. Other requests are not honored until the hold is dropped, i.e., until a synchpoint occurs when Rollback is enabled. 4. PLEX QUEUE mode: A HOLD is established by the first HOLD command issued after the QUEUE command. It is relinquished by: a. The TERMINATE command. The TERMINATE command then causes the database to be updated. Other requests are not honored until the update completes. b. A "get" command is issued without the HOLD option. c. A HOLD is issued for another database.	PLEX program goes into a WAIT state if there is a global hold on the database or a local hold on the requested record(s).

Index

A

accessing SYSTEM 2000 software 2-2
ALLOC command 2-8
ALLOC execution parameter 2-9
allocating dynamically
    database files 2-8
    the Keepfile 5-5, 5-15, 5-18
    the Savefile 5-5, 5-15
ALLOW/NO FULL PASSES command 8-1
APPLY command 5-22
applying logged updates 5-22
ASSIGN authorities command 7-7
assigning
    DBA password 7-14
    master password 7-3
    secondary password authorities 7-7
    secondary passwords 7-6
    Security by Entry 7-11

C

CHANGE PASSWORD command 7-17
changing
    an item from key to non-key 4-4
    an item from non-key to key 4-2
    master, secondary, and DBA passwords 7-17
    secondary password authorities 7-7
CONTROL language commands
    ALLOC 2-8
    ALLOW/NO FULL PASSES 8-1
    APPLY 5-22
    ASSIGN authorities 7-7
    CHANGE PASSWORD 7-17
    CREATE INDEX 4-2
    DATA BASE NAME IS 2-6
    DBA PASSWORD IS 7-14
    ENABLE/DISABLE DBA FOR 7-15
    ENABLE/DISABLE MULTIPLE HOLDS 11-1
    ENABLE/DISABLE ROLLBACK 6-11
    ENABLE/DISABLE ROUTINE 10-1
    ENTRY KEY IS 7-11
    FREE 2-13
    INVALID PASSWORD IS 7-16
    KEEP 5-18
    LIST DBA 7-20
    LIST DBA AND COMMANDS 7-21
    LIST PASSWORDS 7-18
    LIST PASSWORDS AND AUTHORITIES 7-19
    NEW DATA BASE IS 2-3
    overview 1-2

PRINT SIZE 9-1
RELEASE 5-11
RELOAD 3-4
REMOVE INDEX 4-4
REORGANIZE 3-1
RESET ROLLBACK AFTER 6-13
RESTORE 5-12
SAVE 5-4
SUSPEND 5-26
under Multi-User environment A-1
USER 2-2, 7-3
VALID PASSWORD IS 7-6
controlling
DBA password commands 7-15
multiple holds 11-1
non-key where-clause processing 8-1
Rollback Log 6-11
user exits 10-1
Coordinated Recovery 6-1
CREATE INDEX command 4-2
creating indexes 4-1
cycle number, database xii

D

DATA BASE NAME IS command 2-6
database
allocating files dynamically for 2-8
allocating files for 2-1
applying logged updates to 5-22
creating 2-1, 2-3
creating and removing indexes for 4-1
deallocating files dynamically for 2-13
dynamic file allocation for 2-8
dynamic file deallocation for 2-13
indexes, creating and removing 4-1
naming 2-3
opening 2-1
opening existing files for 2-6
passwords 7-1
recovering with Coordinated Recovery 6-1
releasing 5-11
reloading 3-4
reorganizing 3-1
restoring 5-12
saving 5-4
security of 7-1
size statistics for, obtaining 9-1
S2KDBCNT allocation table 2-9
transaction and system failure 6-4
transferring updates for 5-18
database cycle number xii
DBA password
assigning 7-14
changing 7-17

controlling commands for 7-15
invalidating 7-16
listing 7-20
DBA password and commands, listing 7-21
DBA PASSWORD IS command 7-14
deallocating database files dynamically 2-13
dynamic allocation of
database files 2-8
the Keepfile 5-5, 5-15, 5-18
the Savefile 5-5, 5-15

E
ENABLE/DISABLE DBA FOR command 7-15
ENABLE/DISABLE MULTIPLE HOLDS command 11-1
ENABLE/DISABLE ROLLBACK command 6-11
ENABLE/DISABLE ROUTINE command 10-1
ENTRY KEY IS command 7-11

F
FREE command 2-13
FULL PASSES *See* ALLOW/NO FULL PASSES command

I
INVALID PASSWORD IS command 7-16
invalidating secondary and DBA passwords 7-16

J
JCL for
restoring a database 5-15
saving a database 5-7

K
KEEP command 5-18
Keepfile, dynamically allocating 5-5, 5-15, 5-18

L
LIST DBA AND COMMANDS command 7-21
LIST DBA command 7-20
LIST PASSWORDS AND AUTHORITIES command 7-19
LIST PASSWORDS command 7-18
listing
DBA password 7-20
DBA password and commands 7-21
secondary passwords 7-18
secondary passwords and authorities 7-19
logical units of work 6-7

M
master password
assigning 7-3
changing 7-17
Multi-User, CONTROL language commands A-1

multiple holds, controlling 11-1

N
naming a database 2-3
NEW DATA BASE IS command 2-3
non-key where-clause processing, controlling 8-1

O
overview of
    CONTROL language commands 1-2
    SYSTEM 2000 software 1-1

P
passwords 7-1
PRINT SIZE command 9-1

R
reconstructing an active database 3-4
recovering databases
    with Coordinated Recovery 6-1
    with the APPLY command 5-22
RELEASE command 5-11
releasing a database 5-11
RELOAD command 3-4
REMOVE INDEX command 4-4
removing indexes 4-1
REORGANIZE command 3-1
reorganizing a database 3-1
RESET ROLLBACK AFTER command 6-13
RESTORE command 5-12
restoring a database
    overview 5-1
    RESTORE command 5-12
Rollback Log
    controlling 6-11
    description of 6-5
    resetting 6-13

S
save and restore process, examples of 5-28
SAVE command 5-4
Savefile, dynamically allocating 5-5, 5-15
saving a database 5-1, 5-4
    with the Update Log 5-3
    without the Update Log 5-2
secondary password authorities
    assigning and changing 7-7
    listing 7-19
secondary passwords
    assigning 7-6
    changing 7-17
    invalidating 7-16
    listing 7-18

security 7-1
Security by Entry, assigning 7-11
statistics for database size, obtaining 9-1
SUSPEND command 5-26
suspending Update Log 5-26
synchpoints 6-7
system failure 6-4
SYSTEM 2000 software
    accessing 2-2
    introduction 1-1
S2KDBCNT allocation table 2-9

T
THROUGH as part of a component name 7-9
transaction failure 6-4
transferring updates 5-18

U
Update Log 6-6
    controlling 5-4, 5-12
    for Coordinated Recovery 6-6
    suspending 5-26
USER command 2-2, 7-3
user exits, controlling 10-1
user synchpoints 6-7

V
VALID PASSWORD IS command 7-6
varying databases offline 2-9

Your Turn

If you have comments or suggestions about SYSTEM 2000 software or *SYSTEM 2000® CONTROL Language, 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.