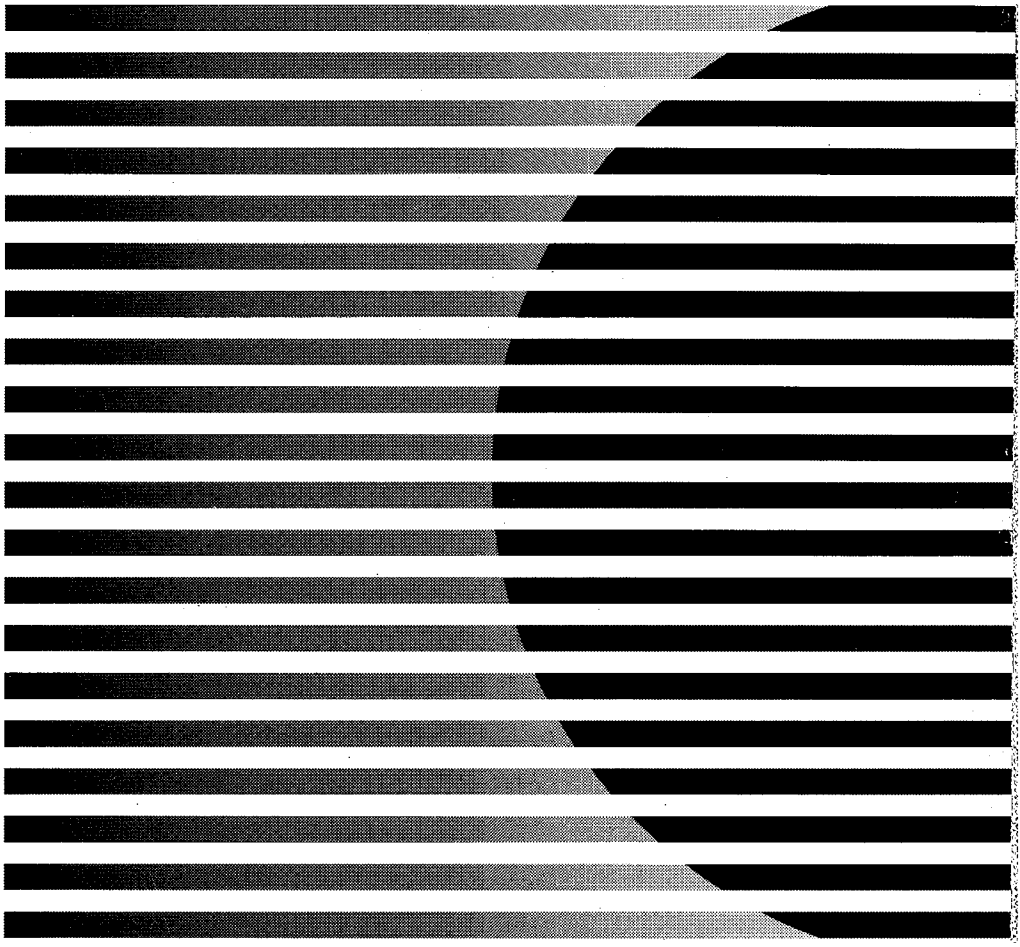
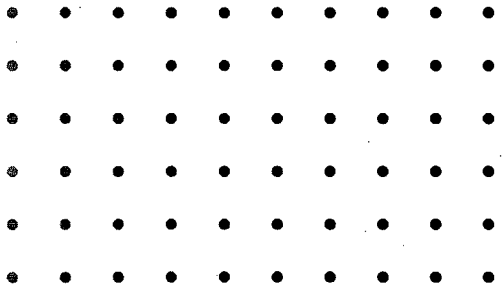
SYSTEM 2000® Data Management Software

Product Support Manual

Version 12
First Edition



55016



SAS® SAS Institute Inc.

SYSTEM 2000®
Product Support Manual

Version 12
First Edition

SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. *SYSTEM 2000® Product Support Manual, Version 12, First Edition*. Cary, NC: SAS Institute Inc., 1991. 400 pp.

SYSTEM 2000® Product Support Manual, Version 12, First Edition

ISBN 1-55544-183-1

1st printing, August 1991

Contents

Illustrations

Using This Book

Purpose

This manual describes the various support functions and tools available for gathering system statistics and analyzing your SYSTEM 2000 configuration. It tells how to use the utilities and macros on the delivery tape and provides examples. The delivery tape includes programs for displaying timing and performance statistics, macros for configuring SYSTEM 2000 software at your site, and a framework for implementing user exits. The manual describes all files associated with SYSTEM 2000 software and discusses allocating database files and work files, initializing Multi-User software, specifying execution parameters, and executing user jobs.

The material in this manual pertains to running SYSTEM 2000 software on IBM mainframes (series 370 or higher, and equivalents) under MVS and CMS environments. For additional information about using SYSTEM 2000 software under CMS, see the *SYSTEM 2000 CMS Supplement, Version 12, First Edition.*

Organization of This Book

This book has reference chapters and appendices, as listed below.

Reference The reference chapters discuss various options you can set for single-user and Multi-User environments, allocating files, and configuring SYSTEM 2000 software.

Chapter 1, "Introduction"

Chapter 2, "SYSTEM 2000 Single-User Environment"

Chapter 3, "SYSTEM 2000 Multi-User Environment"

Chapter 4, "Specifying Files and Buffers"

Chapter 5, "Selecting and Setting SYSTEM 2000 Options"

Chapter 6, "User Exits"

Chapter 7, "Configuring SYSTEM 2000 Software: PRELNK Macro"

Appendices The appendices provide information on various special areas.

Appendix A, "Supplementary PLEX Information"

Appendix B, "Converting Databases to Version 12 Format"

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 notation:

UPPERCASE ROMAN	indicates keywords. They must be spelled exactly as given in the syntax. 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 and options.

Examples are shown in uppercase. However, you can enter your own SYSTEM 2000 commands in lowercase, uppercase, or a mixture of both. Comments appear in italics and within parentheses.

Page Numbering Convention

Pages are numbered sequentially within the chapter number, that is, 6-1, 6-2, and so on. The page numbering convention is reflected in the index. For example, the page number 6-5 means that the reference information starts on the fifth page of Chapter 6.

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

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

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

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

- *SYSTEM 2000 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 Product Support Manual for Version 12. This information is intended for users who have previous experience with SYSTEM 2000 software.

Mixed Case Data

Version 12 accepts and recognizes SYSTEM 2000 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.

Converting Existing Databases to Version 12 Format

All databases created with earlier releases of SYSTEM 2000 software must be rebuilt in the Version 12 format. The purpose and function of each database table has not changed. However, the internal address pointers are now longer to accommodate more data records.

You can "unload" your database definition and the data, then build a Version 12 definition and use PLEX optimized load mode to build the database tables. Or you can load the database with the SCF LOAD command.

Also, the Version 12 delivery tape contains programs that offer two methods of rebuilding your databases. For details about converting your existing databases, see Appendix B.

Dynamically Allocating the SYSTEM 2000 Files

SYSTEM 2000 software will dynamically allocate database files, Savefiles, and Keepfiles if they are not already allocated in the JCL or CLIST. You can use the defaults or issue the CONTROL language ALLOC command with options to change the defaults for database files.

The new ALLOC, FREE, and PREFIX execution parameters control dynamic allocation and deallocation of database files. Also, an S2KDBCNT table is available to control which databases can be allocated dynamically.

The Savefile is now a QSAM file. Its DDname is the first seven characters of the database name, followed by a suffix of S. (TAPES2K is no longer honored for a Savefile.) The Keepfile can also be dynamically allocated, if necessary, with a suffix of K.

Dynamic allocation of the work files is even easier. Most jobs can use the default attributes and temporary data sets. Dynamic allocation does not create any permanent data sets for work files. The software deallocates the temporary work files at the end of the SYSTEM 2000 session. Files that were not dynamically allocated are not deallocated.

You might want to make the S2KPAD00 file permanent in order to save initial formatting time. Also, you might want to create a permanent S2KPARMS file with the default data set name to allow for more buffers.

In summary, dynamic allocation is available for database files, Savefiles, Keepfiles, user files, scratch pads, sort files, the S2KPARMS file, the S2KUSERS file, the S2KSNAP file, and the S2KDBCNT file. For more information about allocating these files, see Chapter 4, "Specifying Files and Buffers." See also *SYSTEM 2000 CONTROL Language, Version 12, First Edition* for information about the NEW DATA BASE IS, DATA BASE NAME IS, SAVE, RESTORE, ALLOC, and FREE commands.

31-Bit Mode

Version 12, when run under MVS/XA™, MVS/ESA™, VM/XA™, or VM/ESA™, exploits the 31-bit address capabilities and provides virtual storage constraint relief by moving most program and data storage above the 16-megabyte line.

Even though your existing PLEX programs are probably 24-bit mode, they will work with Version 12 in single-user jobs or in the Multi-User environment without changes. All communications between your PLEX program and SYSTEM 2000 software are via S2KPLR, which ensures the correct addressing mode. Although it is not necessary, you can take advantage of 31-bit addressing by having the SYSTEM 2000 control blocks within your PLEX program reside above the 16-megabyte line, along with the rest of your code. This situation allows you to have more memory to define your data areas and in-memory tables.

31-bit addressing mode could affect your user-exit code. Your user-exit program is entered in the same addressing mode as SYS2K or S2KPLI. In 31-bit mode, do not clutter the high-order byte of a register or word that contains an address (for example, with flags) since all 31 bits are used as an address. Also, the addresses passed to your user-exit program are likely to be above the 16-megabyte line, and you must be in 31-bit mode to reference them.

Increased Database Components and File Sizes

Version 12 allows a significant increase in the number of data records in a database, the capacity of the data, and the number of components in a database definition. In general, the maximum record count is approximately seven times greater than earlier versions. The byte count for data is approximately three times greater, and the number of components allowed is ten times greater.

You can now have 10,000 components in a database definition, and a database can have up to 119,304,647 data records. The byte count for the Data Table also increased substantially; the capacity depends on the disk type and block size. The index tables have not changed.

The maximum file sizes for a single database are as follows:

File 1 10,000 components and 200 secondary passwords

File 2 536,870,912 bytes (unchanged)

File 3 268,435,455 characters (unchanged)

File 4 357,195,375 maximum number of multiple occurrence pointers (unchanged)

File 5 119,304,647 data records. (Earlier versions allowed 16,777,215 data records.) There is one Hierarchical Table entry for each data record. The entries are fixed length, and the entry size has grown from 14 to 18 bytes.

File 6 The Data Table contains the actual data for a database, collected by schema record type and structured by pointers in the Hierarchical Table (File 5). The maximum number of tracks that can be used for database File 6 is 65,535 (X'FFFF'). Thus, the maximum number of bytes for File 6 depends on the block size and disk device. For example, a 3380 disk using a block size of 23,464 (two blocks per track) could use up to 131,071 blocks or 3,075,426,480 bytes (non-numeric characters). Earlier versions allowed 1,073,741,823 bytes.

Maximum Database Cycle Number

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

User Exits

Several new user exits are included in Version 12. For details, see Chapter 6, "User Exits." Also, see **31-Bit Mode under XA Systems** on page 6-15 concerning effects of 31-bit mode on your user exits.

Execution Parameters

Several new SYSTEM 2000 execution parameters are available in Version 12, including those for dynamic allocation of database files, scratch pads, and sort files. See Chapter 5, "Selecting and Setting SYSTEM 2000 Options," for a summary of all execution parameters.

Chapter 1

Introduction

SYSTEM 2000 software allows you to define new databases and to modify the definition of existing databases, as well as to store, retrieve, and update values in these databases. SYSTEM 2000 software allows the organizational structure of the database to represent information about the data. The designer of a SYSTEM 2000 database determines what relationships are relevant and how data will be represented.

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

SCF offers five basic languages for defining, accessing, and controlling databases. You can call any of the following languages during a SYSTEM 2000 session:

DEFINE	The DEFINE language allows you to define and redefine a SYSTEM 2000 database.
CONTROL	The CONTROL language allows you to perform 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 for more efficient processing of large runs.
REPORT	The REPORT language allows you to produce reports with running subtotals, special formats, and logical pages within the physical page.

The PLEX 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, both of which access SYSTEM 2000 databases. QueX provides menus for viewing, modifying, or entering data. Genius is a conversational system. It prompts you for report specifications, which will display database information in columnar format.

This manual discusses the many system support functions and tools available for configuring and running SYSTEM 2000 software at your site. These include multiple user access, coordinated recovery of multiple databases, accounting information, performance statistics, and overlay configurations. You can use the default settings for these options or tailor them to meet your site requirements.

Chapter 2, SYSTEM 2000 Single-User Environment - discusses executing batch SCF and PLEX jobs when each job is using a separate copy of SYSTEM 2000 software. JCL and execution parameters are described.

Chapter 3, SYSTEM 2000 Multi-User Environment - gives an overview of the Multi-User configuration, which allows multiple SCF and PLEX users to run batch jobs concurrently. Also, if SCF Teleprocessing (SCF TP) is activated, multiple SCF users can access SYSTEM 2000 software interactively. This chapter covers using Multi-User with IBM Cross Memory Services (XMS), as well as generating the Type 2 SVC for Multi-User, and shows how to initialize and terminate a Multi-User session. It also discusses the execution parameters and console operator commands available, along with tools for gathering resource statistics and event data during Multi-User sessions. (For details, see the Accounting Log and Diagnostic Log discussions.)

Chapter 4, Determining File and Buffer Sizes - describes the contents and usage of all files associated with SYSTEM 2000 software. Database files, work files, scratch files, and other optional files used with the SCF TP facility, the Accounting Log, and the Diagnostic Log are discussed. Chapter 4 gives details on allocating and naming the files and describes the file formats. Examples illustrate how to determine page sizes and I/O buffer pools for the files. This chapter also presents the XBUF caching feature.

Chapter 5, Selecting and Setting SYSTEM 2000 Options - shows various methods for setting SYSTEM 2000 execution parameters. A summary shows the purpose of each parameter and refers you to pertinent information about the parameter.

Chapter 6, User Exits - describes how to use the optional user exits while executing SYSTEM 2000 jobs. Each exit is detailed. This chapter also suggests a method for implementing user exits; it describes the macros supplied and provides a sample application.

Chapter 7, Configuring SYSTEM 2000 Software: PRELNK Macro - shows you how to configure the SYSTEM 2000 modules. The default system is flat (non-overlayed), which suits most sites. It requires maximum memory but no I/O for swapping overlays. If you want to configure an overlayed system, read about the PRELNK, GROUP, and ENDLNK macros. A configuration that is fully or partially overlayed uses less memory, but it involves I/O for overlay swapping during the execution of SYSTEM 2000 jobs.

Appendix A, Supplementary PLEX Information - contains tables showing the formats and contents of the PLEX S2KDUM, COMMBLOCK, and SUBSCHEMA control blocks. Also, internal operation codes are listed for all PLEX commands. The last chapter discusses how to use the optional execution-time PLEX exits.

Appendix B, Converting Databases to Version 12 Format - discusses methods for rebuilding databases in the Version 12 format. Internal address pointers are longer to accommodate more data records and larger databases.

Chapter 2

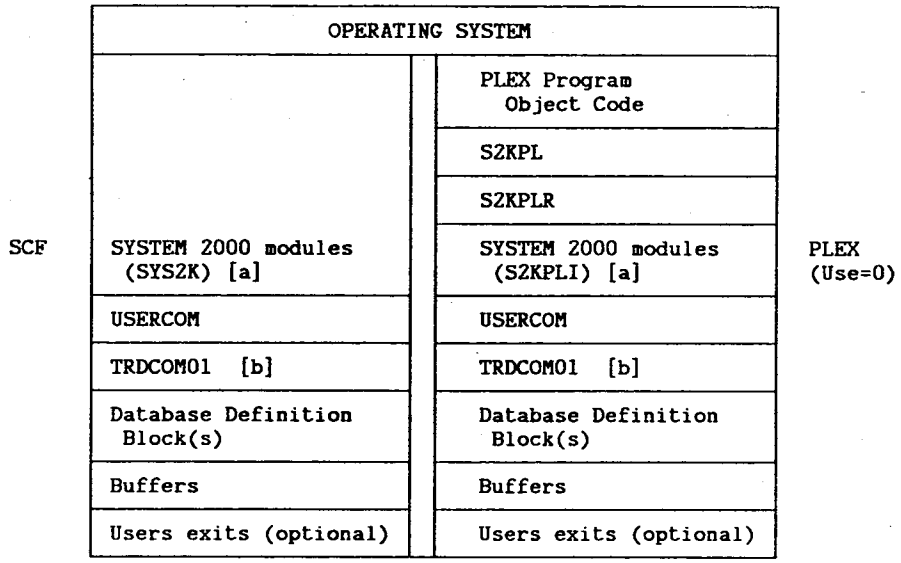
SYSTEM 2000 Single-User Environment

THE SINGLE-USER ENVIRONMENT: OVERVIEW

The single-user version of SYSTEM 2000 software provides basic database management capabilities and an efficient means of implementing large batch updates.

In a single-user environment, each SCF batch job, PLEX batch job, and time-sharing (TSO) session (either SCF or PLEX) executes a distinct copy of the software. Illus. 2.1 on page 2-2 shows an SCF job and a PLEX job executing in a single-user environment.

Illus. 2.1 Single-User Memory Configuration



[a] Actual configuration determined in the running of PRELNK.

[b] Replaced with RWTCOM01 if the REPORT processor will be used (RW=YES).

EXECUTING SINGLE-USER SCF JOBS

Each single-user SCF batch job executes the SYS2K program, a link-edited module containing the required SYSTEM 2000 executable modules. For SCF and PLEX jobs, you can allocate your database files with JCL statements or by specifying the database name in the S2KDBCNT table. Within SCF sessions, you can allocate (and deallocate) them dynamically with the ALLOC and FREE commands, which are described in *SYSTEM 2000 CONTROL Language, Version 12, First Edition*. Also, if you do not allocate them anywhere, SYSTEM 2000 software will allocate them dynamically using default settings. For PLEX jobs, you can either allocate the existing database files in your JCL or let SYSTEM 2000 software look for them and automatically allocate them when you issue an OPEN command. User exits are also available for allocating and deallocating database files in special situations.

You can allocate Savefiles and Keepfiles in your JCL or with the SAVE and RESTORE commands, or you can simply let SYSTEM 2000 software allocate them automatically. The syntax and rules for using SAVE and RESTORE commands are discussed in *SYSTEM 2000 CONTROL Language, Version 12, First Edition*.

User files, such as the default Command File and Message File, can also be allocated in your JCL, or you can let SYSTEM 2000 allocate them automatically. For information on alternate user files, see *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition*.

You can specify the SYSTEM 2000 execution parameter settings for the job, or you can use the default settings. See **Single-User Execution Parameters** on page 2-5.

Scratch files, scratch pads, sort files, S2KPARMS file, and so on can be allocated in your JCL, or you can let SYSTEM 2000 software allocate them as temporary files (using default settings) within an SCF session or a PLEX job. Also, Version 12 provides several SYSTEM 2000 execution parameters to alter the default settings for sort files and S2KPADnn files. Read Chapter 4, "Specifying Files and Buffers," for various ways of allocating database files and work files.

For an SCF job, SYSTEM 2000 software closes the database files when an EXIT command or an end-of-file occurs on the Command File. Also, if you allocated the database files dynamically with the ALLOC command or let the software allocate them, you can deallocate them dynamically at any time with the FREE command in an SCF session. Savefiles and Keepfiles are automatically deallocated by the software after a SAVE, RESTORE, KEEP, or APPLY command completes processing. For PLEX jobs, the CLOSE command or STOP S2K command closes the database files.

Minimal JCL or TSO Call statements

All database files and work files can be dynamically allocated very easily with Version 12, as mentioned before. Therefore, you could use all the defaults and execute a SYSTEM 2000 job with the following minimal JCL statements:

```
//SCF      EXEC PGM=SYS2K,PARM='PREFIX=S2K'
//STEPLIB  DD  DSN=S2K.R120.LOAD,DISP=SHR
```

Or, you could use this TSO CALL statement:

```
CALL 'S2K.R120.LOAD(SYS2K)'
```

Example of specifying your own JCL

Illus. 2.2 on page 2-4 shows a sample procedure for executing a single-user SCF job where you specify your own JCL statements. These statements contain file names for an EMPLOYEE database and indicate DISP=OLD (implying that the database already exists). Database files were allocated in a previous job.

Illus. 2.2 Allocating Files with JCL Statements

```
//GO        EXEC PGM=SYS2K
//STEPLIB   DD   DSN=S2K.R120.LOAD,DISP=SHR
//S2KMSG    DD   SYSOUT=A,DCB=(BLKSIZE=1320,LRECL=132,RECFM=FBA)
//SYSUDUMP  DD   SYSOUT=A
//S2KSNAP   DD   SYSOUT=A,DCB=BLKSIZE=882
//S2KPARMS  DD   DSN=S2K.R120.TEST(S2KPARMS),DISP=SHR
//SF01      DD   UNIT=SYSDA,SPACE=(TRK,(1,1)
//SF02      DD   UNIT=SYSDA,SPACE=(TRK,(1,1)
//SF03      DD   UNIT=SYSDA,SPACE=(TRK,(1,1)
//SF04      DD   UNIT=SYSDA,SPACE=(TRK,(1,1)
//SF05      DD   UNIT=SYSDA,SPACE=(TRK,(1,1)
//SF06      DD   UNIT=SYSDA,SPACE=(TRK,(1,1)
//S2KPAD00  DD   DSN=S2KPAD00,UNIT=SYSDA,SPACE=(CYL,(2,5))
//EMPLOYE1  DD   DSN=S2K.R120.EMPLOYE1,DISP=OLD
//EMPLOYE2  DD   DSN=S2K.R120.EMPLOYE2,DISP=OLD
//EMPLOYE3  DD   DSN=S2K.R120.EMPLOYE3,DISP=OLD
//EMPLOYE4  DD   DSN=S2K.R120.EMPLOYE4,DISP=OLD
//EMPLOYE5  DD   DSN=S2K.R120.EMPLOYE5,DISP=OLD
//EMPLOYE6  DD   DSN=S2K.R120.EMPLOYE6,DISP=OLD
//EMPLOYE7  DD   DSN=S2K.R120.EMPLOYE7,DISP=OLD
//S2KCOMD   DD *
 (SCF commands)
/*
//
```

EXECUTING SINGLE-USER PLEX JOBS

The JCL for executing a single-user PLEX program is documented in the PLEX manual. S2KPL is linked with each PLEX program, and it loads the necessary SYSTEM 2000 executable modules.

The SYSTEM 2000 execution parameter settings for the job must be specified. Otherwise, default settings are in effect. See **Single-User Execution Parameters** on page 2-5.

A single-user PLEX program can interactively query terminals, operator consoles, or other devices for input. However, such execution time interaction must be programmed independent of SYSTEM 2000 facilities.

PLEX programs always use existing database files. You can allocate them with JCL or let SYSTEM 2000 software look for them dynamically. Also, you can allocate sort files, scratch pads, the S2KPARMS file, and so on in the JCL for your PLEX job, or you can let the software set up temporary work files dynamically. In addition, you can specify several execution parameters to alter the defaults for dynamically allocating sort files and scratch pads.

You need to close the database files with the PLEX CLOSE command before you terminate the PLEX program. The software marks the database as damaged if it is not closed after updating. Either a CLOSE command or a STOP S2K command forces pending inserts to be written to the database from the intermediate scratch files.

SINGLE-USER EXECUTION PARAMETERS

You determine the SYSTEM 2000 execution parameter options during run-time initialization of each SCF or PLEX job. SYSTEM 2000 software obtains these execution parameters from one of the following sources, listed in order of priority (highest first):

1. PARM subparameter of EXEC statement in the job
2. S2KPARMS data set
3. SYSTEM 2000 defaults.

For details about setting the parameters with the various methods, see **Setting SYSTEM 2000 Execution Parameters** on page 5-9.

When choosing execution parameters, consider the following factors:

- large scratch file operations, such as loading data and volume where-clause processing
- heavy database file activity
- SAVE/RESTORE operations
- definition processing
- scratch pads. which are required for Collect Files and the REPORT processor only.

User exits allow you to do some site-specific tasks. For example, you can collect statistics and modify input, output, or file allocations during a single-user SCF or PLEX job. For a description of available user exits, see Chapter 6, "User Exits."

You can create a PROC that sets the execution parameters for single-user SCF jobs. Then all users can call that PROC and supply the specific database files used in their jobs. You can also set up a general PROC for single-user PLEX jobs, but the DD statements vary from application to application, for example, the different database files and non-SYSTEM 2000 files required by your application.

Illus. 2.3 on page 2-6 lists the SYSTEM 2000 execution parameters that pertain to single-user jobs.

Illus. 2.3 Single-User Execution Parameters

Name	Purpose	Page
ALLOC	allows dynamic allocation of database files.	4-16
CONVERT	specifies that a database created under a prior version is to be converted for access by the current version. (Not used for Version 12.)	5-15
CORECOV	specifies the scope of database recovery under Coordinated Recovery.	5-15
DBBUFN	alters the number of QSAM buffers used for saving and restoring databases.	5-13
DISK	determines default block size for database files.	4-21
EXITS	directs the dynamic loading of the S2KEXIT user-exit interface.	6-13
FREE	controls the deallocation of database files.	4-17
LDBS	specifies the initial number of Database Definition Blocks used by large databases.	4-55
LDBSIZE	specifies the maximum number of components that can be contained in a large database.	4-55
LIST	displays a list of execution parameter settings.	5-13
OPT000	determines how the software checks for the system separator when unloading CHAR, TEXT, and UNDEFINED values.	5-17
OPT001	determines whether SCF commands are echoed.	5-18
OPT002	determines whether a damaged database is saved.	5-18
OPT003	enables or disables clears for QUEUE sessions.	5-18
OPT005	sets the default ZERO/ZERO SUPPRESS format option.	5-19
OPT006	sets the default NULL/NULL SUPPRESS format option.	5-19
OPT007	sets the default REPEAT/REPEAT SUPPRESS format option.	5-19
OPT008	sets the default NAME/NUMBER format option.	5-19
OPT009	determines whether database Files 2 through 6 are cleared for RELOAD and RELEASE commands.	5-20

continued on next page

Illus. 2.3 *continued*

Name	Purpose	Page
OPT010	sets a limit for updates and retrievals.	5-20
OPT011	sets the maximum number of pages that can be written to the primary scratch file for where-clause processing.	5-20
OPT012	sets the default database page size.	5-20
OPT013	changes the default date format.	5-21
OPT030	determines whether PLEX stacks and Locate Files are cleared when a logical unit of work is committed.	5-21
OPT032	sets the default for TALLY to EACH or ALL.	5-21
OPT033	sets the default mode for clearing update pages.	5-21
OPT035	determines how SYSTEM 2000 execution parameters are displayed.	5-22
OPT038	is reserved for future use.	
OPT039	determines the precision for floating point data in PLEX programs if padding is necessary.	5-23
OPT040	determines the precision for floating point data in PLEX programs if truncation is necessary.	5-23
OPT041	disables or enables the CMS FINIS command for all users.	5-24
OPT042	disables or enables condition codes in single-user JCL.	2-8
OPT043	disables or enables uppercase translation.	5-24
OPT044	disables or enables the processor prompt feature.	5-24
OPT045	disables or enables the processor lookup feature.	5-24
OPT046	sets the default KEY or NON-KEY status when the user is defining new items.	5-25
OPT047	determines whether validity checking occurs for user-specified packed decimal data.	5-25
PADnn	defines scratch pads.	4-41
PADPRI	alters the default scratch pad primary space.	4-41

continued on next page

Illus. 2.3 *continued*

Name	Purpose	Page
PADSEC	alters the default scratch pad secondary space.	4-41
PADSPACE	alters the default scratch pad space unit.	4-41
PADUNIT	alters the default scratch pad UNIT setting.	4-41
POOLn	specifies buffer pool attributes.	4-24
PREFIX	alters the default user prefix	4-17
RW	controls REPORT processor execution.	5-13
SDBS	specifies the initial number of Database Definition Blocks used by small databases.	4-56
SDBSIZE	specifies the maximum number of components that can be contained in a small database.	4-56
SFPRI	alters the default sort file primary space.	4-38
SFSEC	alters the default sort file secondary space.	4-38
SFSPACE	alters the default sort file space unit.	4-38
SFUNIT	alters the default sort file UNIT setting.	4-38
STAE	controls error trapping.	5-14
XBUF	enables or disables the XBUF caching software.	5-16
XBUFSUF	specifies a suffix that identifies a specific XBUF load module if several exist in the load library.	5-16

Setting Condition Codes: OPT042 Parameter

```
OPT042=|NO   (default)
       |YES
```

The OPT042 execution parameter allows you to enable and disable condition codes for the 'COND' parameter on JCL execute statements. YES disables the setting of the condition codes; the condition code will not be set for the next execute statement. NO (the default) means the condition codes function as usual.

DATABASE FILES OPENED WITH READ-ONLY MODE

For Version 12 and later releases, SYSTEM 2000 software opens database files with read-only mode in either of the following situations:

- DISP=SHR for database File 1
- the current password is a secondary password that has no update authority for any database component.

In all other situations, the database files are opened with the update option. (In a Multi-User environment, database files are always opened with the update option.)

Read-only mode in a single-user job establishes compatibility with RACF and other security packages. It also adds another level of protection from unintentional updates.

If an update is attempted on read-only database files, the software issues an error message or a PLEX return code.

- SCF error message -816- or PLEX return code 47 is issued if the password in effect does not have proper authority. The software usually can detect this error before detecting any I/O error.
- For DISP=SHR, any attempt to update the database causes the software to issue SYSTEM 2000 Error Code 815. The software detects this condition before it actually attempts to write on a database file, which would cause an I/O error.
- Any update attempt that was not stopped by either of the conditions described above (improper password authority or DISP=SHR) causes an I/O error on INPUT DCB, followed by a SYNAD error analysis and SYSTEM 2000 Error Code 817.

ABENDS IN SINGLE-USER JOBS

SYSTEM 2000 software formats the Extended Specify Task Abnormal Exit (ESTAE) work area when an abend occurs for a single-user job. The formatted ESTAE work area appears on the user's Message File (S2KMSG). See **Format of Diagnostic Log Records** on page 3-144 for an example of the formatted ESTAE work area. Also, see **STAE Parameter: Enabling Error Trapping** on page 5-14 for a discussion of the STAE execution parameter, ESTAEs, error trapping, and the S2KSNAP file.

For Release 11.6 and later releases, ESTAE macros replace STAE macros. SYSTEM 2000 software checks the system type in order to handle differences in XA and non-XA ESTAE macro expansion.

With ESTAEs, all abends, including XMS-related abends, are trapped in the same manner. ESTAEs also offer the following advantages:

- SYSTEM 2000 software cannot recover (that is, cannot continue execution) from X22 type abends, such as operator cancel, time limit exceeded, and so on; however, X22 type abends are trapped.

- ESTAEs percolate up to higher levels of ESTAEs for all PLEX programs. At the higher level you can use error-trapping techniques or a debugging package to further handle any error.

- If the STAE execution parameter equals NODUMP, ESTAEs are in effect but no dump is taken if an abend occurs. STAE=NODUMP is especially good for COBOL II and PL/I PLEX programs that utilize debugging aids, and, therefore, do not need a SYSTEM 2000 dump.

  ESTAEs are also used in subtasks, such as restoring or saving a database with QSAM format. Subtasks do not produce a dump for X13, X37, and X78 abends. These abends are trapped, and SYSTEM 2000 software issues an appropriate message. All other abends cause a dump to be taken unless the STAE execution parameter equals NODUMP. If you specified STAE=NO, the system produces a dump for all abends.

Chapter 3

SYSTEM 2000® Multi-User™ Environment

THE MULTI-USER ENVIRONMENT: OVERVIEW

Multi-User allows many users to access, retrieve, and update SYSTEM 2000 databases simultaneously. After you initialize a Multi-User session, both SCF and PLEX users can access SYSTEM 2000 software with batch or interactive jobs. All users share one copy of SYSTEM 2000 software.

Multi-User ensures data and structural integrity, allowing multiple users to retrieve and update concurrently. It also handles a mix of short and long running concurrent transactions for one or more databases simultaneously. Multi-User assumes responsibility for scheduling concurrent update requests efficiently. The maximum number of users allowed for a session is established at Multi-User initialization time.

When teleprocessing or batch access rates exceed the throughput capacity of the single-thread configuration, you can request several threads with the THREADS execution parameter. Multiple threads provide concurrent access with interleaving at the resource request (I/O) level. Single-thread processing (the default) allows concurrent access with interleaving at the command level.

Multi-User offers many options that are set by execution parameters when a Multi-User session is initialized. Additionally, during a Multi-User session, the console operator can dynamically change the parameter settings, for example, for priority queuing and for gathering different types of statistics about Multi-User activities. (See **Multi-User Console Operator Commands** on page 3-65 and **MUSTATS Console Operator Commands** on page 3-81.)

Before you initialize Multi-User in a production environment:

- Become familiar with the SYSTEM 2000 database files, sort files, pools, and scratch pads discussed in Chapter 4, "Specifying Files and Buffers."
- If you want to use IBM Cross Memory Services (XMS) with Multi-User, read **Cross Memory Services (XMS) Multi-User Software** on page 3-16. Also, you can use the Multi-User Type 2 SVC discussed in **The Multi-User Type 2 SVC** on page 3-22.
- Refer to the summary chart of execution parameters pertaining to Multi-User (**Multi-User Execution Parameters** on page 3-49) and learn about the different methods of setting and changing the parameter values (**Setting SYSTEM 2000 Execution Parameters** on page 5-9).

- Decide whether you want to activate the Accounting Log during your Multi-User session. The Accounting Log writes a file of statistics such as job names, database usage, file I/Os, and CPU times. Read **Obtaining Resource Statistics: Accounting Log** on page 3-103 for more details.

- Decide whether you want to activate the Diagnostic Log during your Multi-User session. Messages written to the Diagnostic Log provide a sequential listing of events that took place during a Multi-User session. Two utilities are supplied to print the Diagnostic Log and to summarize the event data into several types of reports, for example, thread activity, job activity, and database activity. To find out more about the Diagnostic Log, see **Analyzing Multi-User Events: the Diagnostic Log** on page 3-125.

- Read about the various user exits that can be used to gather statistics, modify input, output, or file allocations during a Multi-User session. For details see Chapter 6, "User Exits."

- Read the discussion of abends under Multi-User in **Abends in the Multi-User Environment** on page 3-46. For specific details about error messages, return codes, user abends, and condition codes relating to Multi-User, see *SYSTEM 2000 Messages and Codes, Version 12, First Edition*.

Multi-User Configurations

Multi-User is initialized as a normal MVS job or as a started task. At most sites, the console operator initializes this job. Multi-User stays inactive until a SYSTEM 2000 user job registers a request via the SYSTEM 2000 Multi-User SVC.

Multi-User services the user request (usually one command), posts the completion of the request to the waiting user job (again via the SVC), and then services the next user request. This cycle of waiting for work, processing commands, and posting of completion continues as long as Multi-User remains active. When a console operator gives a request to terminate Multi-User, the software responds

- after allowing all update commands to complete, closing all databases, and informing all active users.

- after completing the work in progress for all user jobs active at the time of the operator's request.

Multi-User does not require any changes to the operating system. For XMS Multi-User software, you need an SVC table slot, then SYSTEM 2000 software takes care of everything else. If you choose the Type 2 SVC Multi-User software, you must assemble the Type 2 SVC, link into the nucleus, and re-IPL. The memory requirements vary with the number of active users and the number of databases in use at any point in time. Therefore, certain maximums should be considered to reserve sufficient memory for Multi-User. During Multi-User initialization, the maximums used are the values that you specified for the execution parameters.

PLEX programs call an interface, which is linked to the user's PLEX program. All PLEX SYSTEM 2000 commands are passed to the interface. It constructs specific SYSTEM 2000 requests, waits for SYSTEM 2000 completion, and handles special return codes (for example, a directive to terminate the user job).

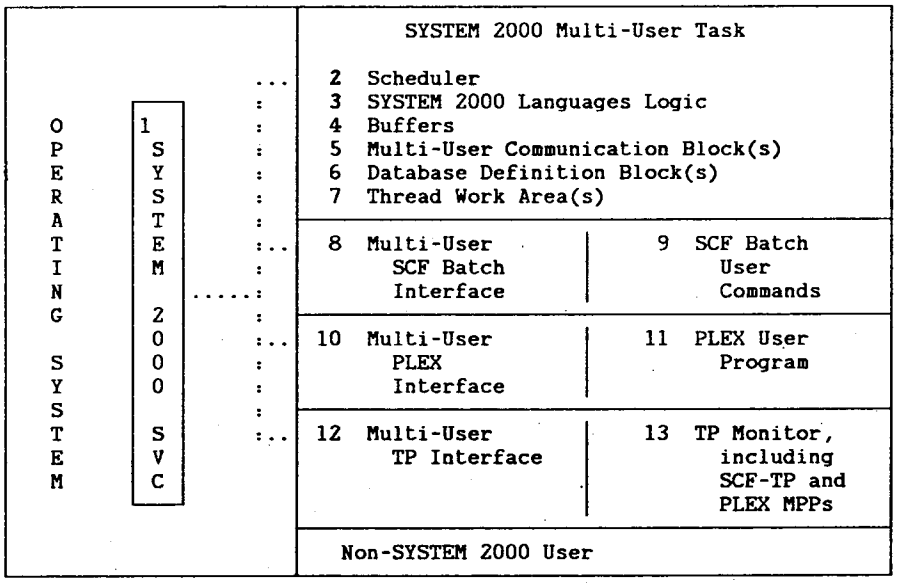
The first use of a SYSTEM 2000 command causes the issuing of ESTAE, and the SVC establishes communication with the SYSTEM 2000 region. SYSTEM 2000 software formats the ESTAE work area when an abend occurs. The formatted work area appears on the Multi-User Diagnostic Log (see **Format of Diagnostic Log Records** on page 3-144).

Note: Multi-User ESTAE processing is independent of the user job ESTAE processing, except that they both attempt to communicate the existence of a problem to each other. For more details on ESTAE processing, see **Abends in the Multi-User Environment** on page 3-46.

The illustration below shows the components of the Multi-User memory configuration. A discussion of each component follows the illustration.

Sizes given for individual components do not include the amount of memory required to open external files. Also, most installations require additional memory for MVS access routines. If the access methods are not resident, they need to be provided. Multi-User uses IBDAM, BSAM, QSAM, and BPAM access methods.

Illus. 3.1 Multi-User Memory Configuration



1 SYSTEM 2000 SVC

The SYSTEM 2000 SVC refers either to the Cross Memory Services (XMS) Multi-User software or to the Type 2 SVC Multi-User software. When details in this manual vary slightly between the two, they will be explicitly differentiated as XMS or Type 2 SVC Multi-User software.

The XMS Multi-User software supplies the PC routine (called S2KPC) that is loaded into CSA storage by the first Multi-User system. Once loaded, S2KPC is fully reentrant and can be shared by other XMS Multi-User systems. The assigned SVC slot contains the address of the S2KPC in CSA, which is updated by the SVCUPDTE macro. S2KPC requires 11K bytes of storage in CSA.

The SYSTEM 2000 Type 2 SVC occupies a minimum of 12K bytes within the nucleus, depending upon the operating system and the maximum number of users allowed.

The SVC coordinates the communication of requests for database activity from user jobs and the responses to those requests to and from Multi-User. The SVC is called by the appropriate Multi-User SCF job, PLEX JOB, or TP interface whenever a user job requests a database activity. When SYSTEM 2000 software completes the requested database activity, the user is notified through the interface via the SVC.

The SVC does not affect MVS control blocks.

For information about Multi-User with IBM Cross Memory Services (XMS), see **Cross Memory Services (XMS) Multi-User Software** on page 3-16.

2 Scheduler

The Scheduler recognizes each user's service request and passes the request to appropriate SYSTEM 2000 code.

3 SYSTEM 2000 Logic

SYSTEM 2000 logic supports all SCF languages and PLEX. The memory requirements depend on the overlay configuration chosen. The default provided on the SYSTEM 2000 delivery tape is a flat system, that is, a configuration that uses no overlays. The PRELNK utility allows you to generate a variety of overlay configurations requiring less memory with slower execution (due to overlay swapping) and more I/Os to the SYSTEM 2000 load library than the default flat system. For details about using PRELNK and sample overlay configurations, see Chapter 7, "Configuring SYSTEM 2000 Software: PRELNK Macro."

SYSTEM 2000 logic satisfies all database activity requests, including buffer management. When Multi-User is initialized, SYSTEM 2000 logic computes the amount of memory required for buffer space, Database Definition Blocks, and control blocks. Then it issues a GETMAIN for that amount of memory. The memory computation is based on the values supplied for the execution parameters in the Multi-User initialization step.

4 Buffers

The memory required for buffers varies depending on your choice of buffer pools. Give one or more POOLn parameter settings (or use the default) to set the size, number of buffers, and pool usage. Before initializing Multi-User, analyze the size, number, and type of buffer pools required for the Multi-User session. Also, read about the PAD*n* parameter that specifies scratch pads used by SYSTEM 2000 scratch files. For a detailed discussion of SYSTEM 2000 files, pools, and scratch pads, see Chapter 4, "Specifying Files and Buffers."

Because buffer allocation is dynamic and because a specific amount of space is reserved for pools and buffers, the potential exists for a database activity request to be unexecutable due to lack of buffer space. In this case, Multi-User issues an appropriate SYSTEM 2000 Error Code. (For details, see *SYSTEM 2000 Messages and Codes, Version 12, First Edition*.)

5 Multi-User Communications Blocks

Multi-User requires a Communication Block for each concurrent user. The memory requirements depend on whether the user run-unit is an SCF TP session, a batch SCF job, or a PLEX program. SCF TP users require no blocks. Each batch SCF user and each PLEX user requires one 10104-byte block. The blocks contain variables that need to be saved and subschema record transfer areas. They are acquired with GETMAIN at initialization time based on the USERS execution parameter.

6 Database Definition Blocks

One Database Definition Block is required for each active database. The block needs an average of 8-1/2K of memory for a database with 430 components. The block contains database DCBs and the database Definition Tables. Multi-User allocates space for this area the first time any user job references the database.

Multi-User issues a GETMAIN when a user requests a database and a Database Definition Block is not available for the additional database (a function of the SDBS and LDBS parameters discussed in **Determining Small Database Definition Blocks: SDBS and SDBSIZE** on page 4-56 and **Determining Large Database Definition Blocks: LDBS and LDBSIZE** on page 4-55). If the GETMAIN is successful, Multi-User retains and reuses the acquired memory until Multi-User is terminated. If the GETMAIN is unsuccessful, an appropriate SCF message or PLEX return code is issued. (For details, see *SYSTEM 2000 Messages and Codes, Version 12, First Edition*.)

7 Thread Work Areas

SYSTEM 2000 software uses a Thread Work Area when processing the command(s) in a segment. Multi-User acquires this space during initialization. It contains the DCBs for the sort files assigned to the thread, common work areas for the SYSTEM 2000 algorithms, save areas for registers, and local storage areas for the SYSTEM 2000 subroutines. One Thread Work Area is needed for each thread specified by the THREADS parameter. The Thread Work Area brought in depends on whether the REPORT processor has been selected (RW=YES execution parameter). A REPORT processor thread requires approximately 9K more bytes of memory.

8 Multi-User SCF Batch Interface

The Multi-User SCF batch interface (SYS2KJOB) is an executable program requiring approximately 30K bytes of memory. This program is a batch job containing SCF commands. The SCF commands from an SYS2KJOB job and the command output results are sent to and from Multi-User via the Multi-User SVC. Multiple copies of the SYS2KJOB interface program can exist in memory at the same time, one for each batch SCF user. JCL statements for alternate user files used in a batch SCF job must be declared in the region in which SYS2KJOB is executed. SYS2KJOB also issues an ESTAE to trap a user job abend and provides for an orderly shutdown of the job.

9 SCF Batch User Commands

The batch SCF input to the SYS2KJOB batch interface program is a fixed-block file containing a variable number of SCF commands. SYS2KJOB reads these commands from the file identified by the DDname S2KCOMD or from an alternate file specified in a COMMAND FILE IS command. The commands are sent to SYSTEM 2000 software for execution via the Multi-User SVC. Results of the commands are sent back via the SVC to SYS2KJOB. Several different batch jobs can run concurrently, each in its own region with its own SYS2KJOB batch interface.

10 Multi-User PLEX Interface

Each PLEX user program invokes the Multi-User PLEX interface MUPLINT. MUPLINT communicates between the PLEX user program and Multi-User. It also issues an ESTAE that traps a user program abend and provides for orderly shutdown of the program.

11 PLEX User Program

The PLEX user program varies in size. It is linked with module S2KPL, which dynamically loads the decision-making module S2KPLR, which loads the Multi-User PLEX interface MUPLINT. S2KPLR decides whether MUPLINT (Multi-User) or S2KPLI (single-user) is to be loaded, which depends on the value of the USE parameter in the PLEX START S2K command. A value of 0 means single-user, and a value of 1 means Multi-User. Several different PLEX user programs can run concurrently, each in its own unique region with its own copy of MUPLINT.

12 SYSTEM 2000 TP Interface for Multi-User

The Multi-User TP interface varies in size depending upon the specific TP monitor being used, for example, TSO or CICS. TSO has the same requirements as batch PLEX described earlier. CICS requires the installation of the SYSTEM 2000 interface to CICS.

13 TP Monitor

The size of the TP monitor depends on which TP monitor is used.

Multi-User Processing

For retrievals and most updates, SCF and PLEX users interact with a database exactly the same way under Multi-User as they do for single-user jobs. Multi-User allows for concurrent updating and schedules the updates. Therefore, end users do not have to manually schedule updates to avoid conflicts as they do when running single-user jobs.

The PQA execution parameter enables you to specify priorities for dispatching into a thread. After the user job is in a thread, Multi-User polls user requests in a fixed, circular fashion, so that no user has a higher priority than any other. Each user is processed "in turn" and is given the chance to execute one SYSTEM 2000 command. Honoring user requests depends on current availability of resources. That is, in any given "turn," Multi-User can bypass a request because another user has exclusive use of a necessary resource.

A WAIT status is caused by a command actually being issued. Therefore, lockout is at the command level, not the program or session level.

SYSTEM 2000 software enforces security at the item and record level by examining each user request. Items and records in the request are checked against the authorities assigned to the password in use. If the user has the proper authorities, the request is passed along for processing. Otherwise, the request is denied, and SYSTEM 2000 software issues an appropriate error message.

Database Status

The most commonly used mode of operation under Multi-User is non-exclusive use of databases. Any SCF or PLEX user has options available to hold the database temporarily from other users when a database is being accessed by multiple users under non-exclusive use. Also, the PLEX HOLD option allows record lockout. The HOLD option is used only for updating one or more specific records; users are allowed to retrieve data concurrently from the rest of the database.

In a Multi-User environment, any user can request exclusive use of a database. Exclusive use is mandatory for several commands. It disallows all access by other users until exclusive use is relinquished. Creating a new database or restoring a database triggers exclusive use of the database automatically. (Each single-user job has exclusive use of a database by default.)

The status of a database is one of the following:

- unopen

  Multi-User has not received any requests for the database.

- open for exclusive use

  One user is accessing the database with exclusive use. Multi-User does not grant exclusive use of a database to a user if any other user has the database open in any status. After exclusive use is granted, no other user can open the database for any use until the exclusive use is relinquished. Exclusive access to the database ends when the user issues an EXIT command or reopens the database with non-exclusive use (that is, CONTROL: DBN IS XYZ: for SCF jobs and, for PLEX jobs, a CLOSE command followed by an OPEN command).

For SCF, a database is automatically under exclusive use when it is being defined, restored, or opened with the EXCLUSIVE DATA BASE NAME IS command.

For PLEX users, a database is under exclusive use when it is opened with the explicit LOCK or EXCLUSIVE option. A PLEX program can open a database for exclusive use only if no other databases are currently open in the program. Several databases can be open at a time in one PLEX program if none are under exclusive use. A CLOSE must be issued for each database under non-exclusive use before the PLEX program is granted exclusive use for a database.

The PLEX user can choose exclusive use for a database to protect results of a LOCATE command. Non-exclusive use allows any user to update or retrieve concurrently in round-robin fashion. Therefore, after a LOCATE is issued, another user could update the database. thus changing data values. Some or all of the data records previously qualified for the Locate File may no longer be qualified by the where-clause.

For example, suppose User 1 locates all employees in the Accounting Department and then User 2 changes an employee from the Accounting Department to the Finance Department. User 1 would inadvertently process that employee as part of the Accounting Department. To guard against this possibility, User 1 could either open the database for exclusive use (LOCK or EXCLUSIVE), or open the database non-exclusively and make sure each employee is still in the Accounting Department when the next GET command is issued. The latter method is preferable, because it enhances overall throughput for a database.

- open for non-exclusive use

The database is open for non-exclusive use by one or more SCF or PLEX users. This unrestricted, non-exclusive use allows concurrent users to interleave update and retrieval requests. Concurrent updates are controlled through the use of the HOLD option for retrieval commands. Each user is polled in turn and given a chance to process a segment consisting of a single SYSTEM 2000 request or several requests. All local and global holds affect a database being accessed with non-exclusive use.

Exclusive use is not granted to any user until all non-exclusive users relinquish access to the database. A database is under non-exclusive use when one or more SCF users opens the database with a DATA BASE NAME IS command or when one or more PLEX programs opens the database with an implicit open, an OPEN, or an OPENR command.

Reusable Space in a Database

In single-user and Multi-User sessions, reusable space created by previous REMOVE TREE command(s) is not available for reuse with INSERT TREE operations, unless the user is the only one using the database. However, in SCF or PLEX queue mode, reusable space created during the session is simply not available for reuse during the current session, regardless of whether that user is the only one using the database.

Note: REORGANIZE, REMOVE TREE, and ASSIGN TREE do not require exclusive use. because they are treated as regular updates. Reusable space is marked for reuse but is temporarily not reused if more than one user has the database open. To utilize reusable space, the user must have exclusive use of the database for the removals and insertions. If only one user opens the database, creating a data record marks reusable space for all data records as usable. Also, any job that opens a database (even for retrieval only) marks reusable space for all data records as usable if it is the first job to open the database.

Local and Global Holds

Local and global holds under non-exclusive use in a Multi-User environment reduce database contention for update processing.

A global hold protects an entire database from updates but allows other SCF and PLEX users to do retrievals. In this case, no other user can obtain a hold or do any updates.

A local hold protects a specific data record in a database from update, except for a REMOVE TREE at a higher level. It also allows other users to obtain local holds and to do updates to other data records in the same database. Local holds allow other users to retrieve records not being held.

UPDATE status requires all retrievals on a database to stop before updates can be processed. An update does not drop a local hold.

SUSPEND status suspends a user job until the current status is dropped and the command can be restarted.

HOLD for PLEX immediate mode A HOLD option from a PLEX user has different meanings, depending on whether the HOLD is requested during immediate mode or queue mode.

For retrieval commands in immediate mode, the scope of a HOLD is reduced to a single data record (local hold) instead of the entire database (global hold). A user in immediate mode is permitted to obtain more than one local hold for each opened database if the MULTIPLE HOLDS option is set. A local hold is maintained until a synchpoint occurs.

PLEX local holds are handled differently than global holds. The rules for local holds are as follows:

- A PLEX user can have more than one local hold active for each opened database if the MULTIPLE HOLDS option is in effect.
- A local hold can be obtained while another PLEX user's local hold is active as long as they are for different data records. Local hold contention is monitored during processing, after the data record to be held is identified.
- Retrievals are processed even if local hold(s) are active.
- Active local holds cause global hold requests to be suspended.
- A global hold causes local hold requests to be suspended.

- An update request can be scheduled while local holds are active. An update request does not drop a local hold.

- UPDATE status causes any new local hold requests to be suspended.

Because an update can be scheduled while local holds are active, a user might do a REMOVE TREE or MOVE TREE that would remove a data record held by another user. SYSTEM 2000 software does not check to prevent this situation. However, if the user holding the removed data record attempts to update it, Multi-User issues Return Code 09, which indicates that the operation cannot be done.

If several PLEX programs are doing updates and the programs process updates across databases, a convention for accessing databases must be established. When a PLEX user requests a local hold on a data record that is held, the request is suspended until the hold is dropped unless another user is waiting for something being held. This is a potential deadly embrace situation. The user is given a non-zero return code, and all his holds are dropped. Otherwise, two or more programs could interlock while retrying local hold requests if each program held a data record requested by the other program.

For example, suppose User 1 (PLEX) has a local hold on data record 650 in database A, and User 2 (PLEX) has a local hold on data record 320 in database B. When User 1 issues HOLD for data record 320 in database B, that user waits for database B. If User 2 requests record 650 in database A, that user receives Return Code 110. User 2 can then issue a HOLD for data record 650 in database A because all User 1 holds were dropped when the 110 return code was issued.

This deadly embrace logic prevents the stacking of users suspended on local holds. It also prevents the situation in which User 1 holds data record 1, User 2 holds data record 2 and is waiting on data record 1, User 3 holds data record 3 and is waiting on data record 2, and so forth. In this example, User 3 receives a 110 return code, instead of being on two levels of hold (that is, 3 waiting on 2, who is waiting on 1).

The PLEX DROP HOLD command can be used to drop all local holds active for the user; pending inserts are not forced. The DROP HOLD command behaves like the COMMIT command and causes a synchpoint to occur.

The HOLD option for the LINK command automatically causes a local hold on each record in the link cascade. For details see the *SYSTEM 2000 PLEX Manual, Version 12, First Edition.*

Consider the following when analyzing PLEX local holds:

- If the LHOLD execution parameter equals YES, local holds are required before each PLEX update request.

- Because all PLEX immediate mode HOLD requests are processed as local holds, the PLEX program receives a return code if the HOLD request is for a data record held by another user, and the potential for a deadly embrace exists.

- PLEX programs can have more than one simultaneous local hold on a database. The user can hold two different data records in a database and then update one or both records. All local holds for all databases are held until a synchpoint occurs.

See the discussion of the LHOLD execution parameter in **LHOLD Parameter** on page 3-58.
Also, see the ENABLE/DISABLE MULTIPLE HOLDS command in *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

HOLD option for PLEX queue mode The HOLD option in PLEX queue mode establishes a global hold before queue mode processing continues. The HOLD option cannot be used to establish a local hold in PLEX queue mode.

Holds for SCF SCF holds are always global holds that are invoked by SYSTEM 2000 software. For example, a global hold is automatically requested for an update command. The global hold remains active during where-clause processing, and UPDATE status is requested after the schema items to be updated have been selected.

Although adding local hold logic for PLEX does not alter the SCF update process, it could cause a delay in scheduling global holds if there are multiple PLEX users doing updates. The reason for a delay is that a PLEX local hold request can be processed while local holds from other PLEX users are active on that same database.

An SCF global hold request is suspended in a thread until all PLEX local holds are dropped. For example, if a PLEX user has an active local hold on database A and an SCF user issues an update command for database A, Multi-User suspends the SCF user in a thread because a global hold is required before the command can be processed.

A global hold is suspended if one or more local holds are active, and it remains suspended until all local holds are dropped. If a second PLEX user issues a retrieval command for database A using the PLEX HOLD option (which means a local hold must be obtained), that user command is processed ahead of the SCF command, because a local hold can be established while other local holds are active. **Note**: This can occur only if an available thread exists for processing additional users. Remember that SCF hold requests are suspended in a thread.

CROSS MEMORY SERVICES (XMS) MULTI-USER SOFTWARE

SYSTEM 2000 software supports Multi-User software that runs with either the IBM Cross Memory Services (XMS) or the Type 2 SVC.

Multi-User running with the IBM Cross Memory Services is called XMS Multi-User. Multi-User running with the Type 2 SVC is called SVC or non-XMS Multi-User (discussed in **The Multi-User Type 2 SVC** on page 3-22)

Even though the installation and JCL of XMS Multi-User and SVC Multi-User are different, SYSTEM 2000 software remains the same.

Differences between SVC and XMS Multi-User Software

XMS Multi-User Software	SVC Multi-User Software
12K of CSA	CSA copy areas required are dependent upon number of users.
Relinking of nucleus is not required.	Must link the Type 2 SVC into the nucleus.
One SVC number can serve several Multi-User systems.	One SVC number is required for each Multi-User.
Data areas reside in Multi-User private area.	Data areas use the Common Service Area (CSA).

Execution of XMS Multi-User Software

The following topics describe the programs, modules, and procedures for executing Multi-User software with XMS.

S2KCMC control program S2KCMC is the control program that sets up the cross memory environment and begins XMS Multi-User execution. You must execute S2KCMC from an authorized library, and you must link it as an authorized program.

At installation time, you must execute a zap (supplied by SAS Institute Inc.) to put the SVC number into the S2KCMC control program. Upon execution, S2KCMC puts the entry point of S2KPC into the system SVC table where all address spaces can access the SVC number. S2KCMC also writes the SVC number to the S2KCOM file.

When you execute S2KCMC, it first checks to see whether the S2KPC load module is in the Common Storage Area (CSA). If it is not, S2KCMC loads it into the CSA. After loading S2KPC, S2KCMC sets up the cross memory environment by executing IBM macros that allow XMS Multi-User to use the IBM Cross Memory Services. Next, S2KCMC builds an S2KCOM file for each Multi-User that it executes. Then S2KCMC attaches S2000 (Multi-User) as an unauthorized subtask. S2000 creates the Multi-User SVC table and calls the SVCADR routine. After attaching S2000, S2KCMC issues a WAIT and waits until S2000 terminates normally or abnormally.

When S2000 terminates, S2KCMC attaches CLEARS2K as an unauthorized subtask. CLEARS2K uses the Multi-User SVC table to post users that were not posted when Multi-User terminated. S2KCMC does a FREEMAIN to release the Multi-User SVC table and releases the IBM Cross Memory Services resources back to MVS. Finally, the S2KCOM file is cleared.

S2KPC load module The S2KPC load module performs the communication between Multi-User and the dependent regions. S2KCMC loads the S2KPC load module into the CSA.

When Multi-User or the dependent region enters S2KPC, S2KPC issues the PCLINK stack macro. This is done to preserve the environment of the address space that issued the program call (PC) instruction. Before S2KPC issues the program transfer (PT) instruction to return, it issues the PCLINK unstack macro to restore the environment of the address space that called it.

When Multi-User terminates, it does not unload the S2KPC load module from the CSA. The only way to clear the S2KPC load module out of the CSA is to re-IPL MVS.

S2KCOM file The S2KCOM file contains control information, which is established by S2KCMC and used by Multi-User and the dependent regions. S2KCOM is a 30-byte file containing the SVC number, cross memory authorization information, and program call information (including the program call number). The S2KCOM file must be a permanent data set, and it must reside on disk.

The S2KCOM file is a VSAM file, mainly to support the CICS command-level interface to SYSTEM 2000 software. If your site was running XMS Multi-User before Version 12, S2KCOM being a VSAM file will have minimum impact at your installation. If you delete your current S2KCOM file and allocate the VSAM file with the same name, you do not have to change any of your old JCL or CLIST procedures. The only change required is to your CICS File Control Table (FCT), where you need to define S2KCOM to CICS as a VSAM file.

After S2KCMC builds the S2KCOM file, S2KCMC issues a system ENQ macro to reserve this file based on the data set name (DSN) and the volume serial number (VOLSER). This prevents another Multi-User from using the same S2KCOM file. When another Multi-User tries to use a DSN and VOLSER already being used by an S2KCOM file, the second Multi-User issues an error message and terminates.

Each Multi-User is identified by its unique S2KCOM file and by its unique program call number within that file.

The DISP=SHR parameter must appear in the S2KCOM statement because the dependent regions using this particular Multi-User need access to this file.

SVCADR routine At installation time, your systems programmer must zap the SVCADR routine at location x'8' with NOP (07010701). SVCADR reads the S2KCOM file to determine the proper SVC number to use. The SVC number that was zapped in at installation time is written to the S2KCOM file by S2KCMC. The S2KCOM file is read only the first time SVCADR is called. See the SYSTEM 2000 installation instructions for the JCL used to perform these zaps.

The SVCADR routine then issues a program call (PC) instruction in Multi-User. The program call is to S2KPC.

CLEARS2K S2KCMC attaches CLEARS2K as an unauthorized subtask, regardless of whether S2000 terminated normally or abnormally. CLEARS2K uses the Multi-User SVC table to post users who were not posted because S2000 terminated abnormally.

In the cross memory environment, the Multi-User SVC table is obtained dynamically with GETMAIN from the private area that Multi-User uses. When CLEARS2K terminates, it passes control back to S2KCMC.

JCL to Initialize XMS Multi-User Software

Some of the JCL required to execute XMS Multi-User is different from that used to initialize SVC Multi-User.

To initialize XMS Multi-User, you must replace the first few lines of JCL required to initialize SVC Multi-User with the following lines of JCL. The rest of the XMS Multi-User JCL is the same as it is for SVC Multi-User.

```
//JCLMU1 JOB
//MUMAIN   PROC OUT=A,
//             WRKUNIT=SYSDA,FCP=1,FCS=1
//S2K      EXEC PGM=S2KCMC
//STEPLIB  DD   DSN=authorized-library-DSN,DISP=SHR
//S2KLOAD  DD   DSN=S2K.R120.LOAD,DISP=SHR
//S2KCOM   DD   DSN=S2K.R120.COM,DISP=SHR
```

Terminating XMS Multi-User Software

You can terminate XMS Multi-User (S2000) with the WHEN, NOW, or FORCE options of the Multi-User console CANCEL command. The FORCE option terminates S2000 abnormally if the WHEN and NOW options fail. The FORCE option can be used only with the S2K option (not the TP option).

When the FORCE option abnormally terminates the S2000 subtask, S2KCMC regains control. S2KCMC then attaches CLEARS2K, which uses the Multi-User SVC table to post the users not already posted. CLEARS2K posts these users with an abnormal termination condition. Also, any updates pending or in progress will not be completed and will probably cause databases to be flagged as damaged.

The users that CLEARS2K posts are not posted if you issue an MVS CANCEL command while running XMS Multi-User. The Multi-User table goes away, and an operator must cancel each user individually. Therefore, do not issue an MVS CANCEL for XMS Multi-User.

If you issue the DUMP option with the FORCE option, a SNAP dump is produced, not a system dump. The ID of the snapshot dump is 61.

The syntax for terminating XMS Multi-User is as follows:

```
|CANCEL  S2K   |,WHEN      [,DUMP]
|C             |,W
               |,NOW
               |,N
               |,FORCE
```

XMS Multi-User Effects on SYSTEM 2000 Execution

The following topics discuss memory requirements for XMS Multi-User software and how to use CICS with XMS Multi-User software.

Memory requirements for the XMS Multi-User region XMS Multi-User uses only 12K of CSA. S2KCMC loads the S2KPC load module into the CSA so that both Multi-User and dependent jobs can access it.

Multi-User uses GETMAINs to obtain data areas from the private area of the Multi-User region. Therefore, in addition to the private area required to execute S2000 object code, you must allow enough space in the Multi-User region for the copy areas S2000 uses. For details about computing the size of copy areas for XMS Multi-User, see **COPYAREAn Parameter** on page 3-54.

Because the Multi-User SVC table also resides in the Multi-User region, you must also calculate the storage needed for this table and add it to the storage Multi-User needs to execute. To calculate the storage needed by the Multi-User SVC table, use the following algorithm:

((USERS + TPTHREADS) * 40) + 172

where

USERS is the execution parameter specifying the maximum number of users of batch SCF, batch PLEX, and PLEX TP that can be executing SYSTEM 2000 software concurrently.

TPTHREADS is the execution parameter specifying the number of active SCF TP users that can be queued for SYSTEM 2000 processing.

Memory requirements for dependent jobs The memory requirements for dependent jobs are the same for XMS Multi-User software and SVC Multi-User software.

How XMS Multi-User software affects databases The database files are the same for XMS Multi-User and SVC Multi-User. The same database can be used by either type of Multi-User. The rules for databases being used by more than one XMS Multi-User are the same as for that situation in SVC Multi-User.

Parameters The syntax of execution parameters is the same for XMS Multi-User software and SVC Multi-User software.

XMS problem investigation In addition to the facilities already available for problem investigation, there is an FRR (Functional Recovery Routine) BINDFRR for XMS Multi-User software. BINDFRR is entered if an error occurs while executing some of the macros used to initialize or terminate the dependent region or XMS Multi-User software.

CICS requirements for XMS Multi-User software Two changes to CICS are required to run XMS Multi-User:

- You must zap the SVCADR routine.
- You must add a DD statement for S2KCOM.

Zap the SVCADR routine in S2KCICS with NOP (07010701) at location x'8'. In the CICS execution JCL, add a DD statement that points to the S2KCOM file for the appropriate Multi-User.

Multi-User console commands for XMS Console commands function the same with XMS Multi-User as with SVC Multi-User except for the FORCE option in the Multi-User console CANCEL command. For SVC Multi-User, the FORCE option is equal to an MVS CANCEL. For XMS Multi-User, the FORCE option functions as an MVS CANCEL, but control is returned to S2KCMC. S2KCMC then attaches CLEARS2K to post any users not posted. See **Terminating XMS Multi-User Software** on page 3-18.

Running More Than One XMS Multi-User System

More than one Multi-User can be running at the same time on the same CPU. You can run more than one XMS Multi-User, more than one SVC Multi-User, or any combination of the two. Both types of Multi-User can run in the same MVS environment, but they must be totally distinct and separate. each with its own SVC number and set of executable load libraries. .To run more than one type of Multi-User, you must install each type of Multi-User separately. The installation process for XMS Multi-User and SVC Multi-User must be distinct and separate.

You can run up to 4096 XMS Multi-User systems using the same S2KPC load module through the same SVC number, because S2KPC is reentrant. To uniquely identify each XMS Multi-User system, you must allocate a unique S2KCOM file for each XMS Multi-User. If you are running a dependent job against an XMS Multi-User, the job must refer to the proper S2KCOM file (DISP=SHR).

If you need a different S2KPC load module, you must zap a different SVC number into S2KCMC to bring up S2KPC. Another 12K of CSA storage must also be available for S2KPC. Another S2KPC load module and SVC number allows you to bring up and run another complete group of XMS Multi-User systems.

SVCUPDTE Services for XMS SVC Code

SVCUPDTE services are supported in Release 11.6 and later releases. The SVCUPDTE macro updates the SVCTABLE for any MVS/XA system. For non-MVS/XA systems, processing has not been altered.

SVCUPDTE services are required as soon as your site starts running level 2.2 of the XA control program. You must either use Release 11.6 (or later) or obtain a Special Zap from the Institute in order to make prior releases of SYSTEM 2000 software compatible with the XA 2.2 control program. XA 2.2 was announced with JES3 2.2.0.

Summary

The space required in the CSA to run XMS Multi-User is less than that required to run SVC Multi-User. With XMS Multi-User, the data reside in the private area. The only part of XMS Multi-User that resides in the CSA is the load module S2KPC. S2KPC performs the communication between the dependent region and Multi-User. S2KPC is loaded into the CSA by the authorized control program S2KCMC that initializes XMS Multi-User. When XMS Multi-User terminates, S2KCMC performs any necessary clean-up. During installation, S2KCMC is zapped with an SVC number.

S2KPC is associated with an SVC number. Because S2KPC is reentrant code and all data areas are now in the private area of the Multi-User region, more than one XMS Multi-User can run using the same S2KPC and SVC number.

Each XMS Multi-User must have a unique S2KCOM file. The S2KCOM file contains control information that is established by S2KCMC and used by XMS Multi-User and the dependent region. When the S2KCOM file is built, S2KCMC writes the SVC number, which was zapped into S2KCMC during installation, to the S2KCOM file.

For more information on Cross Memory Services see the following IBM manuals:

- *Cross Memory Services User's Guide* (GG22-9231)
- *OS/VS2 System Programming Library: Supervisor* (GC28-1046)
- *OS/VS2 System Logic Library, Volume 6* (LY28-1079)
- *OS/VS2 Systems Programming Library: Debugging Handbook* (GC28-1047), (GC28-1048), (GC28-1049)
- *OS/VS2 Environmental Recording Editing and Printing (EREP) Program* (GC28-0772).

THE MULTI-USER TYPE 2 SVC

Introduction

The SVC macro supplied by the Institute facilitates generating the specialized Type 2 SVC used by Multi-User software, which must be tailored to each installation's operating system. **Note**: If you want to use the IBM Cross Memory Services (XMS) feature, see **Cross Memory Services (XMS) Multi-User Software** on page 3-16 for details about setting up one or more Multi-User systems.

Each installation of SYSTEM 2000 software using the Multi-User capability must generate and install a specialized Multi-User Type 2 SVC. The Type 2 SVC macro is implemented as an Assembler macro.

Input to the Type 2 SVC macro is provided through a small number of macro keyword parameters. These parameters specify the type of operating system being supported and other operating system dependencies.

The Multi-User Type 2 SVC

- runs enabled for most of its execution (Type 2 SVC)
- performs extensive validation of input parameters
- causes data movement in the protect key of the receiver where possible
- is generated by the user for a specific operating system environment
- takes MVS error recording techniques into consideration
- occupies a minimum of 12K.

Special Considerations for the Type 2 SVC

- **Operating System Support** Multi-User supports the following systems:

  VM/CMS
  MVS (VS2 SCP R2.0 and later releases)
  MVS/XA and MVS/ESA

- **Mode of Execution** The Multi-User SVC runs enabled, as a Type 2 SVC, during most of its execution. The only time the SVC runs disabled is when the SCP queues are searched or when disabling is required to call certain SCP routines. With enabled mode, both Multi-User and installation throughput is increased. For VS installations, the occurrence of disabled page faults is greatly reduced. For MVS installations, the SCP locking scheme provides the necessary serialization. The local and SA locks are the only locks directly held.

- **System Integrity** The Multi-User SVC does extensive validation of its input parameters. This accomplishes two major objectives: protection against an SVC call by an unauthorized, disruptive user and greater system integrity in the event of a Multi-User programming error. Unauthorized users are prevented from SVC use by validation of supplied data structures. Bounds, limits, and ownership tests done by the SVC reduce possible system damage due to Multi-User programming error.

- **Data Movement** Where possible, the Multi-User SVC does data movement in the protect key of the data receiver. By assuming the receiver's key, the SVC further enhances system integrity.

- **SVC Generation** To provide an SVC tailored to a specific operating system environment, the Multi-User SVC is implemented as an Assembler macro. By setting the keyword macro parameters, you can generate a specialized SVC. Sample JCL for SVC creation is given in Illus. 3.2 on page 3-24. See also sample JCL in the installation instructions for Release 12.0 of SYSTEM 2000 software.

- **MVS Error Recording** The Multi-User SVC, running under the MVS operating system, incorporates the SETRP macro to record error conditions on the system's SYS1.LOGREC data set. This error information should help the DBA and systems personnel in providing support to the Multi-User system.

- **Memory Storage** The Multi-User SVC and S2KCOPY occupy 12K or more bytes of storage within the nucleus, depending on the operating system and parameters selected at generation time.

Generating the Multi-User Type 2 SVC

To provide an SVC that can be tailored to a specific operating system, the Multi-User Type 2 SVC has been written as an Assembler macro (MRISVC). Through the use of a limited number of keyword macro parameters, you can generate a specialized Type 2 SVC. Illus. 3.3 on page 3-25 shows the keyword parameters and their operands and discusses their usage. See the IBM Assembler Language and Assembler Programmer's Guide manuals for the proper method of coding and assembling a macro.

All parameters must be specified when you are generating the Type 2 SVC, except for the FETCH parameter, which is not required for MVS SVC generation. Parameters must be separated by commas. Be sure to specify the proper set of operands when generating the SVC for the following reasons:

- The Type 2 SVC receives the operating system characteristics when SYSTEM 2000 software (or the TP facility) is initialized. If the system characteristics do not match, the SVC rejects the initialization request.

- If the USERS or TPUSERS keyword parameters specified in Type 2 SVC generation are insufficient for the USERS or TPTHREADS SYSTEM 2000 execution parameters, as presented during Multi-User (or TP) initialization, the SVC rejects the initialization request.

The JCL shown in Illus. 3.2 generates the SVC for an MVS system with 32 non-SCF TP users and SCF TP support for five users. The following SCP macros are needed: ABEND. CVT, GETMAIN. IHAASCB, IHAFRRS, IHAPSA, IHASDWA, IHASRB, IHAWSAVT, IKJTCB, PURGEDQ, SCHEDULE, SETFRR, SETLOCK, SETRP, and SPOST. Also, reverse the order of concatenation for the SYSLIB files if MACLIB has a block size larger than S2K.R120.TEST (BLKSIZE = 800).

Illus. 3.2 JCL for Type 2 SVC Generation for MVS

```
//JCLSVC JOB
//*
//*************************************************************
//* INSTALL MULTI-USER SVC FOR MVS SYSTEM
//*************************************************************
//*
//SVCJOB2  PROC OUT=A,WRKUNIT=SYSDA
//ASM      EXEC PGM=IEV90,
//              PARM='NODECK,OBJECT'
//SYSLIB   DD   DSN=SASXYZ.R120.TEST,DISP=SHR,
//              DCB=BLKSIZE=32000
//         DD   DSN=SYS1.MACLIB,DISP=SHR
//         DD   DSN=SYS1.AMODGEN,DISP=SHR
//SYSUT1   DD   UNIT=&WRKUNIT,SPACE=(1700,(400,50))
//SYSUT2   DD   UNIT=&WRKUNIT,SPACE=(1700,(400,50))
//SYSUT3   DD   UNIT=&WRKUNIT,SPACE=(1700,(400,50))
//SYSPRINT DD   SYSOUT=&OUT
//SYSPUNCH DD   DUMMY
//SYSLIN   DD   DSN=&&SVCOBJ(IGCOBJ),DISP=(NEW,PASS),
//              UNIT=&WRKUNIT,SPACE=(3200,(50,1,1))
//SVCLNK   EXEC PGM=IEWL,
//              PARM='MAP,XREF,LET,LIST,SIZE=(164K,24K)'
//SYSLMOD  DD   DSN=SASDWP.R120.LOAD,DISP=OLD
//SYSPRINT DD   SYSOUT=&OUT
//SYSUT1   DD   SPACE=(1024,(400,20)),UNIT=&WRKUNIT
//LOAD     DD   DSN=SASDWP.R120.LOAD,DISP=SHR
//OBJECT   DD   DSN=&&SVCOBJ,DISP=(OLD,PASS,DELETE)
//         PEND
//*************************************************************
//*
//*     CHANGE IGC237 TO 'IGCNNN' WHERE NNN REPRESENTS
//*     THE SITE TYPE 2 SVC NUMBER ASSIGN TO MULTI-USER.
//*
//*************************************************************
//GO       EXEC SVCJOB2
//ASM.SYSIN    DD  *
         MRISVC CSECT=IGC237,OS=MVS,USERS=32,TPUSERS=5
         END
//SVCLNK.SYSLIN  DD  *
  CHANGE S2KCOPY(R120COPY)
  INCLUDE OBJECT(IGCOBJ)
  CHANGE S2KCOPY(R120COPY)
  INCLUDE LOAD(S2KCOPY)
  NAME  IGC237(R)
//*
```

Illus. 3.3 Keyword Parameters for the MRISVC Macro

Keyword and Operand	Purpose/Valid Operands
CSECT=IGC*nnn*,	provides a CSECT name to the SVC being generated. Operand must be equal to IGC*nnn* where *nnn* is a number greater than 199 and less than 256.
OS=*type*,	indicates to the SVC what type of operating system it is to run under. The operand should equal MVS for VS2 Release 2.0 and later.
FETCH=YES | NO,	indicates to the SVC whether fetch-protect exists for this system. This parameter is ignored for MVS systems. NO is the default.
USERS=*nnn*,	indicates to the SVC the maximum number of queued run-units that are batch SCF, batch PLEX, and PLEX-TP jobs (does not include SCF TP users). The operand *nn* represents a number greater than zero and less than 231.
TPUSERS=*nn*,	indicates to the SVC the maximum number of queued SCF TP users to be supported. These are SCF users accessing SYSTEM 2000 software through a TP monitor like CICS or TSO. The operand *nn* represents a number greater than or equal to zero and less than 231.
TSO=YES | NO,	indicates whether code to support TSO users should be included in the SVC. This parameter is not required for MVS systems.

Queued means the job is waiting (for a thread) in an SVC slot. The number of slots is equal to the value of the TPUSERS and USERS parameters. If no slot is available, the user is signed off with a "busy" message.

Installing the Type 2 SVC

After the SVC has been generated, it must be included in the SCP nucleus. Also, the S2KCOPY routine must be included if the interregion copy function is used, that is, under fetch-protected MVS operating systems. S2KCOPY exists as a load module on the SYSTEM 2000 load library and is approximately 3.5K bytes. It must be link-edited with the SVC. Using standard SCP procedures, link the SVC and S2KCOPY (if required) into the SCP nucleus. Use the link-edit parameters XREF, LIST, LET, NCAL, RENT, and SCTR.

In each of the systems where user data areas reside in storage that cannot be read by SYSTEM 2000 software (fetch-protected, MVS, or TSO), an interregion copy area pool is built. In each case, this pool is in the same storage protect key as the Multi-User region so it can be modified by SYSTEM 2000 software while in the problem program state. This pool is built in the Common Service Area (CSA) for MVS.

For MVS only, the SCP SVCTABLE must be zapped to support the SVC. The SCP routine IEAVESVC (component SC1C5) shows the format of the SVCTABLE. This allows non-authorized users to access the SVC and acquire the local lock for the SVC.

Multiple Copies of Non-XMS Multi-User Software

You cannot have more than one S2KCOPY at a time. If you need to have Multi-User for R10.1 and also for R11.5, for example, you must rename one of them. Also, the S2KCOPY for R11 and R11.5 are incompatible; if you need both, rename one of them. For more details, see the installation instructions for Release 12.0 of SYSTEM 2000 software.

CLEARING THE MULTI-USER SVC: CLEARS2K

The CLEARS2K utility initializes the tables in the SVC, posts any user still active when Multi-User terminated, and frees CSA (if MVS) in preparation for initializing Multi-User. When initializing Multi-User, CLEARS2K should be run in Job Step 1, and S2000 should be run in Job Step 2. CLEARS2K must never be executed when Multi-User is running.

If the XMS version of Multi-User is being used when an abend occurs, CLEARS2K is run automatically before terminating. However, if you are using the Type 2 SVC version of Multi-User, you must run CLEARS2K.

If Multi-User is canceled through MVS rather than through Multi-User console operator control, MVS does not inform the SVC of its termination. In this case, you must use the CLEARS2K utility to clear the SVC table before attempting to restart Multi-User.

Note: If this utility is executed inadvertently while Multi-User is already running, it clears the SVC table while users' jobs are executing. A good procedure (and a common one) is for CLEARS2K to be Job Step 1, and S2000 to be Job Step 2 for Multi-User initialization.

Messages and Codes for CLEARS2K

Messages and codes for the CLEARS2K utility are discussed in *SYSTEM 2000 Messages and Codes, Version 12, First Edition.*

JCL for CLEARS2K

This illustration shows sample JCL for executing CLEARS2K.

Illus. 3.4 CLEARS2K Job Control

```
//*     ******************************************
//*     *                                        *
//*     *     CLEAR MULTI-USER SVC               *
//*     *                                        *
//*     ******************************************
//*
//*
//CLEARS2K PROC INDEX=S2K
//CLEAR    EXEC PGM=CLEARS2K
//STEPLIB  DD   DSN=&INDEX..R120.LOAD,DISP=SHR
//         PEND
//*
//*
//         EXEC CLEARS2K
//*
//*
```

INITIALIZING MULTI-USER SOFTWARE

After SYSTEM 2000 software is installed, Multi-User software can be brought up. Many variables apply to a Multi-User environment. Most of these variables are execution parameters, and each has a default setting. Other factors that apply to Multi-User are listed below. Consider each before initializing Multi-User software.

- For SVC Multi-User, install the SVC supplied by the Institute. See **The Multi-User Type 2 SVC** on page 3-22.
- Determine the total number of SYSTEM 2000 users and the number of SCF TP users.
- Consider all Multi-User execution parameters. Each parameter assumes a default setting if not specified. See **Multi-User Execution Parameters** on page 3-49.
- SYSTEM 2000 software can dynamically allocate database files, Savefiles, and Keepfiles, as well as all work files. Or, users can issue the ALLOC command within an SCF session to allocate the database files. Or, SYSTEM 2000 software can look for the database file specifications in the S2KDBCNT table if the ALLOC execution parameter allows the lookup. Therefore, most of the files used in a Multi-User session do not have to be allocated in the Multi-User initialization step.
- If SCF TP users want to access alternate user files during a Multi-User job, specify the DD statements for these files during Multi-User initialization. Each user could have one or more alternate Command Files, Data Files, Report Files, or Message Files. SYS2KTPI (SCF TP) users can also use local files. Batch SCF user files are in the dependent region.

Multi-User is initialized by executing S2000, which loads the necessary load modules. SYSTEM 2000 load library modules are described in Chapter 7, "Configuring SYSTEM 2000 Software: PRELNK Macro."

During Multi-User initialization, you can preallocate all database files, scratch pads, sort files, and other optional SYSTEM 2000 files, or you can let SYSTEM 2000 software allocate them dynamically. The execution parameters are specified for a Multi-User session in general and not for an individual SCF job. For example, if the REPORT processor is not initialized for Multi-User (RW=YES), an individual SCF job cannot use the REPORT processor.

The remaining topics in this section cover various topics pertaining to Multi-User initialization.

Execution Parameters

Execution parameters come from one or more of the following sources (listed in order of priority, highest first):

1. operator replies (if allowed by the OPI parameter)
2. PARM parameter in the EXEC statement in the JCL that initializes Multi-User
3. S2KPARMS data set
4. system defaults.

For details about setting execution parameters, see **Setting SYSTEM 2000 Execution Parameters** on page 5-9. Refer to **Multi-User Execution Parameters** on page 3-49 for details about Multi-User execution parameters.

SCF TP with Multi-User Software

SCF TP can be brought up during Multi-User initialization. Also, the console operator can cancel or initialize SCF TP while Multi-User is active (see **Multi-User Console Operator Commands** on page 3-65). The TPTHREADS execution parameter determines how many active SCF TP users will be allowed to sign on during a Multi-User session.

Consider the following guidelines if SCF TP is going to be active:

- If you set the STARTTP execution parameter to YES during Multi-User initialization, SCF TP is available immediately. If STARTTP=NO, the console operator can initialize SCF TP later, and he can cancel SCF TP at any time. For details, see **Multi-User Console Operator Commands** on page 3-65.
- An 11592 pool size is used for S2KUSERS file I/O.
- The S2KUSERS file needs enough disk space to handle the maximum SCF TP run-units (TPSCRUN parameter) plus one. With the default TPSCRUN=180, S2KUSERS must have 181 blocks. If you have not already allocated the S2KUSERS file, SYSTEM 2000 software will dynamically allocate the file. For details, see **S2KUSERS File** on page 3-30.

Requirements for Executing Multi-User Software

The following rules pertain to executing Multi-User:

- If you use IBM Cross Memory Services (XMS) with Multi-User, multiple Multi-User systems can be active at the same time. See **Cross Memory Services (XMS) Multi-User Software** on page 3-16.

  Only one copy of non-XMS Multi-User per Type 2 SVC can reside in memory at a time. If you need more than one non-XMS Multi-User, you must have more than one Type 2 SVC, and you must rename S2KCOPY so each version has a unique name. See the installation instructions for Release 12.0 of SYSTEM 2000 software for details about renaming S2KCOPY.

- Multi-User begins honoring requests after message S2K1117 MULTI-USER INITIALIZATION PHASE COMPLETE is displayed. User jobs abend with user abend 516 if Multi-User is not available.

After Multi-User software is initialized in its own region, it waits for a request for its resources from either a batch SCF job, an SCF TP user, or a SYSTEM 2000 PLEX program. The console operator can display the status of jobs and databases via console commands (see **Multi-User Console Operator Commands** on page 3-65).

JCL for Initializing Multi-User Software

Sample JCL for initializing and executing Multi-User software is provided in the installation instructions for Release 12.0 of SYSTEM 2000 software.

Alternate User Files

If any SCF TP users will require alternate user files for their jobs (Command Files, Data Files, Message Files, or Report Files), the user's new file names must be part of the Multi-User region and must be specified during Multi-User initialization. The standard file name INPUT is the default for the Command File and the Data File; it does not need to be specified. The standard file name OUTPUT is the default for the Message File and the Report File; it does not need to be specified. When the default INPUT and OUTPUT files are used, input is read from the SCF TP terminals, and output from SYSTEM 2000 software is returned to the terminals automatically. For information about dynamically allocating user files, see *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition*.

To avoid conflict, alternate file names should be unique for each user. DISP=MOD can be specified if several reports on the same file are to be concatenated. The batch interface, SYS2KJOB, can be run in batch or TSO and allows user files to be in that user's region, not the Multi-User region.

Also, for CMS and TSO, local alternate user files can be used if the LOCAL option is selected. See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition*.

S2KUSERS File

The S2KUSERS file is a BDAM file initialized by SYSTEM 2000 software for SCF TP. The S2KUSERS file holds information about all current TP run-units. Thus, the TP user does not need to enter a password, database name, or options for each activity in the session. You can let SYSTEM 2000 software dynamically allocate the S2KUSERS file, or you can allocate it in the Multi-User initialization step.

Dynamically allocating the S2KUSERS file The following rules pertain to dynamic allocation of the S2KUSERS file:

1. If the file is already allocated, the software uses that file and dynamic allocation does not occur.
2. If an S2KUSERS file with the data set name of *prefix*.S2KUSERS exists and can be allocated with DISP=OLD, the software uses that file.
3. If the S2KUSERS file does not exist, SYSTEM 2000 software dynamically allocates a temporary S2KUSERS file, and the space for the allocation is (CYL,(5,1)).

Allocating the S2KUSERS file with JCL This topic gives guidelines for allocating the S2KUSERS file in the Multi-User initialization step. Note that when the SCF TP facility is needed, TSO with the Self-Contained Facility (SYS2KTPI) is considered a TP user.

The maximum number of concurrent TP run-units is 10,000. The default is 180; in the following example, 181 represents the default plus one.

```
//S2KUSERS DD UNIT=SYSDA,SPACE=(11592,(181))
```

The S2KUSERS file requires a block size of 11592 and enough blocks to handle the number designated in the TPSCRUN execution parameter plus one (that is, from 2 to 10,001). With TPSCRUN defaulting to 180, 181 S2KUSERS blocks are needed. If you specify TPSCRUN=500, the S2KUSERS file needs space for at least 501 blocks of 11592 bytes.

Scratch Files for Multi-User Software

Because scratch pads are required by Multi-User, no DD statements are given for Locate Files, Collect Files, S2KSYS, or S2KUS files. However, the sort files are required for each thread. See Chapter 4, "Specifying Files and Buffers," for details.

The Savefile and the Keepfile

The Savefile is used to save and restore a database. The DD statement for this file can be included in the JCL for Multi-User initialization. Or it can be allocated dynamically with options in the SAVE command or by the software, using defaults. The DDname for the Savefile is the first seven characters of the database name plus a suffix of S.

The Savefile is a QSAM file, which can be disk or tape. You can specify the number of QSAM buffers for save and restore processing with the DBBUFN execution parameters (see **DBBUFN Parameter: Altering QSAM Buffers for SAVE/RESTORE** on page 5-13). The DDBUFN execution parameter pertains to buffers for database files. To specify buffers for the Savefile you must use JCL DCB= BUFNO=*nn*.

If the Rollback Log is enabled for a database, the Update Log is required. The Update Log file is treated as a BDAM database file, File 7. Therefore, the database file block size can be specified like any other database file. You can dynamically allocate the Rollback Log and the Update Log.

The Keepfile can be used to hold Update Log records for a database on a permanent file (tape or disk). If the user wants to issue a KEEP command during a Multi-User session and the Keepfile has not been allocated, SYSTEM 2000 software will dynamically allocate it.

For more information about the Savefile, Keepfile, Update Log, and Rollback Log, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

Often the SAVE, RESTORE, KEEP, and APPLY commands accessing these files occur only at predictable times, such as the end of the day. If so, it is more efficient to use single-user jobs to do the maintenance functions after the Multi-User session is completed.

Activating the Accounting Log

If the Multi-User Accounting Log is used, the JCL necessary for the Accounting Log files must be added to the Multi-User step. Also, the ACCT execution parameter must be set to YES. For details of using the Accounting Log, see **Obtaining Resource Statistics: Accounting Log** on page 3-103.

For operating systems that allocate memory in variable regions, such as MVS, the REGION parameter must be increased by the size of the accounting system.

Activating the Diagnostic Log

The Diagnostic Log is written to the data set specified by the S2KDIAG statement. This data set records Multi-User activity and can be written to printer, disk, or tape. The amount and content of output is controlled by the LOGCOUNT and LOGLEVEL parameters. If the Diagnostic Log is written to disk or tape, the LOGDUMP utility can be used to list its contents. Also, the DIAG2000 utility can be used to analyze the Diagnostic Log. For further details about the Diagnostic Log, see **Analyzing Multi-User Events: the Diagnostic Log** on page 3-125.

BATCH SCF JOBS WITH MULTI-USER SOFTWARE

JCL for Batch SCF Jobs

Sample JCL to run a batch SCF job that accesses Multi-User is shown below. Most of these files can be allocated dynamically. For additional sample JCL and procedures, see the installation instructions for Release 12.0 of SYSTEM 2000 software.

```
//S2KJOB      EXEC  PGM=SYS2KJOB
//STEPLIB     DD    DSN=S2K.R120.LOAD,DISP=SHR
//S2KCOM      DD    DSN=S2K.R120.COM,DISP=SHR            (if XMS)
//S2KMSG      DD    SYSOUT=A,DCB=(BLKSIZE=1320,LRECL=132,RECFM=FBA)
//S2KSNAP     DD    SYSOUT=A,DCB=BLKSIZE=882
//SYSUDUMP    DD    SYSOUT=A
//S2KCOMD     DD    *
      .
      .
  (SCF commands)
      .
      .
/*
```

The DDname for the input Command File is S2KCOMD. The DDname for the output Message File and Report File is S2KMSG. Users can assign alternate user files as long as the batch SCF job JCL contains the corresponding DD statements. See *SYSTEM 2000 CONTROL Language, Version 12, First Edition* for more details about alternate user files. SYSTEM 2000 software will dynamically allocate the S2KMSG and S2KSNAP files if no DD statements exist in the JCL. S2KCOMD is allocated to the TSO terminal if it is not specified in the CLIST.

SYS2KJOB is initialized as any normal MVS job step task. During the initialization phase, SYS2KJOB establishes communication with SYSTEM 2000 software by issuing a call to the SVC. If Multi-User has not yet communicated its presence to the SVC, SYS2KJOB is informed via the SVC, and SYS2KJOB terminates with a user completion code of 516 (hex 204).

SYS2KJOB can be run in the TSO foreground, which allows dynamic allocation of user files in the user's TSO region and interactive access to SYSTEM 2000 software. However, there are several disadvantages. The SYS2KTPI interface is much more efficient at sharing Multi-User resources.

Batch SCF Job Execution

Batch SCF jobs (SYS2KJOB) execute in their own region and communicate with Multi-User through the SVC. Multi-User asks SYS2KJOB to read the user's Command File. SYS2KJOB then reads the Command File one record at a time and sends the contents of the record to Multi-User. Output for each command is returned to SYS2KJOB one line at a time.

User exits can be used to modify input or output during batch SCF job execution. For details about specific user exits, see Chapter 6, "User Exits."

SCF Batch Segments

A batch SCF segment is a series of contiguous commands issued in one batch SCF user job. Multi-User may or may not process a segment before considering requests from any other user, depending on resources and polling position. A user-specified SCF segment begins with the $LOCK command and ends with the $END or EXIT command. A segment ties up one thread. If $LOCK and $END are not used, an SCF segment is defined as follows:

- one QUEST language command; ties up one thread for the duration of the command.
- a queue mode session (that is, all commands between QUEUE and TERMINATE); ties up one thread. Retrievals in other threads can be processed concurrently until the TERMINATE command processes any updates given in the queue mode session.
- a CONTROL language session; ties up one thread until the user leaves the CONTROL processor.
- a DEFINE language session; ties up one thread until the user leaves the DEFINE processor. Exclusive use is required.
- a REPORT language session (that is, all commands between REPORT and GENERATE); ties up one thread, allows concurrent retrieval.

Multi-User schedules a batch SCF segment for processing in turn under non-exclusive mode unless the command requires exclusive use. If exclusive use is required for a particular command and the user has not requested exclusive use, Multi-User issues an error message. If exclusive use has been requested and the database is not available, the batch SCF user request is placed in a WAIT state.

Terminating a Batch SCF Job

A batch SCF job terminates when SYSTEM 2000 software finds an EXIT command or an end-of-file on the Command File. However, the databases used during the job may still be in use by other jobs. Individual databases and other files allocated in the Multi-User JCL are coordinated by Multi-User and not by the individual SCF job.

The console operator can cancel a specific batch SCF job running under Multi-User without the risk of damaging the user's database(s). The user's job is allowed to finish the current command if it is updating a database. Otherwise, cancellation is immediate. An operator cancel causes a corresponding user abend code.

User exits can be used to collect statistics or standardize the shutdown of a batch SCF job. For a description of specific user exits, refer to Chapter 6, "User Exits."

SCF TP JOBS WITH MULTI-USER SOFTWARE

SCF TP Segments

An SCF TP segment is an SCF command or a series of SCF commands with a maximum of 1200 characters. The segment begins with the first character transmitted and ends with the character that signals message completion (carriage return, EOB, EOT, and so forth) to the host computer. The message completion character is device-dependent and, in some cases, operating system-dependent.

One thread is tied up for the duration of an SCF TP segment.

The following list describes rules that apply to SCF TP segments:

- An SCF command (or several SCF commands) must be completed in one SCF TP segment. If not, a message indicates a missing carriage return.
- An SCF TP segment can contain up to 1200 characters. If this limit is exceeded, a message indicates a missing carriage return.
- The Command File can be switched to an alternate file in an SCF TP segment. The COMMAND FILE IS command must be the last command in the SCF TP segment; if not, results are unpredictable. The 1200-character segment limit does not apply to an alternate Command File; the entire alternate file is processed. The referenced Command File must be defined by a DD statement in the JCL of the Multi-User initialization job. For TSO and CMS, the LOCAL option can be used for local alternate files, declared in the dependent region.
- If the DITTO execution parameter equals YES, the DITTO operator can be used across SCF TP segments. $LOCK is not required. If the DITTO execution parameter equals NO, the DITTO operator can refer only to an action-clause that is specified in the current SCF TP segment. Otherwise, Multi-User issues an error message.
- If the SAME execution parameter equals YES, the where-clause results are saved, and the SAME operator can be used across SCF TP segments. $LOCK is not required. If the SAME execution parameter is NO, the SAME operator can refer only to a where-clause that is specified in the current SCF TP segment. Otherwise, Multi-User issues an error message.
- DEFINE, QUEUE, FRAME, and REPORT sessions cannot span SCF TP segments. That is, the MAP, TERMINATE, END FRAME, or GENERATE commands, respectively, must be in the current SCF TP segment. (These types of requests are best handled using an alternate Command File.) A DEFINE language session reverts to the CONTROL processor. If a frame is not completed, the database is marked damaged. This triggers Coordinated Recovery (if the Rollback Log is enabled) when the next user accesses the database. See the *SYSTEM 2000 CONTROL Language, Version 12, First Edition* for details about Coordinated Recovery.

SCF TP Input and Output

One SCF TP interface is available for TSO and CMS users. The interface SYS2KTPI sends blocks of commands to Multi-User to be processed as a segment. As long as the segment lasts, the user job remains in a Multi-User thread. During segment processing, Multi-User sends output to the interface in blocks of 4096 bytes.

SYS2KTPI interface The SYS2KTPI interface also does not require user file allocations. However, this interface can use the LOCAL option to specify alternate user input and output files. If a user wants the LOCAL option for alternate files, those files must be allocated in the user address space before the SYS2KTPI interface program is executed. See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for more details about local and central alternate files.

For the SYS2KTPI interface, an intermediate output file (S2KOUTP) can be allocated to hold output destined for the terminal. The S2KOUTP file must be large enough to accommodate all the output for an SCF TP segment. It must not be allocated to the terminal. If you allocate this file, these constraints apply:

- It must be a QSAM file with a fixed-block format and a block length of 4096 bytes.
- It must be named S2KOUTP.
- It must not be allocated to the terminal.

If you do not allocate S2KOUTP, SYSTEM 2000 software dynamically allocates a temporary file. The space information will be (CYL,(1,1)). The value of the PREFIX execution parameter is used if specified (see **PREFIX execution parameter** on page 4-17). If the file exists with a data set name of *prefix*.S2KOUTP and can be allocated with DISP=OLD, the software uses that file.

Command Stacking in a TSO CLIST

TSO users can stack SYSTEM 2000 commands within a CLIST and send them to SYSTEM 2000 software for processing. The ENDSTACK statement allows the SYS2KTPI interface to recognize the end of the stacked input and let any additional TSO commands execute after the interface terminates.

When used as a TSO command, the SYS2KTPI interface for TSO uses GETLINE processing, which lets you build and access a stack of SYSTEM 2000 commands within a CLIST. A stack is created when you execute a CLIST that contains a DATA and ENDDATA sequence. The TSO command parser ignores all statements between the DATA and ENDDATA statements; they are placed in a command stack, which is passed directly to SYSTEM 2000 software.

Since SYS2KTPI uses GETLINE logical processing, you can use continuation characters to generate an SCF TP segment having any number of characters. For example, you can give all the SYSTEM 2000 commands necessary for a report writing session. Logical processing pertains only to commands entered into a stack or entered from a terminal.

How to invoke TSO stack processing To use the TSO stacking feature, you must invoke SYS2KTPI as a command. Do not use the CALL command to invoke SYS2KTPI.

TSO has both commands and command procedures. (A CLIST is a command procedure.) In order for you to invoke the SYS2KTPI interface as a command, it must exist in a load library accessible to your TSO userid. That is, a STEPLIB DD statement must point to it in a library, or it must be in the system LINKLIST concatenation.

The command name is the member name that you assigned when you stored the SYS2KTPI interface. Suppose the member name for the interface is SYS2KTPI. To invoke it with a command, enter SYS2KTPI without any prefix.

Example of TSO command stacking The following example shows an SPF application that starts with a master menu named CSR@MSTR in the ISPF Panel Library. Normally, the SPF main menu references a master menu that users can install for in-house applications. The following example was installed as a sample application from the master menu, so that access in SPF was obtained by using a path of M.CSR.

Example of SPF Application

```
%------------------------  CSR APPLICATION MENU  ------------------------
%SELECT APPLICATION ===>_ZCMD        +
%                                                        +USERID    - &ZUSER
%                                                        +TIME      - &ZTIME
%                                                        +TERMINAL  - &ZTERM
%                                                        +PF KEYS   - &ZKEYS
%
%   B +BPART      - add/modify parts in the PARTS database
%
%   P +VIP        - add/modify VENDOR Information Profiles
%
%   A +ACCESS     - generalized database access
%
%
%
%
%
%
%
%
%
%   C +CONSOLE    - talk to MU/MT via TSO ALTCON program
%
+Press the%END+key to terminate this menu +
)INIT
  .HELP    = BPRT0001     /* HELP for this master menu              */
)PROC

  &XQ1 = TRUNC (&ZCMD,'.')
  &ZSEL = TRUNC (&ZCMD,'.')
  &ZTRAIL = .TRAIL

  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
                B,'CMD(%BUGSHEET)'
               BP,'CMD(%BUGSHEET)'
              BPA,'CMD(%BUGSHEET)'
                V,'CMD(%PIPUPDTE)'
               VI,'CMD(%PIPUPDTE)'
              VIP,'CMD(%PIPUPDTE)'
                A,'CMD(%ACCESSDB)'
              ACC,'CMD(%ACCESSDB)'
           ACCESS,'CMD(%ACCESSDB)'
                C,'CMD(%SYS2KOPR)'
              CON,'CMD(%SYS2KOPR)'
          CONSOLE,'CMD(%SYS2KOPR)'
                X,'EXIT'
              ' ',' '
                *,'?' )
)END
```

If you enter A, ACC, or ACCESS on the command line of the CSR menu shown on the previous page, the ACCESSDB CLIST below is executed.

Notice that other TSO statements follow the DATA/ENDDATA group. Therefore, you must have an ENDSTACK statement after the last SYSTEM 2000 statement to be executed before TSO is to read from the terminal. If you omit the ENDSTACK statement, the END, CONTROL, and FREE commands are all passed on to SYSTEM 2000 software.

```
PROC 0 TRACE
   IF &TRACE = THEN CONTROL LIST SYMLIST CONLIST MSG NOFLUSH
   ELSE  CONTROL MSG NOLIST NOCONLIST NOSYMLIST NOFLUSH
CONTROL NOMSG
ALLOC FI(S2KOUTP) SPACE(1,1) CYL
CONTROL MSG
ALLOC FI(S2KCOM) DA('your-S2KCOM-file') SHR REUSE
ISPEXEC DISPLAY PANEL(ACC0001)
DO WHILE &LASTCC NE 8
   SYS2KTPI
   DATA
      USER,X:ECHO ON:
      &ACCCMD
      ENDSTACK
   ENDDATA
   ISPEXEC DISPLAY PANEL(ACC0001)
END
CONTROL NOMSG
FREE FI(S2KCOM S2KOUTP)
```

The ACCESSDB CLIST displays the panel shown on the next page. This panel lets you select databases and create SYSTEM 2000 commands.

This standard formatted ISPF panel shows how to select ten databases by name and translate the numeric selection code into a SYSTEM 2000 command.

ISPF Database Access Panel

```
)ATTR
 % TYPE(TEXT) INTENS(HIGH) SKIP(ON)
 + TYPE(TEXT) INTENS(LOW) SKIP(ON)
 _ TYPE(INPUT) INTENS(HIGH) SKIP(OFF)
)BODY
%------------------------ CSR General DB Access ----------------------
%COMMAND ===>_ZCMD
%
+ Select database from list below: %Database Code +===>_D %
+
%   1+---> %PARTS          + You can now access a database with the new
%   2+---> %VENDOR         +   SYS2KTPI, which allows the use of stacks
%   3+---> %SPECIAL CUST   +   via the DATA and ENDDATA statements.
%   4+---> %CUSTOMERS      +
%   5+---> %EMPLOYEE       +
%   6+---> %LIBRARY        +
%   7+---> %PERSONNEL      +
%   8+---> %MANUALS        +
%   9+---> %DOCS           +
%  10+---> %CPROB          +
%
%
%
%
%
%
%
+Press%ENTER+to continue or %END+to terminate
)INIT
 &ZCMD = ' '
 .CURSOR = D
)PROC
 VER (&D,RANGE,1,10)
 &ACCCMD = TRANS(&D,7,'USER,pwd1:DBN IS PERSONNEL:'
                     1,'USER,pwd2:DBN IS PARTS:'
                     2,'USER,pwd3:DBN IS VENDOR:'
                     3,'USER,pwd4:DBN IS SPECIAL CUST:'
                     4,'USER,pwd5:DBN IS CUSTOMERS:'
                     5,'USER,pwd6:DBN IS EMPLOYEE:'
                     6,'USER,pwd7:DBN IS LIBRARY:'
                     8,'USER,pwd8:DBN IS MANUALS:'
                     9,'USER,pwd9:DBN IS DOCS:'
                    10,'USER,pwda:DBN IS CPROB:'
         )
)END
```

Shutting Down an SCF TP Run-unit

Individual databases and other files allocated in the Multi-User session are coordinated by Multi-User and not by the individual SCF TP job.

The console operator can cancel a specific SCF TP job running under Multi-User without the risk of damaging the user's database(s). The user's job is allowed to finish the current command if it is updating a database. Otherwise, cancellation is immediate. An operator cancel causes a corresponding user abend code.

User exits can be used to collect statistics or standardize the shutdown of an SCF TP job. For a description of specific user exits, refer to Chapter 6, "User Exits."

PLEX JOBS WITH MULTI-USER SOFTWARE

JCL for PLEX Jobs

The JCL for executing PLEX programs under Multi-User is documented in the *SYSTEM 2000 PLEX Manual, Version 12, First Edition.* Multi-User initialization is independent of individual PLEX jobs.

Initializing a PLEX Job

If the PLEX program is initialized as a normal job step task or TSO on the first execution of a SYSTEM 2000 command, the interface routine linked with the user's program brings a routine named MUPLINT into memory via the LOAD macro. MUPLINT issues requests to Multi-User via the SYSTEM 2000 SVC and routes results back to the PLEX program. Also, MUPLINT traps all abend conditions in the PLEX program. A "graceful" termination of the PLEX job informs Multi-User so that the job status remaining in Multi-User can be cleaned up. MUPLINT is also sensitive to Multi-User termination. If Multi-User has not yet communicated its presence to the SVC, MUPLINT is informed via the SVC, and the PLEX program ends with an appropriate user completion code.

If the PLEX program is initialized as a TP transaction, the appropriate PLEX TP interface routine is linked with the user's program and passes control to the TP monitor being used, for example, CICS.

Execution parameters Multi-User execution parameters are set for an entire Multi-User session and not for each individual PLEX job. For example, buffer pools are allocated when Multi-User is initialized and not when a PLEX job starts.

USE option Within a PLEX program, you can set the USE option in the START S2K command. This allows a run-time parameter to determine whether a compiled and linked PLEX program accesses Multi-User or single-user SYSTEM 2000 software. In this way, the same executable load module can be run in either single-user or Multi-User mode without recompilation.

An example of the USE parameter follows:

```
      .
      .
      .
      1    MU-FLAG PIC 9(9) COMP SYNC VALUE 0.
      .
      .
      .
    LINKAGE SECTION.
     1   PARM-INFO.
         05  PARM-COUNT              PIC S9(4) COMP SYNC.
         05  PARM-DATA               PIC 9.
             88  PARM-MULTI-USER                VALUE 1.
             88  PARM-NOT-MULTI-USER            VALUE 0.
    PROCEDURE DIVISION USING PARM-INFO.
    IF (PARM-MULTI-USER OR PARM-NOT-MULTI-USER) AND
    PARM-COUNT NOT = ZERO
    THEN MOVE PARM-DATA TO MU-FLAG.
    NOTE THAT THE DEFAULT IS 0 FOR SINGLE-USER.
      .
      .
      .
    START S2K(1) USE (MU-FLAG).
      .
      .
      .
     //   EXEC S2KSUJOB,PRGM='PRGABC,PARM=0'
    .  (user database DD statements, scratch files, and so on
    .     for a single-user job)
    .
     //   EXEC S2KMUJOB,PRGM='PRGABC,PARM=1'
    .  (no DD statements for database files or scratch files
    .    for Multi-User jobs)
```

PLEX Job Execution

This section covers using the HOLD option during PLEX updating, acquiring a Multiple Local Holds (MLH) buffer, freeing space in memory and disk by discarding Locate Files, and employing user exits in a PLEX program.

HOLD option The PLEX HOLD option reduces database contention for PLEX update processing under Multi-User. It protects a record from other users' updates, but allows other users to retrieve data from the database. The HOLD option is appropriate when the database is being updated under non-exclusive use. It protects data integrity and currency during updates, without requiring exclusive use of the database. A HOLD is meaningless under exclusive use.

The HOLD option can be used with any PLEX retrieval command, that is, GET/HOLD, GET1/HOLD, GETA/HOLD, GETD/HOLD, and GETP/HOLD. All local holds obtained with the HOLD option are retained until a synchpoint occurs, for example, COMMIT. See the discussion of Coordinated Recovery in *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

Although the HOLD option is not mandatory, it should be considered for all Multi-User PLEX update situations. For example, the HOLD option should always be used when the data value to be modified is a function of its current value, for example, to add 40 hours to TOTAL-HOURS. Suppose User 1 and User 2 happened to retrieve the same data record with the intention of updating the data during the next time slice. Then, suppose User 1 modifies or removes data during his next turn. User 2 is unprotected, because data in his subschema record perhaps have changed or have been removed from the database since his last turn.

Also, the LHOLD=YES execution parameter can be used to require all PLEX update requests to be preceded by a retrieval command having the HOLD option. LHOLD=YES is ignored if the Rollback Log is enabled, if the database is under exclusive use, or if a global hold is in effect. For more information about the LHOLD parameter, see **LHOLD Parameter** on page 3-58.

Note: SCF sessions and single-user PLEX jobs do not need local holds. If the PLEX HOLD option is specified, the software ignores it.

Multiple Local Holds (MLH) buffer *Local hold* means that a specific record is held by a PLEX program in order to update the record without interference from other users. A local hold occurs when you issue the HOLD option for an immediate access retrieval command, for example, GET1/HOLD. Also, local holds occur automatically for each record that is updated when Coordinated Recovery is enabled in a Multi-User environment. Coordinated Recovery automatically holds any record that a PLEX program

- modifies with the MODIFY or REMOVE command
- inserts with the INSERT TREE command
- deletes with the REMOVE TREE command.

Note: For INSERT TREE and REMOVE TREE, the node of attachment (top node) is held until the insertion or removal is completed.

To place a local hold on a particular record, SYSTEM 2000 software enters the Hierarchical Table (HT) address in the local hold buffer for that database. The HT address stays in the buffer until some command, such as COMMIT or DROP HOLDS, causes the local hold to be removed. If multiple local holds are disabled, each PLEX user program can hold only one record in a database at a time; when a new hold is invoked, the previous hold is dropped.

The ENABLE MULTIPLE HOLDS command allows a large number of records to be held by each user for each database. (See *SYSTEM 2000 CONTROL Language, Version 12, First Edition*). The number of HT addresses (local holds) that can be stored in an MLH buffer is (buffer size - 64) / 5. Sixty four bytes are subtracted from the buffer size because the first eight bytes contain header information and 56 additional bytes are used to prevent buffer manager conflict. Each local hold entry consists of five bytes (four bytes for the HT address and one for the userid). A buffer size of 23646 can hold 4680 local hold entries. Each database that has multiple local holds enabled will have one MLH buffer opened.

Prior to Version 12, SYSTEM 2000 software assigned MLH buffers to the pool that had the largest buffers with database usage. Sharing the pool with database pages caused contention. Also, the user had little or no control over the MLH buffer size for any database. The maximum buffer size was 23464.

With Version 12 and later releases, you can specify the MLH buffer size for a specific database by using user exit EXIT07. EXIT07 occurs immediately after the software does a physical open for the database. You can store the required MLH buffer size in the storage area AMHBUFSZ, which is used only by user exits. SYSTEM 2000 software picks up the size from that storage area. A sample exit is provided on the delivery tape; you only need to enter database names and buffer sizes.

The range of sizes is 512 to 50,000 bytes. If you specify less than 512, the software changes it to the minimum of 512. Similarly, if you specify a larger number, the software changes it to the maximum of 50,000. This range allows for 89 to 9,987 local holds (HT addresses) to be held in each MLH buffer.

For MVS/XA and VM/XA systems, the MLH buffer is always requested with a GETMAIN using LOC=ANY,ANY, which means the buffer will be in 31-bit addressable memory, above the 16 megabyte line. If a size was specified with user exit EXIT07, that size is used. If a size was not specified, the default is 23464 bytes.

For non-MVS/XA systems, the MLH buffer is obtained with a GETMAIN only if a size was specified with user exit EXIT07. If the GETMAIN fails or a size was not specified, the buffer is allocated from the pool with the largest buffer size for database usage. For non-XA CMS systems, the MLH buffers are always allocated from the pool.

Discarding Locate Files Discarding Locate Files frees disk space and buffers in memory. This is especially significant in a Multi-User environment where several users, each using several Locate Files, can cause a deadlock. Discarding a Locate File invokes little overhead and should be considered for all PLEX programs. When a Locate File is discarded at an appropriate time, the advantages are significant.

To discard a Locate File, the programmer specifies a LOCATE command that qualifies no data records. For example,

```
LOCATE SCHA WHERE item EQ nonexistent-value
```

A Locate File retains one scratch file buffer from a pool as long as the Locate File exists. However, when the user discards a specific Locate File, the scratch file buffer is released immediately. For a single-user job not using scratch pads, any disk space given to the file is retained to the maximum length that the file has used in the program execution. Multi-User requires scratch pads, so Locate Files do not retain their maximum lengths. However, when a Locate File is no longer needed, it should still be freed in order to free space on scratch pads.

PLEX user exits PLEX user exits can be used to modify input or output during a Multi-User PLEX job execution. For a description of specific user exits, refer to **Execution-Time PLEX Exits** on page A-15.

PLEX Segments

A PLEX segment is a single PLEX command; it ties up a thread for the duration of the command. The HOLD option does not begin a segment, nor does the PLEX QUEUE command. A PLEX user cannot define a segment to be more than one command.

Shutting Down a PLEX Job

The PLEX program should always close each database with a CLOSE command. After all databases are closed, the STOP S2K command should be issued. This command signs the user off properly, and the software does not need to go through error trapping. Thus, for TP mode, the STOP S2K command must be issued before the end of the transaction.

If the STOP S2K command is not issued and the job or transaction terminates, SYSTEM 2000 error recovery gets control and terminates the user from Multi-User. This causes additional overhead because the software must go through error recovery code.

Operator cancellation of a PLEX job To cancel a specific Multi-User job, the console operator gives a MODIFY command. See Console Operator Commands discussed in **Multi-User Console Operator Commands** on page 3-65.

Job names and the status of a job are obtained by one of the DISPLAY commands. The console operator can cancel a specific PLEX job without the risk of damaging the user's database(s). The PLEX job is allowed to finish the current command if it is updating a database. Otherwise, cancellation is immediate. An operator cancel causes an appropriate user abend (or PLEX return code for TP) to be issued. This is true for any job, including PLEX running in TSO foreground.

When the job is terminated with a user abend code, the PLEX job does not get back control to properly close non-database files in the dependent region.

SHUTTING DOWN MULTI-USER SOFTWARE

Normally, Multi-User can be shut down with a console operator CANCEL command. In a teleprocessing environment, this command can be issued from authorized terminals that simulate the master operator console.

The F S2K,CANCEL S2K command has three options: WHEN, NOW, and FORCE. WHEN terminates Multi-User after all users have exited the system. NOW allows jobs that are updating the database to complete their processing. When all jobs complete, Multi-User is terminated. **Note**: The FORCE option cancels Multi-User immediately without doing any clean-up. Try NOW and WHEN before using FORCE, because FORCE could cause damaged databases.

The CANCEL command also can be used to cancel SCF TP. For complete details about canceling Multi-User and SCF TP see **Terminating Multi-User Software or SCF TP** on page 3-77 and **Terminating Multi-User Software When SCF TP Is Active** on page 3-78.

If you are using IBM Cross Memory Services (XMS) with Multi-User, be sure to read about terminating Multi-User in **Cross Memory Services (XMS) Multi-User Software** on page 3-16, especially about the results of the FORCE option.

ABENDS IN THE MULTI-USER ENVIRONMENT

With Release 11.6 and later releases, ESTAE macros replace STAE macros. SYSTEM 2000 software checks the system type in order to handle differences in XA and non-XA ESTAE macro expansion.

With ESTAEs, all abends, including XMS-related abends, are trapped in the same manner. ESTAEs also offer the following advantages:

- SYSTEM 2000 software cannot recover (that is, cannot continue execution) from X22 type abends, such as operator cancel, time limit exceeded, and so on; however, X22 type abends are trapped. If the X22 abend occurs in an interface routine, Multi-User is informed that the user is terminating. If the X22 abend occurs in the Multi-User region, Multi-User issues a message stating that it was canceled by the system. If the XMS version of Multi-User is being used, CLEARS2K is run before terminating. If you are using the SVC version of Multi-User, you must run CLEARS2K.

- ESTAEs percolate up to higher levels of ESTAEs for all PLEX programs. At the higher level you can use error trapping techniques or a debugging package to further handle any error.

If the STAE execution parameter equals NODUMP and an abend occurs, ESTAEs are in effect but no dump is taken. STAE=NODUMP is especially good for COBOL II and PL/I PLEX programs that utilize debugging aids and, therefore, do not need a SYSTEM 2000 dump. Normally, you would not set STAE=NODUMP for production Multi-User jobs; that is, for Multi-User abends you would want to take a dump.

ESTAEs also pertain to subtasks, such as restoring a database or saving a database with the QSAM format. Subtasks do not produce a dump for X13, X37, and X78 abends. These abend codes are trapped and SYSTEM 2000 software issues an appropriate message. All other abends cause a dump to be taken unless the STAE execution parameter equals NODUMP. If

you specify STAE=NO. the system produces a dump for all abends. STAE=NO should be the rare exception, since SYSTEM 2000 software does not receive control and cannot prevent damaged databases when an abend occurs

In addition, if you include an S2KSNAP DD statement in the interface JCL (or if one is provided dynamically by SYSTEM 2000 software), a dump is taken of the SYSTEM 2000 interface routines and region when an abend occurs.

SYSTEM 2000 software formats the Extended Specify Task Abnormal Exit (ESTAE) work area if Multi-User abends and STAE=YES. The formatted work area appears on the Diagnostic Log. See **Format of Diagnostic Log Records** on page 3-144 for an example of the formatted ESTAE work area. Also, see **STAE Parameter: Enabling Error Trapping** on page 5-14 for a discussion of the STAE execution parameter, ESTAEs, error trapping, and the S2KSNAP file.

The following two messages appear before the ESTAE work area:

```
-881- SYSTEM 2000 ABEND PROCESSING INVOKED -
-882- STAE WORK AREA FOLLOWS -
```

The next nine messages contain the completion code, Problem Program PSW, program name, entry point address, abend PSW, and R0 through R15 contents at the entry to the abend. Each item is labeled.

If the ESTAE work area is unavailable, the following message appears:

```
-883-STAE WORK AREA NOT AVAILABLE FOR FORMATTING -
```

In the Multi-User environment, if the ESTAE work area cannot be formatted (because it is not available), it is provided as the first snapshot during abnormal job termination. The format of this portion of memory is as follows:

Fullword #	Description
1	binary zeros (not used)
2	interruption code
3-4	problem program PSW
5-6	abend PSW
7-22	R0-R15, respectively
23-24	program name (always SYS2K)
25-26	EPA
27	binary zeros (not used)

If the STAE execution parameter equals YES (default), Multi-User traps all abends. See the next subtopic for how the software handles S522 abends.

If the error occurs in a user job, that job is terminated and Multi-User continues. If the error occurs in Multi-User, a full snapshot is written to the S2KSNAP file, except for X22 abends. The normal start-up procedure can be used to reactivate Multi-User. All user jobs accessing SYSTEM 2000 software must be canceled, and CLEARS2K is run automatically for XMS; however, if you are using the Type 2 SVC version of Multi-User, you must run CLEARS2K.

If Multi-User abends or is canceled by the operator, CLEARS2K must be executed. (See **Clearing the Multi-User SVC: CLEARS2K** on page 3-26.) If Multi-User is canceled (S122, S222) or runs to maximum time limit (S322), all jobs using Multi-User are terminated by their respective Run-unit Controllers (or should be canceled by the operator).

Multi-User abend codes that can occur are described in *SYSTEM 2000 Messages and Codes, Version 12, First Edition.*

Preventing a Multi-User S522 dump When Multi-User is initialized, it attaches a subtask (N0522) immediately. This subtask issues an STIMER every five minutes. Therefore, if Multi-User has no other activity, subtask N0522 ensures that Multi-User will not be brought down by exceeding the MVS WAIT time period (usually 15 to 30 minutes). Subtask N0522 does nothing but sleep between the five minute intervals.

Issuing an ESTAE under Multi-User software For Multi-User, the Run-unit Controller interface intercepts all calls to SYSTEM 2000 software. The first use of a SYSTEM 2000 command causes the issuing of an ESTAE, and the SVC establishes communication with the SYSTEM 2000 region. The interface constructs specific SYSTEM 2000 requests, waits for SYSTEM 2000 completion, and handles special return codes, for example, a directive to terminate the user job.

Multi-User issues an ESTAE when it is initialized. If STAE=YES and Multi-User abends, an attempt at graceful shutdown is carried out. If error trapping is not possible, each user program will probably have to be canceled.

MVS error recording The MVS version of the SVC incorporates facilities to log software errors to an SCP log file. This error information should help the Database Administrator and systems personnel in providing support to the Multi-User system. Through use of the SCP SETRP and SETFR macros, information regarding software errors is recorded in the SYS1.LOGREC log file. For additional information, see *OS/VS2 System Programming Library: SYS1.LOGREC Error Recording* GC28-0677 *"Recording Software Errors"* and *OS/VS2 System Programming Library: Supervisor* GC28-0628 *"SETRP Macro Instruction"*.

Various Functional Recovery Routines (FRRs) are used by the SVC. In general, they provide error recording, not error recovery. Each of the FRRs writes the System Diagnostic Work Area (SDWA) to the SYS1.LOGREC log file.

MULTI-USER EXECUTION PARAMETERS

This section discusses the SYSTEM 2000 execution parameters that apply to Multi-User only. Other important parameters that serve both single-user jobs and Multi-User are discussed elsewhere in sections for the specific topics. For example, details for the POOLn and PADn parameters are in the sections about database files and scratch pads, the ACCT parameter is included in the Accounting Log section, and the LOGLEVEL and LOGCOUNT parameters are described in the Diagnostic Log section.

The following table summarizes all Multi-User parameters and gives a specific reference to the main discussion about each parameter. For details about how to specify SYSTEM 2000 execution parameters, see **Setting SYSTEM 2000 Execution Parameters** on page 5-9.

Illus. 3.5 Summary of Multi-User Parameters

Name	Purpose	Page
ALLOC	allows dynamic allocation of database files.	4-16
ACCT	controls writing data to the Accounting Log.	3-110
CONVERT	specifies that a database created under a prior version is to be converted for access under the current version.	5-15
COPYAREAn	used by SYSTEM 2000 software with TSO or a fetch-protected operating system.	3-54
CORECOV	specifies the scope of database recovery under Coordinated Recovery.	5-15
DBBUFN	alters the number of QSAM buffers for database files during save and restore operations.	5-13
DISK	determines the default block size for database files.	4-21
DITTO	controls the use of DITTO in separate SCF TP segments.	3-57
EXITS	directs the dynamic loading of the S2KEXIT user-exit interface.	6-13
FPTPSYS	determines whether system areas are to be obtained for fetch-protected systems for TP access.	3-57
FREE	controls the deallocation of database files.	4-17
LDBS	specifies the initial number of Database Definition Blocks used by large databases.	4-55

continued on next page

Illus. 3.5 *continued*

Name	Purpose	Page
LDBSIZE	specifies the maximum number of components that can be contained in a large database.	4-55
LHOLD	controls local holds in PLEX programs.	3-58
LIST	prints a list of execution parameter settings.	5-13
LOGCOUNT	controls the number of messages written to the Diagnostic Log.	3-133
LOGLEVEL	determines what data are written to the Diagnostic Log.	3-133
NLSEG	determines what data are written to the Accounting Log for SCF batch jobs.	3-133
OPI	determines whether console operator intervention is allowed.	3-58
OPT000	determines how the software checks for the system separator when unloading CHAR, TEXT, and UNDEFINED values.	5-17
OPT001	determines whether SCF commands are echoed.	5-18
OPT002	determines whether a damaged database can be saved.	5-18
OPT003	enables or disables clearing for QUEUE sessions.	5-18
OPT004	includes system-related time in Type 5, 6, and 8 Accounting Log records.	3-112
OPT005	sets the default for the ZERO/ZERO SUPPRESS format option.	5-19
OPT006	sets the default for the NULL/NULL SUPPRESS format option.	5-19
OPT007	sets the default for the REPEAT/REPEAT SUPPRESS format option.	5-19
OPT008	sets the default for the NAME/NUMBER format option.	5-19
OPT009	determines whether database Files 2 through 6 are cleared for RELOAD and RELEASE commands.	5-20
OPT010	sets a limit for records to be updated or retrieved.	5-20
OPT011	sets the maximum number of pages that can be written to the primary scratch file for where-clause processing.	5-20
OPT012	sets the default database block size.	5-20
OPT013	changes the default date format.	5-21

continued on next page

Illus. 3.5 *continued*

Name	Purpose	Page
OPT030	determines whether PLEX stacks and Locate Files are cleared when a logical unit of work is committed.	5-21
OPT031	places the number of user calls in Type 4 Accounting Log records.	3-112
OPT032	sets the default for TALLY to EACH or ALL.	5-21
OPT033	sets the default mode for clearing update pages.	5-21
OPT034	determines whether a hold is required for a QueX update.	3-58
OPT035	determines how SYSTEM 2000 execution parameters are displayed.	5-22
OPT036	places step name and program name in Type 6 Accounting Log records.	3-112
OPT037	places the synchpoint ID in ACCTALT field of user's URB.	3-112
OPT038	is reserved for future use.	
OPT039	determines the precision for floating point data in PLEX programs if padding is necessary.	5-23
OPT040	determines the precision for floating point data in PLEX programs if truncation is necessary.	5-23
OPT041	disables or enables the CMS FINIS command for all users.	5-24
OPT043	disables or enables uppercase translation.	5-24
OPT044	disables or enables the processor prompt feature.	5-24
OPT045	disables or enables the processor lookup feature.	5-24
OPT046	sets the default KEY or NON-KEY status when the user is defining new items in a database.	5-25
OPT047	determines whether validity checking occurs for user-specified packed decimal data.	5-25
PADnn	defines scratch pads.	4-41
PADPRI	alters the default scratch pad primary space.	4-41
PADSEC	alters the default scratch pad secondary space.	4-41
PADSPACE	alters the default scratch pad space unit.	4-41

continued on next page

Illus. 3.5 *continued*

Name	Purpose	Page
PADUNIT	alters the default scratch pad UNIT setting.	4-41
PLSEG	controls what data are written to the Accounting Log for PLEX jobs.	3-110
POOLn	specifies buffer pool attributes.	4-24
PQA	determines the priority queuing algorithm.	3-59
PREFIX	alters the default user prefix.	4-17
RW	controls REPORT processor execution.	5-13
SAME	controls use of SAME in separate TP segments.	3-61
SDBS	specifies the initial number of Database Definition Blocks used by small databases.	4-56
SDBSIZE	specifies the maximum number of components that can be contained in a small database.	4-56
SFPRI	alters the default sort file primary space.	4-38
SFSEC	alters the default sort file secondary space.	4-38
SFSPACE	alters the default sort file space unit.	4-38
SFUNIT	alters the default sort file UNIT setting.	4-38
SID	specifies the System ID.	3-61
STAE	controls error trapping.	5-14
STARTTP	controls execution of the SCF TP facility.	3-61
THREADS	specifies the number of threads that can be executing simultaneously.	3-62
TPSCRUN	specifies the maximum number of SCF TP users.	3-62
TPSEG	controls what data are written to the Accounting Log for SCF TP sessions.	3-110
TPTHREADS	specifies the maximum number of SCF TP active users.	3-63
TSO	specifies that Multi-User is to interface with TSO.	3-63

continued on next page

Illus. 3.5 *continued*

Name	Purpose	Page
USERS	specifies the maximum number of users that can execute SYSTEM 2000 software concurrently.	3-64
XBUF	enables or disables the XBUF caching software.	5-16
XBUFSUF	specifies a suffix that identifies a specific XBUF load module if several exist in the load library.	5-16

COPYAREAn Parameter

COPYAREA*n*=*size*K/*m* *(The default is* **COPYAREA1=1K/4.)**

n the interregion copy area number. The range is 1-6, for example, COPYAREA1, COPYAREA2, and so forth. The default is 1. If you specify multiple COPYAREA*n* parameters, the numbers must be consecutive starting with one.

size the size of each area in K notation, for example, 1K or 2K, where 1K = 1024 bytes. The range is 1K-65K. The default is 1K. Each larger value of *n* requires a larger value of *size*.

m the number of areas of this *size* in this copy area. The range is 1-4096. The default is 4.

The sum of *m* for all *n* copy areas specified must be equal to or greater than maximum number of SCF and PLEX users, that is, the value of the USERS execution parameter. For example, if you specify the following, the value for the USERS parameter cannot exceed 11.

```
COPYAREA1=1K/4
COPYAREA2=3K/2
COPYAREA3=4K/5
```

The COPYAREA*n* parameter specifies a common area (or interregion copy area) for data transfer between Multi-User and a dependent region user. This copy area is required for fetch-protected systems because such systems prevent data in one region of the operating system from being directly addressed by another region. The copy area is an area of storage that both Multi-User and the dependent region can address.

During Multi-User initialization, a GETMAIN acquires all the storage that you requested with the COPYAREA parameter. The amount of storage obtained by the GETMAIN is the sum of (*size* x *m*) for all COPYAREA*n* parameters specified. For Type 2 SVC Multi-User software, this storage is obtained from the Common Service Area (CSA). For XMS Multi-User software, the storage is from the Multi-User region. In either case, if the initial GETMAIN fails, SYSTEM 2000 software issues a user abend, and Multi-User does not initialize.

PLEX jobs require copy areas that are larger than the 1K areas for SCF jobs. If Multi-User cannot find a large enough copy area for PLEX from the established copy areas, it issues a GETMAIN dynamically so the PLEX user can continue processing. If the GETMAIN succeeds, the software writes message 0017 to the Diagnostic Log. If this message occurs often, the current copy areas should be increased in either size or number or both. Because these dynamic GETMAINs increase overhead, they should be kept to a minimum. If a GETMAIN fails, the user involved is terminated with a user abend. The space obtained is freed and returned to the operating system when the user request has been processed by Multi-User.

Determining the number and size of copy areas To determine the size and number of copy areas, use the information gathered from the transaction rates, Multi-User Diagnostic Log, and console operator displays.

The number of copy areas depends on the number of concurrent requests made at any particular time. You can specify a maximum of six different copy areas with the COPYAREAn parameter. The default copy area (if required) is COPYAREA1=1K/4. Each SCF TP user requires a 1K copy area. To avoid any GETMAINs for copy areas, you must preallocate enough 1K copy areas for all concurrent users. that is, TPTHREADS and USERS. Otherwise, bigger copy areas are used. This could force PLEX programs to use GETMAINs or to tie up bigger copy areas than they really need. Since a copy area obtained with a GETMAIN is freed at job termination, an inadequate specification of copy areas can cause system degradation due to excessive GETMAIN/FREEMAIN requests. All GETMAIN requests are written to the Multi-User Diagnostic Log if it is active.

The size of your copy areas depends on the mix of SCF and PLEX jobs that you expect to have running in the Multi-User session.

SCF batch jobs and SCF TP jobs execute successfully in the smallest area (1K). When Multi-User is initialized, four of the smallest available areas are provided as the default (COPYAREAn=1K/4).

PLEX programs require a variable size depending on the specific PLEX command and the size and number of parameters in the call to SYSTEM 2000 software, for example, S2KDUM, COMMBLOCK, SUBSCHEMA RECORD, where-clause, and other parameters. When the PLEX program is initialized, the smallest available area is provided until job termination or until a larger area is required.

If a PLEX command requires a larger area, the original area is freed for reuse, and another larger copy area is obtained to process the command. If the larger area cannot be obtained from one of the established copy areas, Multi-User attempts to build a special area by issuing a GETMAIN. If the storage is not available, the PLEX program is terminated.

A PLEX program uses a constant 868 bytes of the copy area for holding pointers to copy results back into the user region. Most PLEX commands, therefore, do not execute in a 1K copy area.

To obtain the copy area size needed for a PLEX command under Multi-User, compute the total number of bytes needed for the following:

- constant: Initial 868 bytes
- S2KDUM: 152 bytes
- COMMBLOCKs: (number of databases in command) times 124 bytes
- SUBSCHEMA RECORD
  - SUBSCHEMA Control Block (SCB): 56 bytes.
  - Item Control Blocks (ECBs): SCBNUM times 56 bytes. SCBNUM is the number of items in the subschema record and is the sixth field in the SCB. (See Illus. A.7 on page A-10 for fields in the SCB.)
  - length of the subschema record I/O area in bytes. This length is in SCBBUF, the third field of the SCB. (See Illus. A.7 on page A-10 for fields in the SCB.)
- where-clause list
  - parameter list: ((number of items and WORKING-STORAGE values in the where-clause) + 1) times 4 bytes
  - Operator Control Block (OCB): (number of operators + 1) times 4 bytes
  - sum of the picture sizes of all items and WORKING-STORAGE values given in the where-clause, each value rounded to a word boundary
  - number of components in the where-clause times 56 bytes
  - number of AT operators times 4 bytes
- ORDER BY list
  - parameter list: (number of ordering items + 1) times 4 bytes
  - Operator Control Block (OCB): same as the length of the parameter list.

DITTO Parameter

```
DITTO=|NO   (default)
      |YES
```

The DITTO parameter can be used to allow the use of the DITTO command in a Multi-User environment. The DITTO command repeats the action-clause of the last SCF command.

If DITTO=YES, SCF users can use the DITTO command to refer to the last action-clause given in the previous segment. This means that DITTO can be used without $LOCK, which ties up a thread.

If DITTO=NO, users cannot use the DITTO command to refer to a previous segment. If a user tries to use the DITTO command, SYSTEM 2000 software issues an error message.

FPTPSYS Parameter

```
FPTPSYS=|YES   (default)
        |NO
```

If FPTPSYS=YES (the default), system areas are obtained when Multi-User is initialized. This allows TP access from different MVS address spaces.

The FPTPSYS parameter should be YES if the host operating system is fetch-protected, such as MVS. The FPTPSYS parameter is directly related to the SYSTEM 2000 Type 2 SVC macro parameter OS=MVS.

If FPTPSYS=NO, the memory overhead for these additional areas is not encountered, but the TP system must be executed in the same address space as Multi-User. If FPTPSYS=NO and TPTHREADS is greater than zero in a fetch-protected system, a warning message is issued to the operator.

LHOLD Parameter

```
LHOLD=|YES   (default)
      |NO
```

If LHOLD=YES, all PLEX programs running in the Multi-User session must obtain a local hold on any data record that is to be updated. The local hold is requested by specifying the HOLD option in the retrieval command preceding the update. If the MULTIPLE HOLDS option is enabled, more than one local hold can be obtained on different records simultaneously for a database. (See the ENABLE/DISABLE MULTIPLE HOLDS command in *SYSTEM 2000 CONTROL Language, Version 12, First Edition*.)

| For information about the buffers used for multiple local holds, see **Multiple Local Holds (MLH) buffer** on page 3-43.

LHOLD=YES is ignored if the database

- has the Rollback Log enabled
- has a global hold
- is under exclusive use.

If LHOLD=NO, a local hold is not required prior to a PLEX update in the Multi-User session. The HOLD option can be specified, but it is not required by Multi-User.

OPI Parameter

```
OPI=|NO     (default)
    |YES
```

The OPI parameter determines whether the software will allow the console operator to select execution parameters during the Multi-User initialization step. At that time the operator can change any execution parameter if OPI=YES. This is the highest priority level for overriding execution parameter settings.

If OPI equals YES, a message requesting parameters or a U appears on the operator console. The operator can enter as many parameters as needed, one at a time. After each parameter is accepted, the message requesting parameters is redisplayed. When all the necessary parameters have been entered, the operator must enter a U, which tells Multi-User to finalize initialization.

OPT034 Parameter

```
OPT034=|NO     (default)
       |YES
```

If OPT034 equals YES, QueX software will allow a user to update a record without requiring a hold on the record. NO (the default) means QueX requires a hold before updating the record. (OPT034 replaces Special Zap #181.)

PQA Parameter

```
PQA=|FIFO                    (default)
    |PRTY[/n/y]
```

FIFO user queuing by first in first out.

PRTY user queuing by user priority with *n* and *y*.

n $0 < n \leq 999$ for user priority queuing. *n* is the number of requests that will be dispatched from the dispatch queue before the software examines the queue for priority adjustments. The default is zero if *n* is not specified.

y $0 < y \leq 255$. *y* is a number used as the increment for increasing a user priority. The default is zero.

The PQA parameter determines the priority queuing algorithm. If PQA=FIFO (default), user queuing is by first-in-first-out. All users have a priority of zero regardless of the value in the user request block.

If you do not specify *n* and *y* when PQA=PRTY, the values of *n and y* are zero. Multi-User does not examine or adjust the queue.

If you specify *n* and *y*, Multi-User examines the queue after every *n* dispatches. Any user not dispatched in the last *n* dispatches has his priority increased by *y* to a maximum of 255.

The console operator can display the current PQA setting and can change it. Changing the PQA parameter settings during a Multi-User session creates additional overhead. If PQA is altered, Multi-User must change the priority of the queue entry for each active user, as well as the PQA parameter.

The accounting exit in dependent region interfaces allows modification of an additional field (URBPRTY) that can have a value of 0-255. This exit is available during signon. The URBPRTY has no effect on FIFO queuing. URBPRTY is a one-byte field at displacement X'1F' into the URB. The relationship between the PQA parameter value and a user's priority is as follows:

- If PQA=FIFO, the user's priority (the value of URBPRTY) is ignored.
- If PQA=PRTY, the value of URBPRTY is stored in the user's queue entry in the Multi-User control blocks, and he is chained onto queues in priority order. The value is zero if the appropriate accounting exit is not linked to the interface or if URBPRTY was not set by the exit.
- If the console operator changes the PQA parameter from FIFO to PRTY, the value of each user's URBPRTY is copied into his Multi-User queue so that all subsequent calls to Multi-User cause the users to be queued in priority order.
- If the console operator changes the PQA parameter from PRTY to FIFO, each user's priority is set to zero in his Multi-User queue entry. His URBPRTY remains unchanged.

The Database Administrator is responsible for seeing that the appropriate exits are linked with the Multi-User interfaces. See also the discussion of user accounting exits in **Exits Used for Changing Fields in Accounting Log Records** on page 3-122.

User priority queuing is written to the Accounting Log. Also, the Diagnostic Log thread start message shows user priorities.

Notes on queuing and wait list processing The SYSTEM 2000 SVC for Multi-User maintains a Dispatch Queue (DQ) in priority order or FIFO, based on the PQA execution parameter. If priority queuing is specified (PQA=PRTY/*n*/*y*), the priority is obtained from the User Request Block (URB) and can be adjusted by the user accounting exit. The priority range is 0-255. The lowest priority is zero (default).

The Multi-User Scheduler always takes the first entry off the Dispatch Queue if a thread is available or if that user is already in a thread. Otherwise, Multi-User searches the queue for a user in a thread and dispatches the user.

Multi-User has only one ECB that is used for posting any user request. After the SVC chains a request to the Dispatch Queue, it posts this ECB (only if waiting).

SAME Parameter

```
SAME=|YES    (default)
     |NO
```

The SAME parameter controls the use of a previous where-clause across transmission segments. If SAME=YES, the SAME operator can be used to refer to a where-clause that was given in a previous segment. That is, SYSTEM 2000 software saves the where-clause results across segments. The $LOCK command that ties up a thread is not needed.

Where-clause results are retained on scratch pads, but no new buffers are locked. In a busy system, some additional I/O might be required to write the saved results to disk and to read them back later if they are referenced. If many SCF users require large where-clauses and the use of the SAME operator, consider giving more disk space to scratch pads. (See PAD*nn* parameter discussion in **PAD*nn* execution parameter** on page 4-41.)

If SAME=NO, the results of a where-clause are not saved across segments. If a user tries to use the SAME operator, SYSTEM 2000 software issues an error message.

SID Parameter

```
SID=n                    (The default is 1.)
```

n a range of 0-99.

The SID parameter is used in WTO messages. (For details, see *SYSTEM 2000 Messages and Codes, Version 12, First Edition.*) The SID parameter also sets the System ID that appears in Multi-User Accounting Log messages.

STARTTP Parameter

```
STARTTP=|YES    (default)
        |NO
```

If STARTTP=YES, the SCF TP facility is enabled during system initialization. The operator does not have to enter the START TP command.

If STARTTP=NO, SCF TP initialization is delayed until the console operator gives the START TP command. This can be used to allow critical SYSTEM 2000 sessions to be executed with reduced interaction from the SCF TP facility.

THREADS Parameter

THREADS=*n* *(The default is* **1.)**

n a range of 1-63.

The THREADS parameter specifies the number of threads that can be used simultaneously in a Multi-User session. For a single-user environment, THREADS must equal 1.

The Thread Work Areas occupy space in the Multi-User region. The size of each Thread Work Area (TRDCOM01) is 52K without the REPORT processor (RW=NO). The size with the REPORT processor (RWTCOM01) is 62K, that is, when RW=YES. One set of SF files (sort files) is required for each thread.

TPSCRUN Parameter

TPSCRUN=*n* *(The default is* **180.)**

n a range of 1-10000.

The TPSCRUN parameter specifies the maximum number of active and inactive SCF TP users. The inactive users are signed on but are not presently active. The S2KUSERS file must be allocated with enough disk space to handle the number of SCF TP users specified. Page size for the S2KUSERS file is 11592 bytes.

The number of concurrently active SCF TP users affects the buffers set up by the POOL*n* parameter as follows:

- There must be as many buffers of 11592 bytes or larger as there are SCF TP users, that is, TPTHREADS or THREADS, whichever is fewer.
- All SCF TP users receive output in 4K blocks.

TPTHREADS Parameter

TPTHREADS=*n* (*The default is* 5.)

n a range of 0-230.

The TPTHREADS parameter specifies the maximum number of active SCF TP users that can be queued for SYSTEM 2000 processing. Queued means the session is waiting in an SVC slot for a thread. If a slot is not available, an SCF TP user cannot sign on and is given a 'busy' message.

For XMS Multi-User software, the TPTHREADS execution parameter sets the limit. For Type 2 SVC MUlti-User software, the value for the TPTHREADS parameter must not exceed the value that was specified for the TPUSERS parameter in the SYSTEM 2000 SVC macro (see **The Multi-User Type 2 SVC** on page 3-22). If it does, a fatal error message is issued to the operator, and the Multi-User session continues without the SCF TP facility. To avoid re-generating the SVC, set the number in the SVC TPUSERS parameter high so that it will accommodate any increase in the number specified in TPTHREADS.

The TPTHREADS value can exceed the THREADS parameter value to provide for command queuing. This action risks SYSTEM 2000 saturation with TP activity at times of heavy TP message traffic. However, it can reduce command retries in the TP transaction caused when SYSTEM 2000 software signals a busy condition (due to the absence of an available queue slot).

TSO Parameter

TSO=|NO (*default*)
|YES

The TSO parameter is not necessary for MVS systems.

USERS Parameter

USERS=*n* (*The default is* **16.**)

n a range of 1-230.

The USERS parameter specifies the total number of jobs that can be executing SYSTEM 2000 software concurrently, that is, the total number of

- batch SCF jobs
- batch PLEX jobs
- PLEX TP jobs.

For XMS Multi-User software, the USERS execution parameter sets the limit. For Type 2 SVC MUlti-User software, the value of the Multi-User USERS parameter must not exceed the value that you specified for the USERS parameter when you generated the SYSTEM 2000 Type 2 SVC. If it does, a user abend is issued, and the Multi-User session ends. To avoid re-generating the SVC, set the number in the SVC USERS parameter high so that it will accommodate any increase in the number specified in USERS execution parameter.

Consider the value assigned to this parameter carefully, because a large number means more system overhead. When USERS, TPTHREADS, or TPSCRUN is larger than needed, Multi-User must build more internal tables than will actually be used. This uses more memory and more CPU time to scan the tables.

MULTI-USER CONSOLE OPERATOR COMMANDS

Operations under Multi-User involve two or more jobs and several types of activities happening simultaneously. The following information contains the console commands available for monitoring and controlling Multi-User activity, as well as a description of the messages issued by Multi-User to the operator console. You can issue console commands from the master console or from an alternate console.

S2OP is a program that looks like an alternate console to Multi-User software. It is an optional service that runs in three environments: TSO, CMS, and MVS batch.

As an alternate console, S2OP accepts all master console operator commands. WTO message 3216 appears when the alternate console interface is available.

A null line terminates a TSO or CMS session. An end-of-file (EOF) on the S2KCOMD file terminates an MVS batch session.

In MVS batch, S2OP requires allocations for the S2KMSG file and the S2KCOMD file. The S2KMSG file must be a SYSOUT data set or must have a DCB of RECFM=VB,LRECL=133, BLKSIZE=137. The S2KCOMD file must be a SYSIN data set or must have a DCB of RECFM=FB,LRECL=80. If the S2KMSG file is not present, it will be allocated dynamically. If S2KCOMD is not present, User Abend 517 occurs.

In an XMS Multi-User environment, S2OP requires an allocation for the S2KCOM file.

Special Zap #198 is available to limit S2OP to DISPLAY commands only, which prevents other activities such as canceling a job.

The master console operator communicates with Multi-User by using the MODIFY (or F) command. Using MODIFY commands prevents the need of any outstanding WTOR for Multi-User. The alternate console operator uses the S2OP command instead of the MODIFY command.

```
|MODIFY   jobname , text          (master console)
|F

S2OP text                         (alternate console)
```

jobname The SYSTEM 2000 Multi-User jobname. In the examples given in this section, S2K is used for the jobname.

text The different values for the text are described in the following sections. The text can contain embedded blanks. Use quotes around the text if required by the operating system.

The console operator commands provide the following functions while Multi-User is active:

- displaying the status of all open databases
- displaying the status of all active and inactive jobs or terminals under Multi-User and, optionally, the status of database(s) being used by each job or terminal

- displaying the status of a specific job or terminal and, optionally, the status of the database(s) it is using
- initializing the SCF TP facility
- canceling a specific SYSTEM 2000 job or terminal
- terminating the Multi-User session or canceling the SCF TP facility
- displaying user priorities (PQA)
- modifying Diagnostic Log and Accounting Log parameters
- varying databases offline and online.

Note: A set of MUSTATS console operator commands is available as part of the tuning tools feature. The MUSTATS commands are described in **MUSTATS Console Operator Commands** on page 3-81.

Repeating Console Operator commands

For CMS, TSO, and MVS batch environments, you can use the plus sign (+) to repeat the previous console command from an alternate terminal. The "+" must be the only character on the line. This option is not available from the master console. CICS users can also repeat a console command. For details, see *SYSTEM 2000 Interface to CICS (Command Level), Version 12, First Edition.*

Displaying the Status of All Databases

To display the status of all databases connected to Multi-User at any given time, use the following syntax:

```
|MODIFY S2K,    |DISPLAY   ß     |STATUS
|F              |D               |S
```

The level of control (status of the database) and the user jobname or terminal ID are displayed in the message. Only the highest level of control is displayed. Multiple users could have the database open, but only the one with the highest control level is listed. The control level hierarchy for database status and associated symbols follow:

Level of Control	Mnemonic
LOCK	L
ROLLBACK	R
HOLD	H
UPDATE	U

A retrieval status or inactive status contains blanks in the status and jobname fields. For a more detailed explanation see Illus. 3.6 on page 3-67.

Illus. 3.6 Status Display Symbols

Type of Job	
SCF	SCF including all SCF TP run-units
PLX	Program Language Extension (PLEX)
ALT CON	Alternate console.

Wait/Execute Status		
E		Executing
WT	|TAPE	Waiting in a thread:
	|PAD	TAPE - SAVE, RESTORE, KEEP or APPLY needs a tape
	|BUFFER	PAD - user needs an allocation unit
	|OVERLY	BUFFER - user needs a buffer
	|UPDLOG.	OVERLY - user needs an overlay segment that is currently swapped out of memory
		UPDLOG - user needs access to an Update Log
W.		Waiting in the input queue (suspended)
.		Not executing and not waiting for resources (no database hold)
N		The user has a global hold on a database, but the user is not currently active in the Multi-User region. For example, a FRAME has executed successfully, but the program has not yet returned to do another PLEX command. The database remains in hold status.

Current Resource Status	
OPEN	Request for non-exclusive use of a database: waits if not granted except returns to TP. (OPEN command in PLEX or DBN command in SCF.)
OPENR	Same as OPEN except issues a return code to PLEX user program if not granted. (OPENR command in PLEX and not applicable in SCF.)
LOCK	Request for exclusive use of a database; waits if not granted except returns to TP. (OPEN/LOCK command in PLEX or EXCLUSIVE DBN command in SCF.)
LOCKR	Same as LOCK except issues a return code to PLEX user program if not granted. (OPENR/LOCK command in PLEX and not applicable in SCF.)

continued on next page

Illus. 3.6 *continued*

Current Resource Status	
HOLD	Requests a global hold on a database to allow retrievals but prevents updating until the hold is relinquished. (HOLD option in PLEX queue mode; for SCF implicitly holds during the where-clause phase of any QUEST update command or the QUEUE selection phase if the session contained updates.)
UPDATE	Database being updated due to a command in an active thread. (MODIFY, REMOVE, INSERT or TERMINATE commands in PLEX; for SCF, TERMINATE command, following the selection phase if the session contained updates, or the action-clause of a QUEST update request.) The duration is the completion of the command.
RETREV	Database in retrieval mode due to a command in an active thread. (Any SCF or PLEX command whose action verb does not imply an alteration of the database.) The duration is the completion of the command.
ROLBK	The job is waiting to get control of the database to perform recovery. It must wait until all current updates, retrievals, and global holds are finished.
FRAME	The job is waiting to FRAME a database (an SCF or PLEX FRAME command). It must wait until all current updates and global holds are finished.
LHOLD	The job is waiting to obtain a local hold on a specific record (a PLEX GET/HOLD or update if rollback is enabled). It must wait until the user who is currently holding the record relinquishes it.
NKUPD	The job is waiting to do a non-key update. It must wait until all updates and global holds are finished.
EXECUT	Not in retrieval, hold, or update mode.
NO ACT	No SYSTEM 2000 activity requested. The user program is currently active in the dependent region.
UP ACT	User partition activity. This status indicates that a thread is tied up on behalf of the indicated user, but control has passed to the user partition. This could happen, for example, if a message for the S2KMSG data set is returned to a batch user.

The following example illustrates requesting the status of all databases:

```
F S2K,DISPLAY STATUS

MODIFY S2K,D S
```

(**Note**: Use quotes if required by the operating system, for example, F 'DISPLAY STATUS' or 'D S'.)

The resulting output is as follows:

```
S2K1429/01-        DBN          PLX SCF S    JOB     -
S2K1431/01- DBONE               001 002 -
S2K1431/01- DBTWO               002 003 -
S2K1430/01- DBTHREE             000 001 U  JOB123    -
S2K1430/01- DBFOUR              001 001 H  JOBPQR    -
S2K1430/01- DBFIVE              002 001 R  JOBABC    -
```

The first line of output is a column heading where PLX is the PLEX user count and SCF is the SCF user count, including any SCF TP users signed on to the system.

DBONE has one PLEX user and two SCF users. The database is in either retrieval or inactive status.

DBTWO has two PLEX users and three SCF users. The database is either in retrieval or inactive status.

DBTHREE has a user (JOB123) that has a specific level of control over the database (UPDATE).

DBFOUR has user JOBPQR with a HOLD on the database.

DBFIVE has user JOBABC doing rollback on the database.

If there are no active databases, the display output is 'NO ACTIVE DATA BASES'.

Displaying the Status of All Jobs

To display the status of all Multi-User jobs, use the following syntax:

```
|MODIFY   S2K,     |DISPLAY  ъ   |ACTIVE     |,STATUS
|F                 |D            |A          |,S
```

A list of the jobnames and TP terminal IDs currently running under Multi-User is displayed. If the STATUS operand is also entered, the list contains the databases each user has open. Inactive terminal run-units are also listed with the WTO message S2K1437/*sid*. The absence of an active database is indicated by blanks. If there are no active jobs, WTO message S2K1401/*sid* is issued.

The following example shows an operator request for the status of all active jobs without the STATUS operand and the resulting output:

```
F S2K,DISPLAY ACTIVE
```

(**Note**: Use quotes if required by the operating system, that is, 'DISPLAY ACTIVE'.)

The resulting output is as follows:

```
S2K1432/01-  JOB890  SCF     000 .  NO ACT -
S2K1432/01-  JOBABC  PLX     000 E  RETREV -
S2K1432/01-  JOB123  SCF     000 E  UPDATE -
S2K1432/01-  JOBPQR  PLX     000 E  EXECUT -
S2K1432/01-  JOBXYZ  SCF     000 E  HOLD   -
S2K1433/01-  JOB345  PLX     000 W. UPDATE DBTHREE -
S2K1433/01-  JOBLMN  SCF     000 W. LOCK DBTHREE -
S2K1437/01-  TER123  TP      DBTWO            -

*  system ID
```

This display indicates the following facts when the symbols are interpreted according to Illus. 3.6 on page 3-67:

```
JOB890    SCF      =   SCF job name or active terminal ID
          000      =   job priority
          .        =   not in a thread
          NO ACT   =   no SYSTEM 2000 activity requested

JOBABC    PLX      =   PLEX program
          000      =   job priority
          E        =   executing
          RETREV   =   in retrieval mode

JOB123    SCF      =   SCF jobname or terminal ID
          000      =   job priority
          E        =   executing
          UPDATE   =   in update mode

JOBPQR    PLX      =   PLEX program
          000      =   job priority
          E        =   executing
          EXECUT   =   not in retrieval, hold, or update mode

JOBXYZ    SCF      =   SCF jobname or terminal ID
          000      =   job priority
          E        =   executing
          HOLD     =   has a HOLD on a database
```

```
JOB345    PLX       =   PLEX program
          000       =   job priority
          W.        =   waiting in the input queue
          UPDATE    =   resource request pending on DBTHREE

JOBLMN    SCF       =   SCF jobname or terminal ID
          000       =   job priority
          W.        =   waiting in the input queue
          LOCK      =   resource request pending on DBTHREE

TER123    TP        =   inactive terminal run-unit using database DBTWO
```

If one of the active users is an alternate console, such as an 'S2OP' CICS transaction, the user type is ALT CON. WT is followed by the name of the resource in message S2K1432, as shown in Illus. 3.6 on page 3-67.

This example shows an operator request for the status of all active jobs. It includes the STATUS operand.

```
F    S2K,D A,S
```

The resulting output is as follows:

```
S2K1432/00- JOB890   SCF      000 .  NO ACT -
S2K1429/00-       DBN        PLX SCF S    JOB    -
S2K1431/00- DBONE            001 002 -
S2K1432/00- JOBABC   PLX      000 E  RETREV -
S2K1431/00- DBONE            001 002 -
S2K1430/00- DBFIVE           002 001    JOBABC   -
```

This is the same command as the previous DISPLAY ACTIVE command except the name and status of each database that is open for the job (or terminal ID) is listed along with the job status.

Displaying the Status of a Specific Job

To display the status of a specific job or terminal run-unit, use the following syntax:

```
|MODIFY    S2K,    |DISPLAY    b    |terminal-ID    |,STATUS
|F                 |D                |jobname        |,S
```

The optional STATUS operand displays the name and status of each database that the job (or terminal) has open.

The following operator command (without the STATUS operand) requests the status of a specific SYSTEM 2000 job, JOBABC:

```
F   S2K,DISPLAY JOBABC
```

(**Note**: Use quotes if required by the operating system, that is, 'DISPLAY JOBABC'.)

The resulting output is as follows:

```
S2K1432/00-  JOBABC  PLX     000 E  UPDAT  -
```

The display indicates JOBABC is a PLEX program. It is executing database updates.

The following example uses the STATUS operand in the request for the status of job. JOBABC:

```
F   S2K,D   JOBABC,STATUS
```

The resulting output is as follows:

```
S2K1432/00- JOBABC    PLX       000 E  UPDAT  -
S2K1429/00-       DBN          PLX SCF S    JOB    -
S2K1431/00- DBONE              001 002 -
S2K1430/00- DBFIVE             002 001 U  JOBABC   -
```

This display is the same as the previous display for JOBABC except it shows that JOBABC has two databases open and is updating DBFIVE. The total number of SCF users (including SCF TP users) and PLEX users for each database is also shown.

Changing the PQA Setting

To change the PQA execution parameter (Priority Queuing Algorithm), use the following syntax:

```
|MODIFY    S2K,PQA =    |PRTY [/n/y]
|F                      |FIFO            (The default is FIFO.)
```

FIFO user queuing by first-in-first-out.

PRTY user queuing by user priority.

n $0 \leq n \leq 999$. For user priority queuing, *n* is the number of requests dispatched from the dispatch queue before the software examines the queue for priority adjustments. The default is zero if *n* is not specified.

y $0 < y \leq 255$. This number becomes the increment when a user priority is being increased. Multi-User examines the dispatch queue every *n* dispatches. Any user not dispatched in the last *n* dispatches has his priority increased by *y* to a maximum of 255.

Changing the PQA setting during a Multi-User session creates additional overhead. If PQA is altered, every job queue entry associated with an active user must be changed as well as the PQA parameter setting.

The output message is

```
S2K1436/01- PRIORITY QUEUING ALGORITHM HAS BEEN MODIFIED -
```

Displaying the Current PQA Setting

To display the current PQA setting, use the following syntax:

```
|MODIFY    S2K,    |DISPLAY   PQA
|F                 |D
```

For example, if the operator enters the F S2K, D PQA command, the output is one of the following replies, depending on the current setting of the PQA parameter:

```
S2K1435/01- PQA = FIFO -
S2K1435/01- PQA = PRTY -
S2K1435/01- PQA = PRTY/n/y
```

Initializing the SCF TP Facility

To initialize the SCF TP facility after Multi-User has been initialized, use the following syntax:

```
|MODIFY    S2K,     |START   b   TP
|F                  |S
```

This sample command initializes SCF TP.

```
F   S2K,START TP
```

(**Note**: Use quotes if required by the operating system, that is, 'START TP'.)

If successful, the response is

```
S2K1421/00 - SCF TP ACTIVATED -
```

If SCF TP was quiesced , the response is

```
S2K1422/00 - SCF TP RE-ACTIVATED -
```

If the request is not successful, one of the following messages is issued:

```
S2K1425/00 - FATAL ERROR DURING 'START TP' -
S2K1426/00 - NO SCF TP THREADS ALLOCATED -
S2K1423/00 - SCF TP ALREADY ACTIVE -
```

Changing the Level of Diagnostic Log Messages

To request a different level of Diagnostic Log messages. the operator can reset the LOGLEVEL execution parameter settings with the following MODIFY command:

```
|MODIFY   S2K    LOGLEVEL=  (same as exec parameters)
|F
```

If LOGLEVEL=NO when Multi-User was initialized, the operator cannot change the setting.

The following command changes the active LOGLEVEL parameter settings:

```
F   S2K,LOGLEVEL=UINIT,TSTRT
```

When changing levels of messages, the operator must respecify all levels that are to be in effect.

The following examples show how to request a display of the active settings of the LOGLEVEL parameter:

```
F   S2K,DISPLAY LOGLEVEL
F   S2K,D LOGLEVEL
```

Changing the Status of Segment Statistics

To dynamically change the status of Multi-User Accounting Log segment statistics, use the following syntax:

```
|MODIFY  S2K,     |START  STAT      |NL   (Start or stop segment
|F                |STOP             |TP    statistics for SCF,
                                    |PL    SCF TP,  PLEX, or ALL.)
                                    |ALL
```

These commands start and stop logging of statistics, but they cannot reset the thresholds that were set by the execution parameters.

The following example starts segment statistics for PLEX jobs:

```
F   S2K,START STAT PL
```

For more details about segment statistics, see **Generating Accounting Data: SYSTEM 2000 Execution Parameters** on page 3-110.

Canceling a Specific Job with an Optional Dump

To cancel a specific Multi-User job, use the following syntax:

```
|MODIFY   S2K       |CANCEL   b   jobname   [,DUMP]
|F                  |C
```

jobname is the name of the specific job or terminal ID to be canceled.

DUMP generates a dump of the entire Multi-User region before the specified jobname is canceled. If the job name does not exist, a dump is still taken, and a message appears informing you that the job was not found.

Specifying the DUMP option when canceling a job is equivalent to issuing the DUMP command followed by a CANCEL *jobname* command.

Job names and job status are obtained by one of the DISPLAY commands. Canceling a specific job or TP run-unit with this console operator command does not damage the user's database. The user is allowed to finish the current command if he is updating a database. Otherwise, cancellation is immediate. An operator cancel causes user abend code 551 for any job, including PLEX running in TSO foreground.

The following sample command cancels job JOBABC:

```
F  S2K,CANCEL JOBABC
```

(**Note**: Use quotes if required by the operating system, that is, 'CANCEL JOBABC'.)

The operator receives the following successful cancellation message if job JOBABC is not updating a database:

```
S2K1403/00- JOB ABCDEF MARKED FOR CANCELLATION -
```

The operator receives the following message if job JOBABC is updating a database:

```
S2K1409/00- JOB ABCDEF MARKED FOR CANCELLATION
AWAITING UPDATE COMPLETION -
```

For an unsuccessful cancellation of a job or TP terminal run-unit, the following message is issued:

```
S2K1404/00- JOB jobname NOT FOUND -
```

Canceling More Than One Job

To cancel more than one job (or TP run-unit) at a time or to cancel a specific terminal ID for a specified job name, use the following syntax:

```
|MODIFY S2K,    |CANCEL JOB=jobname |,*
|F              |C                  |,termid
```

jobname is the name of the job.

* means cancel all jobs or TP run-units, including duplicates, that are active under the specified job name.

termid means cancel a specific TP run-unit for that job name, including duplicates.

If an update command is being processed, the job is marked for cancellation and is not canceled until the update finishes.

The CANCEL JOB command is convenient. For example, CANCEL JOB=CICSPROD,* cancels all terminals associated with CICSPROD. This allows you to cancel all CICSPROD jobs without terminating the Multi-User session.

Dumping without Terminating the Multi-User Session

To request a dump without terminating the Multi-User session, use the following syntax:

```
|MODIFY,DUMP
|F
```

The DUMP command generates a dump of the entire Multi-User region without disturbing any Multi-User activities. (You can also take a dump when you cancel a specific job or when you terminate Multi-User.)

Terminating Multi-User Software or SCF TP

To terminate Multi-User or SCF TP, use the following syntax:

```
|MODIFY S2K,    |CANCEL  |S2K    |,WHEN  [,DUMP]
|F              |C       |TP     |,W              (The default is
                                 |,NOW             WHEN without a
                                 |,N               dump.)
                                 |,FORCE
```

The CANCEL S2K command allows the operator to terminate a Multi-User session or the SCF TP facility without the risk of damaging any user's database(s). After the command is issued, no new SYSTEM 2000 users (or SCF TP users if 'TP' was given) are allowed in the system.

Exactly when Multi-User or SCF TP is terminated depends on whether the WHEN. NOW, or FORCE option is entered. A PLEX program under the TP monitor is not like an SCF TP job; after the PLEX program starts executing, it is treated as a batch PLEX program.

W
WHEN — means Multi-User or SCF TP ends when all the current SYSTEM 2000 users (or SCF TP jobs) have finished their individual sessions and have exited from Multi-User. No new users are allowed to sign on.

N
NOW — means Multi-User generates a specific cancellation for each current Multi-User job name or SCF TP run-unit. If the job or run-unit is updating a database, the update command in progress is allowed to finish. Otherwise, it is canceled immediately. When all jobs (or SCF TP run-units) have been canceled, Multi-User (or SCF TP) terminates.

FORCE means Multi-User abends without doing any clean-up. It is equivalent to an MVS-cancel and disables all ESTAE processing. Therefore, try to cancel Multi-User with WHEN or NOW before using the FORCE option. **Note**: The FORCE option can cause damaged databases. The FORCE option can only be used with the S2K option. (It cannot be used to cancel SCF TP.)

The FORCE option does not produce a system dump. It produces a SNAP dump. The ID for the snapshot is 61.

For details about the FORCE option when Multi-User is running with IBM Cross Memory Services (XMS), see **Cross Memory Services (XMS) Multi-User Software** on page 3-16.

DUMP means a snapshot dump is taken. The DUMP option must be preceded by appropriately positioned parameters, that is, 'C S2K,DUMP' is invalid but 'C S2K,W,DUMP' is valid. The file DDname is S2KSNAP.

The following console operator commands show sample syntax for the various forms of the CANCEL command along with the message issued for each command:

Sample Commands	Response
F S2K,CANCEL S2K,WHEN	S2K1411/00- SYSTEM 2000 WILL TERMINATE WHEN THERE ARE NO USERS -
F S2K,C S2K,N,DUMP	S2K1410/00- SYSTEM 2000 WILL TERMINATE NOW - indicates that the termination request has been received. This message is followed by a cancellation message for each active SYSTEM 2000 user.
F S2K,C TP,NOW	S2K1420/00- SCF TP ACTIVITY BEING TERMINATED cancels each SCF TP run-unit immediately or after conclusion of the current update command.
F S2K,C TP,W	S2K1419/00- SCF TP ACTIVITY BEING QUIESCED - terminates SCF TP when all SCF TP run-units have exited the system.

(**Note**: Use quotes if required by the operating system, that is, 'C S2K,WHEN'.)

Multi-User regards SCF TP as being active as long as any terminal user is still signed on to the system. Thus, the CANCEL S2K command does not terminate Multi-User if any SCF TP user has been initialized. A 'CANCEL S2K, W' or a request to cancel SCF TP prevents any new SCF TP users from being accepted. Users attempting to sign on after such a cancellation receive a message that the system is being quiesced. When SCF TP has terminated, WTO message S2K1424/00 is issued. Cancellation messages for each job or SCF TP run-unit are issued and, also, if the Multi-User session is terminated, S2K1215/00 is issued as the final message.

To resume SCF TP activity following a CANCEL TP command, the MODIFY command START TP can be issued again. Also, SCF TP can be initialized again at Multi-User initialization time with the STARTTP=YES execution parameter.

Terminating Multi-User Software When SCF TP Is Active

To terminate Multi-User when SCF TP is active, first issue the following command to cancel Multi-User:

```
F jobname,C S2K          (implied WHEN)
```

Then, cancel SCF TP by giving the following command:

```
F jobname,C TP,WHEN
```

When the last SCF TP user issues an EXIT command, SCF TP terminates. This termination allows Multi-User to terminate.

If the NOW option is used, the console operator must first cancel SCF TP with the NOW option.

```
F jobname,C TP,NOW
```

Then the operator can issue the MODIFY command to cancel Multi-User.

```
F jobname,C S2K NOW
```

(**Note**: Use quotes if required by the operating system.)

Varying Databases Offline/Online

To vary a database offline (and online), use the following syntax:

```
|MODIFY S2K,   VARY  database      |OFFLINE
|F                                 |ONLINE
```

where *database* is the name of the database to be flagged offline (deallocated) or online (allocated).

The VARY console command applies only to databases specified in the S2KDBCNT table. The S2KDBCNT table controls access to databases that can be dynamically allocated by SYSTEM 2000 software. For more information about the S2KDBCNT table, see **S2KDBCNT Table** on page 4-17. For information about the ALLOC execution parameter, see **ALLOC execution parameter** on page 4-16.

Note: If the ALLOC execution parameter equals NO, the VARY command is not honored and an error message appears.

OFFLINE sets a flag indicating that Multi-User should dynamically deallocate the specified database when the last user closes the database. The OFFLINE option also prevents new users from accessing the database. The flag is set only if the database is specified in the S2KDBCNT table. (See also, the CANCEL *database* console command, discussed next.)

After a database is varied offline with the VARY command, the database is unavailable until you bring it online with the ONLINE option.

ONLINE brings a database back online if you varied it offline with a previous VARY console command. ONLINE makes the database available to users again.

The VARY console command is similar in function to the SCF ALLOC and FREE commands, but VARY is a Multi-User system command and takes precedence.

Canceling All Users on a Database

To cancel all users on a database, use the following syntax:

```
|MODIFY   S2K    |CANCEL   database
|F               |C
```

where *database* is the name of a database. This console command is useful in conjunction with the VARY OFFLINE console command, since it offers an effective way of freeing a database.

If an update command is being processed when you issue this command, the user is marked for cancellation and is not canceled until the update finishes.

If the requested database is not open, WTO message 1447 S2K1447/*sid- database* DATABASE NOT OPEN - is issued.

MUSTATS CONSOLE OPERATOR COMMANDS

The Multi-User status (MUSTATS) commands display details about buffers, databases, users, threads, scratch pads, and queues. You can submit these commands from the master console or from an alternate console.

The MUSTATS console operator commands are part of the tuning tools feature. Other tuning tools available include SAS procedures that generate Accounting Log and Diagnostic Log reports. Data are extracted from these logs to create SAS data sets. You can also produce Accounting Log graphics with SAS/GRAPH. For details about the Multi-User tuning tools, see *Technical Report: S2-106 Multi-User Tuning Tools for SYSTEM 2000 Software, Release 11.6 under OS and CMS.*

Here is a summary of the MUSTATS commands:

Command	Description
BUFFERS	displays information about buffers
DBNS	displays the names of open databases
DBN=	displays information about a specific database
DBNU=	displays users of a database
DBSTAT=	displays database permissions and holds
HELP	displays a list of available MUSTATS commands
MLH=	displays local holds on a specific database
PADS	displays scratch pad information
POOLS	displays information about buffer pool usage
QUEUES	displays the length of the four Multi-User work queues
THREADS	displays information about thread usage
USER=	displays information about a user
WHY=	displays the reason for a user wait

MUSTATS Command Syntax

The general format for MUSTATS console commands appears below. SEND MUSTATS can be abbreviated with a pound sign (#).

```
|SEND MUSTATS command
|# command
```

If you submit MUSTATS commands from the master console under MVS, you must precede each command with one of the following. Alternate consoles do not need these identifiers.

```
|MODIFY jobname,
|F jobname,
```

In the first two examples, $MU is the Multi-User jobname.

```
F $MU, #DBNU=                          (master console under MVS)
MODIFY $MU, SEND MUSTATS DBNU=         (master console under MVS)
#DBNU=                                 (alternate console)
SEND MUSTATS DBNU=                     (alternate console)
```

The individual commands are described in alphabetical order, beginning on the next page. If the keyword for a command is plural, the command does not require any parameters. If the keyword for the command is singular, the command requires a parameter value.

Repeating a Console Command

If you are using S2OP as an alternate console, you can use the plus sign (+) to repeat the previous MUSTATS command. In fact, the plus sign repeats any Multi-User console operator command. for example, the D A, S command. The plus sign must be the only character on the line. **Note**: The plus sign cannot be used from the master console.

The example below illustrates repeating the MUSTATS DBNS command.

```
#DBNS
  ...        ...        ...        ...
+
  ...        ...        ...        ...
```

BUFFERS Command

The BUFFERS command displays information about current buffer usage. This command has no parameters. To request buffer information, enter

#BUFFERS

The output consists of one line of information for each defined buffer.

```
VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV
-BUFFERS                                     I L S D M L
-                                            O O C B O O
- P#    OWNER     PAGE                 AGE   P K R   D G
  1       2         3                   4    5 6 7 8 9 10
><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
```

1 is the pool number of the pool that contains this buffer.

2 is the entity this buffer is assigned to, for example, database File 6. UNUSED means the buffer is not in use. If this field contains an eight-digit hexadecimal number, the buffer is being used for some general purpose, such as an internal array or control block.

3 is the page number of the page currently in the buffer, for example, the page number of one of the database files.

4 is a relative measure of time elapsed since this buffer was assigned. The oldest buffers have the smallest numbers. In fact, 0 means the buffer was assigned when the Multi-User software was initialized. Buffers assigned more recently have higher numbers.

5 indicates whether an I/O is in progress. Y means YES.

6 indicates whether the buffer is locked against reassignment to another use. Y means YES.

7 indicates whether the buffer is assigned to a scratch file. Y means YES.

8 indicates whether the buffer is assigned to a database file. Y means YES.

9 indicates whether any data on this page have been modified. Y means YES.

10 indicates whether this page (if modified) needs to be written to the Update Log.

DBNS Command

The DBNS command displays the names of all databases currently open in the Multi-User environment. This command has no parameters.

```
#DBNS
```

The output is a list of database names in the following format:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-DBNS          1       -2             1       -2
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
```

1 is the name of a database that is currently open in the Multi-User session.

2 indicates whether the database is defined as small or large. S means small, and L means large.

DBN= Command

The DBN= command displays information about a specific database that is open in the Multi-User environment. DBN= requires one parameter, the name of a database.

#DBN=*database*

The information displayed for the specified database is shown below. One line appears for each database file. Six lines always appear for database Files 1 through 6. Seven lines appear if update logging is activated (File 7). Eight appear if rollback is enabled (File 8).

```
++++++++++++++++++++++++++++++++++++++++++++++++++++
-DBN=  1
 #   BLK   CURR  IOCOUNT  P/ALLOC  P/USED  RPM  WPM
 2    3     4       5        6        7     8    9
 LAST SAVE/CREATE 10
   11
 RESET RB AT  12 PAGES/  13%/  14 SYNCHPOINTS
 APPLY LEGAL -  15 LOG -   KEEPFILE = 16
 MLH ENABLED - OF   17 ENTRIES   18 ARE USED
<><><><><><><><><><><><><><><><><><><><><><><><><><><
```

1 displays the database name, the current cycle number, and the date and time of the last update.

2 identifies the database file number for the information displayed on the line.

3 is the block size (page size) for this database file.

4 is the number of buffers currently assigned to this database file.

5 is the number of I/O operations that were performed against this database file.

6 is the number of pages formatted in the underlying data set.

7 is the number of logical pages used in this database file.

8 is the rate of reads-per-minute for this database file if XBUF is enabled. If the XBUF software is not enabled, this field contains blanks. If the rate is too old, asterisks appear.

9 is the rate of writes-per minute for this database file if XBUF is enabled. If the XBUF software is not enabled, this field contains blanks. If the rate is too old, asterisks appear.

10 is the cycle, date, and time at the last save of this database or at the creation of this database.

11 contains the message DATA BASE IS DAMAGED if this database is damaged.

12 is a threshold number of pages for File 8 (the Rollback Log). This number is calculated from the percentage shown in Field 13. If the threshold is exceeded, the Rollback Log will be reset. If rollback is disabled, the message ROLLBACK LOG NOT ENABLED appears on this line.

13 is a threshold percentage of pages used (Field 12) divided by total pages in File 8. If this threshold is exceeded, the Rollback Log will be reset. The default is 50%.

14 is a threshold number of synchpoints. If the number of synchpoints encountered exceeds this threshold, the Rollback Log will be reset. The default is 999,999.

15 indicates active update logging. If update logging is not activated, the message UPDATE LOG NOT ACTIVE appears on this line.

16 is the volume serial number of the Keepfile.

17 is the maximum number of entries that the multiple holds buffer can contain. If multiple local holds are not allowed, the message MLH NOT ENABLED appears on this line.

18 is the current number of multiple local holds.

DBNU= Command

The DBNU= command displays users accessing a given database in the Multi-User environment. DBNU= requires one parameter, the name of a database.

#DBNU=*database*

The following information is displayed for the specified database:

```
DUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUDUD
-DBNU= 1
 2          2          2          2          2
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><
```

1 displays the database name, the current cycle number, and the date and time of the last update.

2 is the userid of a user who has this database open.

DBSTAT= Command

The DBSTAT= command displays the various database permissions currently in effect for the specified database. DBSTAT= requires one parameter, the name of a database.

#DBSTAT=*database*

The output shows the following information:

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
-DBSTAT= 1
          SCF  PLEX  RETV  SCF-T  LHSES  LHOLD
COUNT      2     3     4     5      6      7
--------------------------------------------------
     LOCK    ROLBAK     HOLD     UPDAT  NKUPD
USER    8         9       10        11    12
--------------------------------------------------
    UT   U  HT   H  RT   R  SS   F  SO  PO  RB   L  LS  LH  NK
PEND13  14  15  16  17  18  19  20  21  22  23  24  25  26  27
  28             29  30                  28          29  30
<><><><><><><><><><><><><><><><><><><><><><><><><
```

1 displays the database name, the current cycle number, and the date and time of the last update.

2 is a count of SCF users who have this database open.

3 is a count of PLEX users who have this database open.

4 is a count of users doing retrievals.

5 is a count of SCF users currently in a thread.

6 is a count of users attempting to initialize a local hold session.

7 is a count of users who have a local hold.

8 is the userid that has exclusive use of this database. (The PLEX syntax is OPEN/LOCK.)

9 is the userid that has rollback permission on this database, that is, the userid that triggered rollback processing.

10 is the userid that has a global hold on this database.

11 is the userid that has update permission on this database.

12 is the userid that has non-key update permission on this database.

13 is Y if a user is in a thread waiting for update permission.

14 is Y if a user is in the suspend queue waiting for update permission.

15 is Y if a user is in a thread waiting for global hold permission.

16 is Y if a user is in the suspend queue waiting for global hold permission.

17 is Y if a user is in a thread waiting for retrieval permission.

18 is Y if a user is in the suspend queue waiting for retrieval permission.

19 is Y if an SCF user is waiting to start a session.

20 is Y if a user is in the suspend queue waiting for frame permission.

21 is Y if an SCF user is waiting for permission to open this database.

22 is Y if a PLEX user is waiting for permission to open this database.

23 is Y if a user is waiting for permission to roll back this database.

24 is Y if a user is waiting to open this database for exclusive use (or OPEN/LOCK in PLEX).

25 is Y if a user is waiting for local hold session permission prior to obtaining a local hold.

26 is Y if a user is waiting to establish a local hold on this database.

27 is Y if a user is waiting for non-key update permission on this database.

28 is the userid that has at least one local hold on this database.

29 is a unique value (logical address) assigned to each record in a database. The user identified in Field 28 either has a local hold on this record (and possibly others) or is waiting for a local hold on this record (depending on the status of Field 30).

30 is an asterisk (*) or a blank. An asterisk means the userid in Field 28 is waiting for a local hold on the record identified in Field 29. A blank means the record identified by Field 29 is the last record held by the userid identified in Field 28.

HELP Command

The HELP command displays the new MUSTATS (Multi-User status) commands.

```
#HELP
```

The output from the HELP command is shown below.

```
::::::::::::::::::::::::::::::::::::::::::::::::::::
 BUFFERS -- DISPLAY INFORMATION ABOUT BUFFERS
 DBNS    -- DISPLAY OPEN DATA BASE NAMES
 DBN=    -- DISPLAY INFORMATION ABOUT A DATA BASE
 DBNU=   -- DISPLAY USERS ON A DATA BASE
 DBSTAT= -- DISPLAY DATA BASE PERMISSIONS/HOLDS
 MLH=    -- DISPLAY LOCAL HOLDS ON A DATA BASE
 PADS    -- DISPLAY SCRATCH PAD INFORMATION
 POOLS   -- DISPLAY BUFFER POOL USAGE
 QUEUES  -- DISPLAY QUEUE LENGTHS
 THREADS -- DISPLAY THREAD USAGE
 USER=   -- DISPLAY INFORMATION ABOUT A USER
 WHY=    -- DISPLAY REASON FOR USER WAIT
<><><><><><><><><><><><><><><><><><><><><><><><><><
```

MLH= Command

The MLH= command displays the maximum number of multiple local holds allowed and the number of local holds currently in effect for all users of a specified database. The output also shows a list of the records held and the userid for each record held. MLH= requires one parameter, the name of a database.

```
#MLH=database
```

The output shows the following information:

```
-------------------------------------------------------
-MLH= 1
   MAX =      2    CURRENT =      3
        4   5            4   5          4   5
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
```

1 is the name of the specified database, the current cycle number, and the date and time of the last update.

2 is the maximum number of multiple local holds that can be contained in the buffer assigned to hold local hold information for this database. If the maximum is exceeded, PLEX users receive Return Code 110.

3 is the number of local holds currently in effect by all users of the specified database.

4 indicates a record that has a local hold on it. This field contains a unique number (logical address) assigned to each record in the database. Field 5 shows the userid with the hold on this record.

5 is the userid of the user holding the record identified in Field 4.

Note: If the multiple local holds option is disabled for this database, the message MULTIPLE LOCAL HOLDS NOT ACTIVE appears.

PADS Command

The PADS command displays information about scratch pads. This command has no parameters.

```
#PADS
```

The output shows the following information for each scratch pad defined:

```
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
-PADS
  PAD1  USE         2          3%  HI       4        5%
     BLKSIZE    # AU'S   BLKS/AU      AU'S   PAGES
        6          7        8           9     10
     PRIMARY    SECONDARY  UNIT        EXT
       11         12        13          14
     DEVICE     TOTAL  CYL  T/C         P/T
       15         16        17          18
<><><><><><><><><><><><><><><><><><><><><><><><><><
```

1 is the scratch pad number.

2 is the number of allocation units in use.

3 is the highest percentage of space ever used for this scratch pad.

4 is the highest number of allocation units ever used for this scratch pad.

5 is the percentage of the whole scratch pad that one allocation unit represents.

6 is the block size for scratch pads.

7 is the number of allocation units specified for the scratch pad.

8 is the number of blocks per allocation unit.

9 is the number of allocation units formatted. This field is the same as Field 7 unless secondary extents are used.

10 is the number of pages formatted.

11 is the primary allocation, expressed in the units shown in Field 13.

12 is the secondary allocation, expressed in the units shown in Field 13.

13 is the type of unit for Field 11 and Field 12, that is, CYL for cylinders, TRACK for tracks, BLOCK for blocks, or UNKNWN for unknown.

14 is 0 or 1. A zero means no secondary extents have been used. A one means one (or more) secondary extents have been used.

15 is the device assigned to this scratch pad.

16 is the total number of cylinders on this device.

17 is the number of tracks per cylinder for the device assigned to this scratch pad.

18 is the number of pages per track.

POOLS Command

The POOLS command displays information about pool definitions and usage. This command has no parameters.

#POOLS

Each line of output displays information about one defined pool.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-POOLS #    BLK   NUM FREE USE      SIZE   RPM   WPM   DPM
       1      2     3    4   5         6     7     8     9
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
```

1 is the pool number of a pool defined in the S2KPARMS file.

2 is the block size of the buffers in the pool.

3 is the number of buffers defined for the pool.

4 is the number of unassigned buffers in the pool.

5 is the usage defined for the buffers in the pool. Position 1 is S, D, or B for scratch files, database files, or both. Position 2 is either E for exact fit only or a blank for accepting a request for a buffer of this size or smaller.

6 is the size of the pool in bytes, that is, block size (Field 2) times the number of buffers in the pool (Field 3).

7 is the rate of reads-per-minute to the pool if XBUF software is enabled.

8 is the rate of writes-per-minute to the pool if XBUF software is enabled.

9 is the rate that buffers are being reused in the pool (drops-per-minute) if XBUF software is enabled.

Note: Fields 7, 8, and 9 contain blanks if the XBUF software is not enabled. If the rate is too old, asterisks appear.

QUEUES Command

The QUEUES command shows how many users are in the four Multi-User work queues. This command has no parameters.

```
#QUEUES
```

The output contains the information shown below.

```
-------------------------------------------------
-QUEUES SVC =  1 INT = 2  SUSPEND =  3 I/O =  4
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
```

1 is a count of users in the SVC queue. These users have not been serviced by Multi-User software because of a shortage of CPU cycles or threads.

2 is a count of users in the internal dispatch queue. These users have completed an I/O operation or were suspended, waiting for database permission; they have now been acquitted to request the permission again. The users in this queue have not been serviced by Multi-User software because of a shortage of CPU cycles or threads.

3 is a count of users who have been suspended because the database permission that they requested could not be granted. The users in this queue will be moved to the internal dispatch queue when the permission they require can be granted.

4 is a count of users who are waiting for an I/O operation to complete.

THREADS Command

The THREADS command displays current and high thread usage. This command has no parameters.

```
#THREADS
```

The output contains the information shown below.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
-THREADS 1-32   12345678901234567890123456789012
  HIGH( 1)      2................................2
 THREADS 33-63  34567890123456789012345678901 23
                2................................2
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
```

1 is the thread number of the highest thread that the Multi-User software attempted to schedule.

2 shows current thread usage.

- C indicates the current thread, which is very often the S2OP command.
- E indicates a thread that has an assigned user with the user's request executing.
- A hyphen (-) means that the thread is defined but not in use.

USER= Command

The USER= command displays information about a specified user. USER= requires one parameter, a userid.

```
#USER=userid
```

The output contains the information shown below.

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
-USER= 1          2           3          4
  5         6         7    /    8
 ->DEP(   9  )   ->MU(  10  )      CC= 11
 QUEUE= 12      THREAD= 13      AGE= 14 15
                                      (optional line: see Note 1)
 CALLS= 16         IO= 17       DB#= 18
                                      (optional line: see Note 2)
 ETIME= 19         CPU= 20            (if Accounting Log active)
 DATE= 21     V= 22    23    PGM= 24  (if PLEX job)
 OP= 25   RC= 26   MAX= 27    28      (if PLEX job)
                                      (optional line: see Note 3)
   29                                 (if PLEX job)
     30                               (if PLEX job)
31
32 31                                 (if SCF/TP or ALTCON)
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
```

1 is the user's job name.

2 is the user's step name.

3 is the user's program name.

4 is the user's terminal name.

5 is the type of user job, that is, PLEX, SCF, SCF/TP, TPI, SAS, or ALTCON. ALTCON is an alternate console for Multi-User operator commands. TPI is SYS2KTPI.

6 is TSO if the user is running under TSO.

7 is ACTIVE REQUEST if the user is currently presenting a request to Multi-User software for processing; set by the interface that resides in the dependant region.

8 indicates the user's status within the Multi-User system. This field contains EXECUTING, INACTIVE, NO ACTIVITY, or U.P.ACTIVITY (which means the user is waiting for a response in the dependent region to some request that Multi-User software has made); set by Multi-User software.

9 indicates what the user is doing or what the Multi-User software needs to communicate to the user. This field contains the code (interpreted) in the user's Event Control Block (ECB), for example, UNUSED, WAITING, CMD DONE.

10 either indicates what the user expects of Multi-User software or is the user's response to Multi-User software's last user partition request. This field contains the last code (interpreted) that the user sent to Multi-User software, for example, CMD DONE.

11 is the condition code (hexadecimal) if the user has been terminated.

12 indicates which work queue the user is on, that is, SUSPEND, SVC, INTERNAL, I/O, or NONE.

13 is the number of the thread assigned to this user.

14 is the count of users who were dispatched prior to this user being placed on a queue. This field is a relative measure of when this user was placed on the queue.

15 is the resource that this user is waiting for if the user is on the suspend queue.

16 is the number of calls that this user has presented to Multi-User software.

17 is the number of I/Os performed for this user.

18 is the number of databases that this user has open.

19 is the elapsed time since this user job identified itself to Multi-User software. The line containing this field appears only if the Accounting Log is enabled.

20 is the amount of CPU time charged to this user (for PLEX users). The line containing this field appears only if the Accounting Log is enabled.

21 is the date and time that the PLEX program was processed by the PLEX processor. This field pertains to PLEX jobs only.

22 is the SYSTEM 2000 version number of the PLEX processor that processed this PLEX program.

23 indicates which language was used in this PLEX program, PLI, PLIF, COB, FORT, or ASM, for PL/I, PL/I, COBOL, FORTRAN or Assembler, respectively. This field pertains to PLEX jobs only.

24 is the alternate interface name that was recognized by the PLEX processor, if present. This field pertains to PLEX jobs only.

25 is the operation code (in hexadecimal) of the last command presented to the Multi-User software. It comes from the S2KDUM control block in the user's PLEX program. This field pertains to PLEX jobs only.

26 is the last return code returned to S2KDUM in the user's PLEX program. This field pertains to PLEX jobs only.

27 is the number of Locate Files requested by the PLEX program in the START S2K command. This field pertains to PLEX jobs only.

28 is the last PLEX command sent to Multi-User software by the PLEX program. This field is the interpreted form of the operation code shown in Field 25. For example, if Field 25 is 50 (hexadecimal), Field 28 contains MODIFY. This field pertains to PLEX jobs only.

29 is the database name for the last PLEX command, taken from the COMMBLOCK in the PLEX program. This field pertains to PLEX jobs only.

30 is the subschema name for the last PLEX command, if present. This field pertains to PLEX jobs only.

31 is the first 48 bytes of the last SCF command presented to Multi-User software. If the job is SCF/TP or ALTCON, this field is preceded by Field 32 (the terminal ID of the user).

32 is the terminal ID of the user if the job is SCF/TP or ALTCON.

Note 1: After the QUEUE= line, an additional line might appear saying TERM IN EXUSER, TERM IN STAE, or CANCELLATION PENDING.

Note 2: If PQA=PRTY is specified in the SYSTEM 2000 execution parameters, the line shown below appears after the CALLS= line.

```
PQA=PRTY      JOB PRTY= x     DISP= y          ADJ= z       -
```

where

- *x* is the user's current priority.
- *y* is the number of dispatches since the user was put on the queue.
- *z* is the amount the user's priority has been raised due to excessive time on the queue.

Note 3: If the operation code that Multi-User software is processing differs from the one in S2KDUM, an additional line appears after the OP= line for PLEX jobs only.

```
PIOP= opcode   text
```

The operation code displayed is in hexadecimal. The text is the English equivalent of the opcode.

For example, suppose that PLEX inserts are queued. The next command on a different stack causes the inserts to be processed. In this situation, PIOP is the INSERT opcode, but OP in the previous line of output is the command in the other stack.

User swapped out For an SCF/TP user who is swapped out to the S2KUSERS file, the following output appears for the USER= command:

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
-USER= 1    2        SCF/TP
  DBN= 3
                   4
<><><><><><><><><><><><><><><><><><><><><><><><><><
```

1 is the user's terminal ID.

2 is the user's job name.

3 is the current database name.

4 can be one of the following messages:

```
FORCED EXIT
USER HOLDS LOCKS
SWAP IN PROGRESS
CANCEL REQUEST
```

Userid not found If the userid that you specified in the USER= command cannot be found, the following output appears:

```
/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
-USER= 1                       USER NOT FOUND
<><><><><><><><><><><><><><><><><><><><><><><><><><
```

1 is the user's job name.

WHY= Command

The WHY= command displays the reason for this user wait. WHY= requires one parameter, the userid of a user who is waiting.

```
#WHY=userid
```

The following wait information is displayed for the specified user:

```
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
-WHY=  1          2                3          4
     5                         6
<><><><><><><><><><><><><><><><><><><><><><><><><><><><><
```

1 is the userid that is waiting.

2 is the step name.

3 is the program name, if available.

4 is the type of permission for which this userid is waiting.

5 is the reason why the userid is waiting for the permission shown in Field 4.

6 is more information about the reason (Field 5), such as a userid or a count of local holds.

THE MULTI-THREAD ENVIRONMENT

With Multi-Thread, Multi-User processes several users concurrently using several threads. Multi-Thread provides concurrent access with interleaving at the resource request (I/O) level. With more than one thread, several SYSTEM 2000 commands can be processed concurrently.

Multi-Thread improves performance for installations where teleprocessing or batch demand rates exceed the throughput capabilities of a single-thread Multi-User system. Multi-Thread queues transaction input and user output and controls the concurrent processing of multiple SYSTEM 2000 updates and retrievals. The number of threads used (specified in the THREADS execution parameter) is based on demand rate, main memory availability, and peripheral storage considerations.

Multi-Thread makes optimal use of resources, the most critical resource being time. A user (or thread) is given control of the system until he requests an I/O or a resource that is not readily available. At that time, the user relinquishes control to another user (thread) and waits for the I/O or resource request to be satisfied. Examples of requests that cannot be satisfied immediately are those involving external devices, user tasks, or sharable SYSTEM 2000 resources. Polling of both the user and thread lists is done in a round-robin fashion dependent upon shared-resource contentions. Therefore, the polling does not favor entries in the top of the list. Safeguards against deadlock are also incorporated in the Multi-Thread logic.

SYSTEM 2000 software algorithms are written as reentrant code that can be shared by up to 63 concurrent processing threads. Multi-User provides control over user-request scheduling, thread scheduling, and resource and memory allocation. The system also provides operator and system communication. Some of the benefits of this approach are protection against destructive concurrent updates of a database, economies in memory usage due to pooling and sharing of buffers among databases, and monitoring of database activity.

Multi-Thread scheduling philosophy is based on pseudo-interrupts issued by SYSTEM 2000 software rather than on machine interrupts of MVS subtasking. Pseudo-interrupts provide minimum control program overhead and facilitate SYSTEM 2000 control of shared resources.

Multi-Thread requires no external syntax for SCF jobs or PLEX programs. The maximum number of threads, users, buffers, shared scratch files, and concurrent databases are established at Multi-User initialization time using the execution parameters. Default values can be overridden by JCL PARM statements or by the operator when Multi-User is initialized.

Single-user jobs can also run along with Multi-User as long as they do not attempt to access a database being used in the Multi-User environment.

JCL for Multi-Thread

Because Multi-Thread is internal to SYSTEM 2000 software and invisible to the user, the JCL considerations are the same as those for Multi-User, except a set of SF files is required for each thread.

Specifying Several Threads: THREADS Parameter

When Multi-User is initialized, the THREADS parameter specifies the number of threads that will be available. The default is single thread (THREADS=1). (See **THREADS Parameter** on page 3-62.) The size of each thread depends on whether the REPORT processor has been selected. The REPORT processor requires 9K bytes more per thread area than normal.

SCF and PLEX Job Execution with Multi-Thread

SCF jobs and PLEX programs execute under Multi-Thread in the same manner as under Multi-User, except for the multiple threads that are invisible to the user. The SCF $LOCK command locks only the thread that the user is using. Other threads are not affected.

OBTAINING RESOURCE STATISTICS: ACCOUNTING LOG

You can obtain resource usage statistics by instructing SYSTEM 2000 software to write accounting information on the Accounting Log.

The Accounting Log includes statistics on CPU time and I/O count used by each SYSTEM 2000 session and each Multi-User segment. You can store the Accounting Log file on either tape or disk. When the file resides on disk, you must use the ACTUTIL utility to dump the file contents to a sequential file in secondary storage.

This section provides the information you need to use the Accounting Log, as outlined here.

- **Records in the Accounting Log File** on page 3-104 describes the various types of records that can appear in the Accounting Log file.
- **Using the Accounting Log** on page 3-106 presents general considerations for using the Accounting Log.
- **Using ACTUTIL to Build the Disk Data Sets** on page 3-107 describes how to use the ACTUTIL utility to build two disk data sets for the Accounting Log file.
- **Using ACTUTIL to Dump the Disk Data Sets** on page 3-108 describes how to use the ACTUTIL utility to dump the contents of a disk data set to a sequential file.
- **Generating Accounting Data: SYSTEM 2000 Execution Parameters** on page 3-110 describes how to set the ACCT, NLSEG, PLSEG, and TPSEG execution parameters to control what data are written to the Accounting Log.
- **Format of Accounting Log Records** on page 3-115 details the format of the records in the Accounting Log file. You need this information to write a program that reads those records and then formats the accounting data according to your standards.
- **Exits Used for Changing Fields in Accounting Log Records** on page 3-122 discusses user-written exits for changing the values of jobname, stepname, program name, and user priority that SYSTEM 2000 software writes to the Accounting Log.

Records in the Accounting Log File

The Accounting Log file can contain seven types of records. Each type of record is described in detail in **Format of Accounting Log Records** on page 3-115.

- **Multi-User Initialization** records - contain the Multi-User parameters selected, the time and date of Multi-User initialization, and the Multi-User region size. This record is written whenever Multi-User is initialized.

- **User-termination** records - contain the following data, which are written when a SYSTEM 2000 user session terminates:

  - date and time of start and termination of a user job
  - user's jobname, stepname, and program name[1]
  - terminal id (if TP user)
  - SYSTEM 2000 release number
  - CPU time used
  - database I/Os
  - scratch file I/Os
  - total I/Os for database files, scratch files, sort files, user files, and so forth
  - Multi-User region size
  - user priority.

- **Multi-User Termination** records - contain the time and date when Multi-User terminates.

- **Multi-User Segment** records - contain a subset of the data in a User-termination record. However, the data pertain to a specific Multi-User segment.

  These records are written whenever your specified CPU-time or I/O-count thresholds are met.

  You can request information on PLEX segments, batch SCF segments, and SCF TP segments.

---

[1] You can change the values for jobname, stepname, program name, and user priority by invoking user-written exits. See **Exits Used for Changing Fields in Accounting Log Records** on page 3-122 for details.

- **Lost-data** records - contain the number of Accounting Log records that are lost when the two disk data sets allocated to the Accounting Log are full.

  When the Accounting Log is implemented on disk, two data sets are set up to receive the accounting data. As each data set is filled, Multi-User issues a message indicating that the ACTUTIL utility should be executed to dump the contents of that data set to a sequential file in secondary storage.

  If both data sets are full, Multi-User cannot write accounting data on them and the accounting data are lost. However, Multi-User keeps track of how many records with accounting data are being lost. After you execute ACTUTIL to dump one of the data sets to a sequential file, Multi-User first writes a Lost-data record.

- **Header** record - contains a date and time stamp indicating when ACTUTIL started dumping the contents of a data set to a sequential file in secondary storage.

- **Trailer** record - contains a date and time stamp indicating when ACTUTIL finished dumping the contents of a data set to a sequential file in secondary storage.

Using the Accounting Log

As mentioned at the beginning of this section, the Accounting Log can be implemented on tape or disk.

If you implement the Accounting Log on tape, you need to specify only the execution parameters that determine what accounting statistics are compiled. Multi-User writes the records with accounting data directly to a sequential file on tape. When that file becomes full, Multi-User stops and does not resume writing until the operator satisfies the operating system request for a new tape. Tape implementations, however, have the serious drawback of requiring a dedicated tape drive. Therefore, the next three sections focus on implementing the Accounting Log on disk, which is the preferred practice. The discussion on execution parameters in **Generating Accounting Data: SYSTEM 2000 Execution Parameters** on page 3-110 applies to both implementations.

Note that if you decide to use the Accounting Log, you must add the needed DD statements to the JCL that executes Multi-User.

If the Accounting Log is implemented on disk, you need

```
//S2KMANX   DD   DSN=dsnacl-X,DISP=SHR
//S2KMANY   DD   DSN=dsnacl-Y,DISP=SHR
```

where

dsnacl is the data set name for the Accounting Log data set described in **Using ACTUTIL to Build the Disk Data Sets** on page 3-107.

If the Accounting Log is implemented on tape, you need

```
//S2KMANX   DD   DSN=dsnacl,DISP=(NEW,CATLG),
//                DCB=BLKSIZE=blksize,UNIT=TAPE
```

where

dsnacl is the data set name for the Accounting Log data set described in **Using ACTUTIL to Build the Disk Data Sets** on page 3-107.

To calculate the block size, use this formula.

```
BLKSIZE = (252 x blocking-factor) + 4
```

That is, the BLKSIZE equals the LRECL of the largest record times the blocking factor plus 4 bytes for the block descriptor word.

Using ACTUTIL to Build the Disk Data Sets

Before Multi-User can write accounting data to a disk data set, you need to execute ACTUTIL to allocate and initialize two BDAM data sets. Two data sets are required to allow Multi-User to write data on one data set while data on the other data set are being dumped to secondary storage. The DDnames for the two data sets are S2KMANX and S2KMANY.

Use the following JCL to build the data sets for the Accounting Log. It must be executed once for each data set.

```
//BUILD     EXEC  PGM=ACTUTIL,
//                PARM='FUNCTION=BUILD,NBLOCKS=nbl'
//STEPLIB   DD    DSN=dsns2k,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//ACCTFILE  DD    DSN=dsnacl,DISP=(NEW,CATLG,DELETE),
//                UNIT=SYSDA,VOL=SER=ser
//                DCB=BLKSIZE=blszacl,
//                SPACE=(blszacl,nbl)
```

where

nbl is the number of blocks to be initialized.

dsns2k is the data set name of the SYSTEM 2000 load library.

dsnacl is the data set name of the Accounting Log data set being initialized.

ser is the volume serial number of the disk.

blszacl is the block size of the Accounting Log file.

Notice that secondary space allocations are not allowed.

When determining the number of blocks to initialize for the data sets, keep in mind that if the data sets are not large enough, their contents will have to be dumped more frequently to prevent loss of accounting data. Accounting data are lost when the two data sets become full and Multi-User cannot find space in them to write new accounting data.

When determining what block size to specify, consider these factors:

- A small block size means more I/Os to the disk but fewer records will be lost if the system fails.
- A large block size means fewer I/Os to the disk but more records will be lost if the system fails.
- The types of Accounting Log records that will be written to the disk. The most common records will be either the User-termination records with an LRECL of 104 bytes or the Multi-User segment records with an LRECL of 252 bytes.
- The track size of the disk. You want to minimize wasted space on the disk.

Because the Accounting Log file consists of variable length records and because the largest record size is 252 (LRECL=252), the block size must be larger than 256, (that is, 252 + 4 bytes for the block descriptor word).

In the following sample JCL, suppose no Multi-User segment accounting is being logged and, therefore, the most common record is the User-termination record (LRECL=104). The block size of 1044 allows for 10 records per block (10 x 104 + 4 bytes for the block descriptor word) and wastes little space on a 3350 disk track.

```
//BUILD     EXEC  PGM=ACTUTIL,
//                PARM='FUNCTION=BUILD,NBLOCKS=1000'
//STEPLIB   DD    DSN=S2K.PROD.LOAD,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//ACCTFILE  DD    DSN=S2K.PROD.ACTMANX,DISP=(NEW,CATLG,DELETE),
//                UNIT=SYSDA,VOL=SER=DISK05,
//                DCB=BLKSIZE=1044,
//                SPACE=(1044,1000)
```

See *SYSTEM 2000 Messages and Codes, Version 12, First Edition* for information on the WTO and SYSPRINT messages that can be issued when executing ACTUTIL.

Using ACTUTIL to Dump the Disk Data Sets

After Multi-User fills one disk data set (for example, S2KMANX) with accounting data, Multi-User automatically starts writing on the other data set and issues two WTO messages:

```
S2K1305/sid- S2KMANX DATA SET FULL- PLEASE DUMP
S2K1307/sid- NOW RECORDING ACCOUNTING DATA ON S2KMANY -
```

sid is the system identifier specified in the SID execution parameter described in **SID Parameter** on page 3-61.

The corresponding messages for S2KMANY are S2K1304 and S2K1307.

After WTO message S2K1304 (or S2K1305) appears, the operator must execute ACTUTIL to dump the data to secondary storage. If the data are not dumped, Multi-User issues these messages:

```
S2K1311/sid- NO AVAILABLE ACCOUNTING LOG FILES -
S2K1309/sid- ACCOUNTING DATA BEING LOST DUE TO FULL
                 ACCOUNTING LOG FILE- PLEASE DUMP
```

After issuing these messages, Multi-User stops writing accounting records on the file but keeps a tally of how many records are being lost. After the data are dumped, Multi-User writes a Lost-data record on whichever data set is available, S2KMANX or S2KMANY.

To prevent loss of data, dump each data set as it becomes full. That is, dump after WTO message 1304 (or 1305) appears. However, do not dump an active data set (one that Multi-User is writing on) because unpredictable results can occur. If you attempt to dump an active data set, ACTUTIL issues a warning WTO message.

After you dump the data, you can execute a program to process that data. Someone at your site must write this program so that the output meets your specific requirements. Or, you can produce reports and graphics with the tuning tools provided by the Institute; for details, see *Technical Report: S2-106 Multi-User Tuning Tools for SYSTEM 2000 Software, Release 11.6 under IBM OS and CMS*.

The DUMPACTIVE parameter for the ACTUTIL utility program allows you to specify whether the active Multi-User Accounting Log should be dumped. However, if the DUMPACTIVE parameter does not equal YES, Y, NO, or N, the usual WTO message 2202 appears, prompting the console operator to give a reply of Y or N.

```
DUMPACTIVE= |YES
            |NO
```

where

YES means the active Accounting Log should be dumped.

NO means the active Accounting Log should not be dumped.

With Release 11.6 and later releases, the ACTUTIL program no longer needs an operator reply to determine which action to take for dumping the active Accounting Log when it is full. (The DUMPACTIVE parameter replaces Special Zaps #174 and #149.)

Use the following JCL to dump the data from S2KMANX (or S2KMANY) to a sequential file in secondary storage:

```
//DUMP      EXEC  PGM=ACTUTIL,
//                PARM='FUNCTION=DUMP'
//STEPLIB   DD    DSN=dsns2k,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//ACCTFILE  DD    DSN=dsnacl,DISP=SHR
//DUMPFILE  DD    DSN=dsndump,DISP=(NEW,CATLG,DELETE),
//                UNIT=type,
//                DCB=(LRECL=252,BLKSIZE=blksize,RECFM=VB)
```

where

dsns2k is the data set name of the SYSTEM 2000 load library.
dsnacl is the data set name of the Accounting Log data set being dumped.
dsndump is the data set name of the receiving sequential file.
type is the tape or disk unit type of the sequential file.
blksize is the block size of the sequential file.

To calculate the block size, use the following formula:

BLKSIZE = (252 x *blocking-factor*) + 4

That is, the LRECL of the largest record times the blocking factor plus four bytes for the block descriptor word.

Abbreviations: YES = Y
NO = N

The following sample JCL uses a blocking factor of 25:

```
//DUMP      EXEC  PGM=ACTUTIL,
//                PARM='FUNCTION=DUMP'
//STEPLIB   DD    DSN=S2K.PROD.LOAD,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//ACCTFILE  DD    DSN=S2K.PROD.ACTMANX,DISP=SHR
//DUMPFILE  DD    DSN=ACCTDUMP,DISP=(NEW,CATLG,DELETE),
//                UNIT=TAPE,
//                DCB=(LRECL=252,BLKSIZE=6304,RECFM=VB)
```

See *SYSTEM 2000 Messages and Codes, Version 12, First Edition* for information on the WTO and SYSPRINT messages that can be issued when executing ACTUTIL.

Generating Accounting Data: SYSTEM 2000 Execution Parameters

ACCT Parameter After you have initialized the two data sets, you must set the ACCT execution parameter to YES so that Multi-User will start writing accounting data to the Accounting Log.

```
ACCT=|NO          (default)
     |YES
```

YES generates accounting data. NO (the default) stops generating accounting data. The YES or NO status of the SYSTEM 2000 ACCT execution parameter cannot be changed from the console.

When ACCT=YES, the following Accounting Log records are generated:

- Multi-User Initialization records
- User-termination records
- Multi-User Termination records.

NLSEG, TPSEG, and PLSEG Parameters If you also require Multi-User segment accounting data, you must specify one or more of the following three execution parameters:

NLSEG to obtain accounting data for batch SCF segments.

TPSEG to obtain accounting data for SCF TP segments.

PLSEG to obtain accounting data for PLEX Multi-User segments.

You can control how much accounting data are generated for Multi-User segments by specifying, as execution parameter options, CPU-time and I/O-count thresholds.

See **Setting SYSTEM 2000 Execution Parameters** on page 5-9 for information about ways to set up SYSTEM 2000 execution parameters.

The syntax of NLSEG, PLSEG, and TPSEG also allows you to delay the generation of segment data until Multi-User receives a command from the console. Also, the generation of segment accounting data can be interrupted or stopped from the console.

```
|NLSEG =CPU-time/IO-count[/DELAY]
|PLSEG
|TPSEG
```

CPU-time is the threshold time in hundredths of a second, specified as an integer in the 0 to 999999 range. Multi-User produces accounting data for segments that meet the IO criterion and whose CPU-time equals or exceeds the specified time.

If 0 is specified, accounting data for all segments are written if the IO-count criterion is met. If NO is specified, the CPU-time threshold is not used as a criterion.

IO-count is the threshold number of database I/Os specified as an integer in the 0 to 999999 range. Multi-User produces accounting data for segments meeting the CPU-time criterion and whose I/O count equals or exceeds the specified number.

If 0 is specified, accounting data for all segments are written if the CPU-time criterion is met. If NO is specified, the I/O-count threshold is not used as a a criterion.

DELAY means that Multi-User segment accounting is delayed until a START request is received from the console.

If DELAY is not specified, Multi-User segment accounting starts immediately.

You can also issue an operator console command to control segment data generation. Here is the syntax for that command:

```
|MODIFY jobname,     |START  STAT      |NL
|F                   |STOP             |PL
                                       |TP
                                       |ALL
```

jobname refers to the SYSTEM 2000 job name.

START tells Multi-User to start writing segment accounting data on the Accounting Log file. This applies if the writing was either delayed because DELAY was specified in the execution parameter or stopped because of a previous STOP specification.

STOP tells Multi-User to stop writing segment accounting data on the Accounting Log file. STOP applies if the writing started either because DELAY was not specified in the execution parameter or because of a previous START.

NL controls writing of accounting data for batch SCF segments.

PL controls writing of accounting data for PLEX Multi-User segments.

TP controls writing of accounting data for SCF TP segments.

ALL controls writing of accounting data for all three types of Multi-User segments.

OPT004 Parameter The OPT004 execution parameter includes system-related time in Types 5, 6, and 8 records.

```
OPT004=|NO      (default)
       |YES
```

YES includes SRB (system-related) time with PRB (problem-related) time in Accounting Log Type 5, 6, and 8 records. NO (the default) means that only PRB time is specified in the Accounting Log records. (OPT004 replaces Special Zap #024.)

OPT031 Parameter The OPT031 execution parameter includes the number of user calls in Type 4 records.

```
OPT031=|NO      (default)
       |YES
```

YES places the number of SYSTEM 2000 calls by a user in the user termination record (Accounting Log Type 4 record). The number of calls replaces the SYSTEM 2000 region size field. NO (the default) means the user termination records (Type 4) are not altered. (OPT031 replaces Special Zap #131.)

OPT036 Parameter The OPT036 execution parameter includes the step and program names in Type 6 records.

```
OPT036=|NO      (default)
       |YES
```

YES places the step name and program name in Accounting Log MU segment records (Type 6). NO (the default) does not place the step name or the program name in Type 6 records. (OPT036 replaces Special Zap #190.)

OPT037 Parameter The OPT037 execution parameter places the synchpoint ID in the ACCTALT field of the user's URB.

```
OPT037=|NO      (default)
       |YES
```

YES places the synchpoint ID for a logical unit of work (LUW) in the ACCTALT field of the user's URB. NO (the default) means the ACCTALT field of the user's URB is not affected. (OPT037 replaces Special Zap #195.)

Examples: Execution Parameters that Affect the Accounting Log

The following examples illustrate the use of the execution parameters and the console commands affecting the Accounting Log.

Example 1

If you specify these parameters, no accounting statistics are written to the Accounting Log file.

```
ACCT=NO
NLSEG=100/100
PLSEG=500/500
```

Example 2

If you specify these parameters, Multi-User writes Multi-User Initialization records, User-termination records, and Multi-User Termination records. OPT031=YES places user calls in the user-termination records.

```
ACCT=YES
OPT031=YES
```

Example 3

If you specify these parameters, Multi-User writes Multi-User Initialization records, User-termination records, and Multi-User Termination records.

```
ACCT=YES
TPSEG=100/200
PLSEG=500/600
OPT037=YES
```

Records for SCF TP segments are written only for those segments whose CPU-time exceeds one second (100 hundredths of a second) and whose I/O-count exceed 200. Records for PLEX segments are written only for those segments whose CPU-time exceeds five seconds and whose I/O-count exceeds 600. OPT037 places the synchpoint ID for an LUW in the ACCTALT field of the user's URB.

Example 4

If you specify these parameters, Multi-User writes the same records as in Example 2.

```
ACCT=YES
TPSEG=100/200/DELAY
PLSEG=500/600
OPT031=YES
```

However, records for SCF TP segments are not written until the following console command is issued:

```
F S2KMU,START STAT TP
```

Example 5

If you specify these parameters, Multi-User writes Multi-User Initialization records, User-termination records, and Multi-User Termination records. Records for SCF TP segments are written only for those segments whose CPU-time exceeds one second, regardless of their I/O-count. Records for PLEX segments are always written.

```
ACCT=YES
TPSEG=100/NO
PLSEG=0/0
```

Format of Accounting Log Records

This section describes each of the seven types of records that can be written to the Accounting Log file.

Multi-User Initialization Record

Offsets Dec	(Hex)	Length	Format	Contents
0	(0)	2	binary	Record length
2	(2)	2	binary	Reserved
4	(4)	2	binary	Record ID=0
6	(6)	2	EBCDIC	System ID
8	(8)	4	EBCDIC	SYSTEM 2000 release number
12	(C)	4	packed	Date system started: '00YYDDDF', where F is the sign
16	(10)	4	binary	Time system started in hundredths of a second
20	(14)	4	binary	SYSTEM 2000 region size in 1K blocks
SYSTEM 2000 execution parameters				
24	(18)	2	binary	Reserved
26	(1A)	2	binary	THREADS parameter value
28	(1C)	2	binary	AREAS parameter value
30	(1E)	2	binary	SDBS parameter value
32	(20)	2	binary	SDBSIZE parameter value
34	(22)	2	binary	LDBS parameter value
36	(24)	2	binary	LDBSIZE parameter value
38	(26)	2	binary	SID parameter value
40	(28)	4	binary	POOL0 ...
44	(2C)	4	binary	POOL1 :
48	(30)	4	binary	POOL2 :
56	(38)	4	binary	POOL3 size/number
52	(34)	4	binary	POOL4 of buffers
60	(3C)	4	binary	POOL5 :
64	(40)	4	binary	POOL6 :
68	(44)	4	binary	POOL7 ..:
72	(48)	4	binary	COPYAREA1 ...
76	(4C)	4	binary	COPYAREA2 :
84	(54)	4	binary	COPYAREA3 size/number
80	(50)	4	binary	COPYAREA4 of buffers
88	(58)	4	binary	COPYAREA5 :
92	(5C)	4	binary	COPYAREA6 ..:
96	(60)	2	binary	LOGCOUNT parameter value
98	(62)	2	binary	USER parameter value
100	(64)	2	binary	TPTHREADS parameter value
102	(66)	2	binary	TPSCRUN parameter value

104	(68)	2	binary	ACCT parameter value: 1 means YES, 2 means NO
106	(6A)	2	binary	OPI parameter value: 1 means YES, 2 means NO
108	(6C)	2	binary	Reserved
110	(6E)	2	binary	Reserved
112	(70)	2	binary	LOGLEVEL parameter value: Bit Meaning When Set 0 UINIT 1 POPEN 2 TSTRT 3 USEGM 4 USPND 5 TSPND 6 MFLOW 7 TSPIO 8 LOPEN
114	(72)	2	binary	Reserved
116	(74)	2	binary	Reserved
118	(76)	2	binary	RW parameter value: 1 means YES, 2 means NO
120	(78)	2	binary	DISK parameter value: Bit Meaning When Set 1 3380 2 3390 4 3350 8 3375 16 3370 32 3310
122	(7A)	4	binary	Reserved
126	(7E)	2	binary	LIST parameter value: 1 means YES, 2 means NO
128	(80)	2	binary	Reserved
130	(82)	2	binary	TSO parameter value: 1 means YES, 2 means NO
132	(84)	2	binary	Reserved
134	(86)	2	binary	STARTTP parameter value: 1 means YES, 2 means NO
136	(88)	2	binary	FPTPSYS parameter value: 1 means YES, 2 means NO
138	(8A)	2	binary	Reserved
140	(8C)	2	binary	Reserved
142	(8E)	2	binary	Reserved
144	(90)	2	binary	Reserved
146	(92)	2	binary	LHOLD parameter value: 1 means YES, 2 means NO
148	(94)	4	binary	PLSEG parameter CPU-time value: no DELAY - high order bit=0 with DELAY- high order bit=1 for NO, value='7FFFFFFF'

152	(98)	4	binary	PLSEG parameter I/O-count value: no DELAY - high order bit=0 with DELAY- high order bit=1 for NO, value='7FFFFFFF'
156	(9C)	4	binary	TPSEG parameter CPU-time value: no DELAY - high order bit=0 with DELAY- high order bit=1 for NO, value='7FFFFFFF'
160	(A0)	4	binary	TPSEG parameter I/O count value: no DELAY - high order bit=0 with DELAY- high order bit=1 for NO, value='7FFFFFFF'
164	(A4)	4	binary	NLSEG parameter CPU-time value: no DELAY - high order bit=0 with DELAY- high order bit=1 for NO, value='7FFFFFFF'
168	(A8)	4	binary	NLSEG parameter I/O-count value: no DELAY - high order bit=0 with DELAY- high order bit=1 for NO, value='7FFFFFFF'
172	(AC)	2	binary	EXITS parameter value: 1 means YES, 2 means NO
174	(AE)	4	binary	PQA parameter value: for FIFO, value='FFFFFFFF' for PRTY/n/m, value='nnnnmmmm'
178	(B2)	2	binary	CONVERT parameter value: 1 means YES, 2 means NO
180	(B4)	2	binary	CORECOV parameter value: 1 means YES, 2 means NO
182	(B6)	4	binary	PAD00 .....
186	(BA)	4	binary	PAD01 :
190	(BE)	4	binary	PAD02 :
194	(C2)	4	binary	PAD03 :
198	(C6)	4	binary	PAD04 :
202	(CA)	4	binary	PAD05 :
206	(CE)	4	binary	PAD06 :
210	(D2)	4	binary	PAD07 allocation units/
214	(D6)	4	binary	PAD08 blocks per
218	(DA)	4	binary	PAD09 allocation unit
222	(DE)	4	binary	PAD10 :
226	(E2)	4	binary	PAD11 :
230	(E6)	4	binary	PAD12 :
234	(EA)	4	binary	PAD13 :
238	(EE)	4	binary	PAD14 :
242	(F2)	4	binary	PAD15 .....:
246	(F6)	2		Reserved
248	(F8)	4		Reserved

User-termination Record

Offsets Dec	(Hex)	Length	Format	Contents
0	(0)	2	binary	Record length
2	(2)	2	binary	Reserved
4	(4)	2	binary	Record ID=4
6	(6)	2	EBCDIC	System ID
8	(8)	4	EBCDIC	SYSTEM 2000 release number
12	(C)	4	packed	Date user terminated: '00YYDDDF', where F is the sign
16	(10)	4	binary	Time user terminated in hundredths of a second
20	(14)	8	EBCDIC	Job name
28	(1C)	8	EBCDIC	Step name
36	(24)	8	EBCDIC	Program name(Transaction ID for TP)
44	(2C)	8	EBCDIC	Terminal ID (Blanks if not TP user)
52	(34)	2	binary	User priority: 0 if PQA=FIFO
54	(36)	14	EBCDIC	Reserved
68	(44)	4	packed	Date user started: '00YYDDDF', where F is the sign
72	(48)	4	binary	Time user started in hundredths of a second
76	(4C)	4	binary	CPU time used in hundredths of a second
80	(50)	4	binary	Database file I/Os; Files 1 through 8
84	(54)	4	binary	Scratch file I/Os; includes all scratch files, except for scratch and sort file I/Os for sorts
88	(58)	4	binary	TOTAL I/Os, all database, scratch, and sort files.
92	(5C)	4	binary	Reserved
96	(60)	4	binary	SYSTEM 2000 region size in 1K blocks; number of user calls if OPT031 parameter equals YES.
100	(64)	4	binary	Reserved

Multi-User Segment Record

Offsets Dec	(Hex)	Length	Format	Contents
common record area				
0	(0)	2	binary	Record length
2	(2)	2	binary	Reserved
4	(4)	2	binary	Record ID 5 for PLEX 6 for SCF TP 8 for SCF BATCH
6	(6)	2	EBCDIC	System ID
8	(8)	4	EBCDIC	SYSTEM 2000 release number
12	(C)	4	packed	Current date: '00YYDDDF', where F is the sign
16	(10)	4	binary	Start time
20	(14)	4	binary	Stop time
24	(18)	4	binary	Elapsed CPU time in hundredths of a second
28	(1C)	8	packed	Database file I/O count; Files 1 through 8
36	(24)	8	packed	Scratch file I/O count; includes all scratch files except for scratch and sort file I/Os for sorts
44	(2C)	2	binary	User priority; (can be an adjusted priority)
46	(2E)	6	EBCDIC	Reserved
52	(34)	8	EBCDIC	Job name / terminal ID
60	(3C)	8	EBCDIC	Step name (blank for TP)
68	(44)	8	EBCDIC	Program name (blank for TP)
SCF batch and SCF TP record area				
76	(4C)	4	binary	Four-byte length field
80	(50)	172	EBCDIC	Command text (up to 172 characters)
PLEX record area				
76	(4C)	4	binary	Operation code
80	(50)	8	EBCDIC	Database name
88	(58)	30	EBCDIC	Subschema record name
118	(76)	4	binary	Four-byte length field
122	(7A)	130	EBCDIC	DYNAMIC where-clause text (up to 130 characters)

Multi-User Termination Record

Offsets Dec	(Hex)	Length	Format	Contents
0	(0)	2	binary	Record length
2	(2)	2	binary	Reserved
4	(4)	2	binary	Record ID=12
6	(6)	2	EBCDIC	System ID
8	(8)	4	EBCDIC	SYSTEM 2000 release number
12	(C)	4	packed	Date system terminated: '00YYDDDF', where F is the sign
16	(10)	4	binary	Time system terminated in hundredths of a second

Lost-data Record

Offsets Dec	(Hex)	Length	Format	Contents
0	(0)	2	binary	Record length
2	(2)	2	binary	Reserved
4	(4)	2	binary	Record ID=7
6	(6)	2	EBCDIC	System ID
8	(8)	4	EBCDIC	SYSTEM 2000 release number
12	(C)	4	packed	Date of data loss: '00YYDDDF', where F is the sign
16	(10)	4	binary	Time of data loss in hundredths of a second
20	(14)	4	binary	Number of accounting records lost
24	(18)	4	packed	Date when this record was written: '00YYDDDF', where F is the sign
28	(1C)	4	binary	Time when this record was written in hundredths of a second

Header Record

Offsets Dec	(Hex)	Length	Format	Contents
0	(0)	2	binary	Record length
2	(2)	2	binary	Reserved
4	(4)	2	binary	Record ID=2
6	(6)	6	EBCDIC	Reserved
12	(C)	4	packed	Date when this record was written: '00YYDDDF', where F is the sign
16	(10)	4	binary	Time when this record was written in hundredths of a second

Trailer Record

Offsets Dec	(Hex)	Length	Format	Contents
0	(0)	2	binary	Record length
2	(2)	2	binary	Reserved
4	(4)	2	binary	Record ID=3
6	(6)	6	EBCDIC	Reserved
12	(C)	4	packed	Date when this record was written: '00YYDDDF', where F is the sign
16	(10)	4	binary	Time when this record was written in hundredths of a second

Exits Used for Changing Fields in Accounting Log Records

Four fields in the User-termination record can be changed by invoking user-written exits at user-initialization time. The fields that can be changed are the jobname, the stepname, the program name, and the user priority (URBPRTY).

To modify values, the exit(s) must modify four fields in an internal control block called the User Request Block (URB). Within the URB, the length and displacement for each field are as follows:

Name	Length (bytes)	Displacement
jobname	8	X'50'
stepname	8	X'58'
program name	8	X'60'
URBPRTY	1	X'1F'

The jobname, stepname, and program name fields are copied into the Multi-User region after the user exit has had a chance to modify them. They are eventually stored in the appropriate fields of the User-termination records. See **Format of Accounting Log Records** on page 3-115. Prior to modification, the jobname field contains the MVS jobname; the stepname field contains the MVS stepname; and the program name field contains the Program-ID for PLEX jobs and 'SYS2KPGM' for SCF jobs. The URBPRTY field in the URB can be set by the accounting exit regardless of the value of the PQA parameter described in **PQA Parameter** on page 3-59.

The accounting exit is taken one time only, at initialization, for all SCF and PLEX users; it is taken before each command segment for TP users. The user's priority in the Multi-User control blocks is set from this value. Therefore, after the accounting exit has been taken, the user's priority cannot be changed. TP users can change priorities for each command segment, but not during a specific command segment.

If the jobname field in the URB is modified, Multi-User will then recognize a given job by some name other than the one recognized by the host operating system. This situation causes Multi-User console displays and other console commands to produce potentially confusing results.

There are several reasons for considering user exits. Information from the job-accounting fields (jobname, stepname, program name) can be examined and modified, causing that information to be written into the User-termination records for later reporting. Unique user-identification data can be generated in various ways and stored into a portion of some field. If a violation of an installation standard is detected, the user exit can issue a user abend code, and Multi-User would never know the job was active. Note that the user abend code must not be one that is already in use. Those currently in use are described in *SYSTEM 2000 Messages and Codes, Version 12, First Edition*.

Several interfaces to Multi-User are provided by SAS Institute Inc. Individual user exits can be linked into each interface. The interface protocol is identical for each user exit. An external reference to a certain user-exit name is coded in the interface, and before initial communication with Multi-User is established, a test is done to see if a user exit has been linked into the interface. If an exit is present, it is called once using standard MVS linkage conventions. Register 0 contains the address of the URB, and Register 13 contains the address of an 18-word save area for use by the exit. The address of the user's Task Control Block (TCB) is the fourth word of the URB; it can be inspected but not modified. Any working storage needed by the exit must be acquired and released by the exit. Registers that are

altered should be restored. Standard linkage conventions should be used to return control from the exit to the calling routine. When the interface receives control from the exit, the initialization process continues.

User exits available in the CICS interface are described in the *SYSTEM 2000 Interface to CICS (Command Level), Version 12, First Edition*. The initialization-time user exit for SCF is named SCFUEX4 for CICS. For PLEX programs it is named PLXUEX1. General purpose routines present in all interfaces invoke the initialization-time exits: SCFINTF for SCF and PLXINTF for PLEX. Therefore, SCFINTF calls the exit SCFUEX1, and PLXINTF calls PLXUEX1. Each user exit is expected to be a callable CICS command-level program. The operator-command facility is the same as SCF.

The other four Multi-User interfaces are non-overlayed load modules that all customers receive as members of the standard SYSTEM 2000 load library. They are SYS2KJOB, the batch SCF interface invoked directly by JCL; MUPLINT, loaded dynamically at run time by batch or TSO PLEX jobs; SYS2KTPI, the CMS SCF TP interface and TSO SCF TP interface invoked in TSO by a "CALL" command. Each of these three modules contains a CSECT that has an external reference to a unique user-exit name, as shown below. S2KDMV6 is the SAS Version 6 SCF interface invoked by executing PROC QUEST.

When you are including user-exit code with the linkage-editor, the proper entry point name must be assigned to the resulting load module, as shown in the following illustration. The interface routine GETS2K, which calls the user exit, is also shown in the illustration.

Illus. 3.7 Interface Modules and User Exits

Executable Load Module Name	User Exit Name	Calling Routine in Interface	Entry Point
SYS2KJOB	EXJOB	GETS2K	MSYS2K
MUPLINT	EXPLI	GETS2K	MSYSTM2
SYS2KTPI	EXTSO	SCFTP2I	SCFTP2I
S2KDMV6	EXTSO	S2KDMV6	S2KDMV6

GETS2K resides in both SYS2KJOB and MUPLINT. The illustration below shows sample linkage-editor JCL and control statements for including the appropriate user exit in SYS2KJOB. Illus. 3.9 on page 3-124 shows a simple Assembly language user exit for batch SCF. Copies of SYS2KJOB, MUPLINT, and SYS2KTPI must be saved before any relinking is attempted, so that the functional version still exists if problems occur.

Illus. 3.8 Link-Edit Including EXJOB in SYS2KJOB

```
//LINKSTEP    EXEC  PGM=IEWL,PARM='MAP,LIST'
//SYSPRINT    DD  SYSOUT=A
//SYSLMOD     DD  DSN=SYSTEM 2000 load-library,DISP=OLD
//SYSUT1      DD  UNIT=SYSDA,SPACE=(1024,(100,20))
//OBJECT      DD  DSN=USEROBJ,DISP=SHR
//SYSLIN      DD  *
   INCLUDE    OBJECT          *PREVIOUSLY COMPILED 'EXJOB'
   INCLUDE    SYSLMOD(SYS2KJOB)
   ENTRY      MSYS2K
   NAME       SYS2KJOB(R)
/*
```

Illus. 3.9 Sample Batch SCF User Exit

```
*          SAMPLE BATCH SELF-CONTAINED FACILITY INITIALIZATION EXIT
EXJOB      CSECT
           USING  *,15
           B      12, (,R15)
           DC     CL7'EXJOB'
           STM    14,12,12(13)
           LR     14,13
           CNOP   4,8
           BAL    13,*+76
           DROP   15
           USING  *,12
           DC     18F'0'
           LR     5,0                 ADDRESS URB
           LR     12,13
           ST     13,8(,14)
           ST     14,4(,13)
*CONVENTIONAL MVS LINKAGE COMPLETE
*ADDRESS OF URB IN REGISTER 5
           .
           .
           .
(USER-WRITTEN CODE TO BUILD A STEPNAME IN THE 8-BYTE FIELD 'USERSTEP')
           .
           .
           .
           MVC    88(8,5),USERSTEP    PUT IT IN THE URB
           L      13,4(,13)
           LM     2,12,28(13)
           L      14,12(,13)
           BR     14                  RETURN TO GETS2K
USERSTEP   DC     CL8'XXXXXXXX'
END        EXJOB
```

ANALYZING MULTI-USER EVENTS: THE DIAGNOSTIC LOG

In a Multi-User environment, many events happen simultaneously. Although event data are transparent to the end user, you can gather information about these events if you activate the Diagnostic Log. Each record on the Diagnostic Log contains a message about some event that happened during the Multi-User session.

Various users can access SYSTEM 2000 databases through SCF jobs and PLEX programs. Sometimes a job or program has to wait for I/O, a database, a scratch pad, a buffer, an overlay, or a tape. User jobs are suspended in a thread if there are conflicts among global or local holds on a database. Wait time, user suspension, thread suspension, and overlay swapping can cause performance degradation and inefficient use of resources.

During a Multi-User session, the Diagnostic Log captures event data about database use, thread use, and user job activity. These data can help when you are determining buffer sizes, scratch pad sizes, Update Log page sizes, pool sizes, and more efficient overlay configurations. For some PLEX programs, the Diagnostic Log can serve as a debugging tool for tracing the program logic. Reasons for using the Diagnostic Log usually depend on specific site situations.

Normally, the Diagnostic Log has only the log level UINIT specified. Turn additional log levels on when you want to study particular events that may be influencing the performance of Multi-User. Because of I/O processing overhead, take time beforehand to determine the specific types of event data and the maximum number of messages to be logged during the Multi-User session. Then, carefully set the values for the LOGLEVEL and LOGCOUNT execution parameters. They control the type and amount of data logged.

The Diagnostic Log is written on the S2KDIAG data set. It can be allocated to disk, tape, or a printer. Disk is usually preferred because it avoids dedicating a tape for the entire Multi-User session.

If the Diagnostic Log is allocated to disk or tape, you can use the LOGDUMP or DIAG2000 utilities to display and organize the information. You can also write your own programs to read data from the log and produce customized reports.

The next topic describes listings and reports you can produce from the Diagnostic Log data. Read this material before you activate the Diagnostic Log so you will know what specific kinds of data to capture and how the Diagnostic Log can help you analyze your Multi-User system performance.

Displaying the Diagnostic Log

This discussion shows ways of displaying Diagnostic Log data and suggests some guidelines for using and interpreting the data. Specify the LOGLEVEL and LOGCOUNT parameter values appropriately to capture the type(s) and amount of data needed. (See **Generating Diagnostic Log Data: Execution Parameters** on page 3-133 for details.)

After writing the data to the log, you can use

- the LOGDUMP utility to obtain sequential listings of the event data
- the DIAG2000 utility to obtain detailed and summary reports about specific types of events
- the Diagnostic Log tuning tools reports described in *Technical Report: S2-106 Multi-User Tuning Tools for SYSTEM 2000 Software, Release 11.6 under IBM OS and CMS*
- your own code to obtain customized reports.

Sequential listings Use the LOGDUMP utility to dump a disk or tape log to the printer. You can list all of the Diagnostic Log messages or a subset of them. For details on executing LOGDUMP, see **How to Use the LOGDUMP Utility** on page 3-136.

LOGDUMP produces an unformatted, sequential listing of the messages written on the Diagnostic Log. The listings contain all events or any of the following subsets in any combination:

- a specific time period
- one or more specific user jobs
- one or more userids
- one or more thread numbers
- specific messages, identified by message number.

These sequential listings can be helpful when you are analyzing events that occurred during specific time intervals in a Multi-User session. If you request a listing of events for a specific job, thread, or userid, you can perhaps isolate potential problem areas that may be causing poor performance or inefficient use of resources. Listings can also be used as a debugging tool for tracing the logic in a PLEX program. You can see when the database(s) were opened and closed and when each PLEX command was processed.

All Diagnostic Log messages are given in **Format of Diagnostic Log Records** on page 3-144.

Database Activity Detail Report Use DIAG2000 to produce the Database Activity Detail Report. For details about compiling, link-editing, and executing DIAG2000, see **How to Use the DIAG2000 Utility** on page 3-139. To gather the data required by this report, specify the following LOGLEVEL variables: LOPEN, POPEN, UINIT, and USPND.

The Database Activity Detail Report shows the following statistics for each database accessed during the Multi-User session:

- the number of physical opens for the database
- the number of logical opens for the database
- the number of suspensions for users trying to
  - lock the database (LOCK)
  - obtain a global hold on the database (HOLD)
  - obtain update status on the database (UPDATE)
  - obtain retrieval status on the database (RETR)
  - open the database (OPEN)
  - frame the database (FRAME)
  - obtain rollback status on the database (ROLBK)
  - obtain non-key update status on the database (NKUPD)
  - obtain a local hold on the database (HOLDL)
  - obtain local hold session status on the database (obtained when Multi-User first realizes a user will be acquiring local holds; used to prevent a deadlock between two users who are requesting a common local hold) (LHSES)
- the number of SCF suspensions for users trying to access a database on which another user has a hold if the SCF user would be dispatched in the last available thread (printed under the NLSES heading).

If the report shows an excessive number of logical opens, perhaps PLEX programs are switching databases too often. A logical open occurs every time a PLEX program accesses a different database and then returns to a database that is already open.

If the numbers in any other column of the report seem larger than usual, it indicates a Multi-User database contention problem. These numbers vary widely from installation to installation.

Here is a sample Database Activity Detail Report.

```
DATA BASE NAME                D A T A   B A S E   S U S P E N S I O N   A N A L Y S I S
  PHY-OPEN  LOG-OPEN  LOCKS  HOLDS  UPDAT  RETR  OPEN  FRAME  ROLBK  NLSES  NKUPD  HOLDL  LHSES
LIBRARY
        01       156     00     00     00    00    00     00     00     00     00     00     00
PUBLISHERS
        01       153     00     00     00    00    00     00     00     00     00     00     00
```

Database Activity Summary Report Use DIAG2000 to produce the Database Activity Summary Report. For details about compiling, link-editing, and executing DIAG2000, see **How to Use the DIAG2000 Utility** on page 3-139. To gather the data required by this report, specify the following LOGLEVEL variables: LOPEN, POPEN, TSPIO, and USPND.

The Database Activity Summary Report shows the following statistics for each database accessed during the Multi-User session:

- total number of physical opens for the database
- total number of logical opens for the database
- total number of user suspensions for any status on that database
- number of reads and writes for each file associated with the database.

The I/O numbers in this report are based on the number of times that message 211 occurred on the Diagnostic Log. This message indicates thread suspension for I/O. This I/O count will not necessarily match I/O counts from other sources, such as SMF records or JES I/O counts.

The Database Activity Summary Report shows you which databases are experiencing an abnormally high number of user suspensions. Also, by analyzing the number of database accesses, you can identify which databases have greater I/O counts than expected.

Here is a sample Database Activity Summary Report.

```
                    TOTAL       TOTAL       TOTAL
  DATA BASE NAME    PHYS-OPENS  LOGL-OPENS  SUSPENDS

LIBRARY                 01          156         00

LIBRARY7      01R   00W   LIBRARY1      01R       00W   LIBRARY3   01R   00W   LIBRARY2   02R   00W
LIBRARY5      01R   00W   LIBRARY6      01R       00W

PUBLISHERS              01          153         00

PUBLISH5      01R   00W   PUBLISH6      01R       00W   PUBLISH3   01R   00W

** WARNING ** THE I/O COUNTS PRINTED ABOVE ARE BASED ON THE S2KDIAG FILE AND DO NOT NECESSARILY MATCH OTHER SOURCES.
```

Thread Activity Detail Report Use DIAG2000 to produce the Thread Activity Detail Report. For details about compiling, link-editing, and executing DIAG2000, see **How to Use the DIAG2000 Utility** on page 3-139. To gather the data required by this report, specify the following LOGLEVEL variables: TSPIO, TSPND, TSTRT, and UINIT.

This report shows the number of times each thread was initialized and the number of times each thread had to wait for

- buffers
- databases
- overlays
- tape
- I/O
- scratch pads
- Update Log(s).

This report also shows the total for all threads for each category that had any waits.

Some of the numbers displayed should be close to zero, for example, the number of waits for databases, tapes, overlays, and scratch pads. If these numbers are quite large, the problem may be a fairly obvious one, such as a system that is too highly overlayed or scratch pads that are too small.

The number of waits for I/O should be the highest number, relatively speaking. A problem with excessive I/O will probably not show up here.

The number of waits for buffers and Update Logs varies widely from installation to installation, so it is difficult to suggest what they should be. Generally, these numbers should be as low as possible. However, in virtual storage systems there is a tradeoff between the number of buffers specified for Multi-User and system-wide paging considerations.

Here is a sample Thread Activity Detail Report.

THREAD NBR	INIT COUNT	B-WAIT COUNT	D-WAIT COUNT	O-WAIT COUNT	T-WAIT COUNT	IO-WAIT COUNT	P-WAIT COUNT	U-WAIT COUNT
1	36	00	00	01	00	10	00	00
2	79	00	00	07	00	00	00	00
3	207	00	00	09	00	00	00	00

TOTAL INITIALIZATIONS	322
TOTAL OVERLAY WAITS	17
TOTAL IO WAITS	10

Thread Activity Summary Report Use DIAG2000 to produce the Thread Activity Summary Report. For details about compiling, link-editing, and executing DIAG2000, see **How to Use the DIAG2000 Utility** on page 3-139. To gather the data required by this report, specify the following LOGLEVEL variables: TSPIO, TSPND,TSTRT, and UINIT.

The Thread Activity Summary Report gives the following statistics for each thread:

- total time thread was active
- percentage of total active time spent in the thread
- total wait time used by the thread
- percentage of active thread time spent in the wait state
- percentage of thread wait time spent waiting for
  - buffers
  - databases
  - overlays
  - tapes
  - I/Os
  - scratch pads
  - Update Logs.

Active means the thread was initialized by Multi-User for a user. Active time includes both CPU and wait time used by Multi-User while that user was in the thread. If a thread was active across midnight, the number for Total Active Time will not be accurate. All timing numbers are based on elapsed times written to the Diagnostic Log by Multi-User.

The Thread Activity Summary Report can help you identify unnecessary threads, that is, those that are active a very small percentage of the time. Under normal circumstances, wait time should form a small percentage (for example, 10% to 20%) of a thread's active time. Of this wait time, the greatest percentage should be I/O wait time. Numbers in the report not conforming to this pattern indicate a problem.

Here is a sample Thread Activity Summary Report.

THREAD	TOTAL	PERCENT	TOTAL	PERCENT	WAIT TIME ANALYSIS							
NBR	ACTIVE TIME	ACTIVE	WAIT TIME	WAIT	BUFFR	DB	OVLY	TAPE	I/O	PAD	U-LOG	
1	0:03:5260	15:0%	0:00:0045	0.0%	0.0%	0.0%	11.0%	0.0%	88.0%	0.0%	0.0%	
2	0:05:0750	20:0%	0:00:0332	1.0%	0.0%	0.0%	100.0%	0.0%	0.0%	0.0%	0.0%	
3	0:15:5413	63:0%	0:00:2995	3.0%	0.0%	0.0%	100.0%	0.0%	0.0%	0.0%	0.0%	

User Job Activity Report Use DIAG2000 to produce the User Job Activity Report. For details about compiling, link-editing, and executing DIAG2000, see **How to Use the DIAG2000 Utility** on page 3-139. To gather the data required by this report, specify the following LOGLEVEL variables: LOPEN, POPEN, TSPIO, TSPND, TSTRT, UINIT, and USPND.

The User Job Activity Report shows the following statistics for each SYSTEM 2000 session written to the Diagnostic Log:

- job name. (The same job name could appear more than once.)
- start and stop times, as written to the Diagnostic Log.
- number of physical database opens that occurred in the job.
- number of logical database opens that occurred in the job.
- number of thread initializations. (This is roughly equivalent to the number of segments issued by the job.)
  number of thread suspensions (except for I/O).
- number of user suspensions for database status.
- number of suspensions for I/O.
- number of Update Log suspensions.

The statistics in this report help in isolating specific jobs that are causing excessive contention. You can also use the report to identify a job that is causing a performance problem found while analyzing another DIAG2000 report.

Here is a sample User Job Activity Report.

JOBNAME	START TIME	STOP TIME	PHYSICAL OPENS	LOGICAL OPENS	THREAD INITS	THREAD SUSPENDS	USER SUSPENDS	IO SUSPENDS	UPD LOG SUSPENDS
USERSBE	10:01:0673	10:02:4312	00	01	06	02	00	00	00
$LCPLXMA	10:00:5595	10:14:4347	00	105	108	09	00	00	00
$LCPLXMA	10:00:4679	10:22:5293	02	203	206	06	00	10	00
USERSBE	10:22:5327	10:23:0371	00	00	01	00	00	00	00
USERSH8	10:25:0415	10:23:1018	00	00	01	00	00	00	00

Setting Up the Diagnostic Log File

Multi-User writes the Diagnostic Log to the S2KDIAG data set, which can be a tape, disk, or printer. This section discusses the advantages and disadvantages of the various device types and gives sample JCL statements for describing the S2KDIAG data set.

Allocating the S2KDIAG data set The S2KDIAG file contains variable-length records, with a maximum record length of (LRECL-36) bytes. Each record contains a message about a Multi-User event. We recommend a large block size for S2KDIAG to reduce disk and tape I/O.

message length Release 11.6 and later releases allow you to have larger LRECL and BLKSIZE values for the Diagnostic Log (S2KDIAG file). For compatibility, these releases honor the LRECL and BLKSIZE that you specify in the JCL. Otherwise, SYSTEM 2000 software establishes defaults.

The maxmsg field (also known as the length of the variable data) is contingent upon the LRECL and the record-prefix length, which is 32. (Previously, the maxmsg field was up to 94 characters.) Under Release 11.6 and later releases, it can be up to (32K - 4 - 32), depending on the LRECL value. The difference is most noticeable for message numbers 301 and 302.

The following chart shows the values supplied, which depend on the values you specify for LRECL and BLKSIZE:

User supplies	SYSTEM 2000 software supplies
LRECL and BLKSIZE in DCB	Maxmsg = LRECL - record_prefix
LRECL in DCB	BLKSIZE = LRECL + 4 Maxmsg = LRECL - record_prefix
BLKSIZE in DCB	LRECL = BLKSIZE - 4 Maxmsg = LRECL - record_prefix
Neither in DCB	LRECL = 4092 BLKSIZE = 4096 Maxmsg = 4060

tape If you anticipate an extremely large number of Diagnostic Log records, use tape for the S2KDIAG data set. However, this requires a dedicated tape drive throughout the Multi-User session. If the tape fills, additional tape mounts will be requested. The tape is never reinitialized.

disk If you allocate the S2KDIAG data set to disk, its size must be large enough to contain the maximum number of messages set by the LOGCOUNT execution parameter. If this limit is exceeded, the log is automatically reinitialized (closed and opened again). New messages are written on top of the old ones, destroying those already on the log. Before the log becomes full, use LOGDUMP or a user-written routine to dump the disk at appropriate intervals. **Note**: LOGDUMP reinitializes the disk file after dumping the output to the printer. This means that unless Multi-User is down, data might be lost when you want to produce other listings or reports from the data logged. (If the S2KDIAG data set is tape or the printer, the file is not reinitialized.)

Whenever the disk data set is reinitialized, the first record on the log is a message that shows the number of times reinitialization has occurred. The message LOG INIT.xxxx TIMES occurs immediately after the time stamp, starting with a blank in Column 15. The total length of this Diagnostic Log record is 35 characters.

printer If you use a printer for the S2KDIAG data set, the printer output is spooled. With a lot of data, you might fill the spooler queue. If the Diagnostic Log messages are being sent directly to a printer and the LOGLEVEL variable PDATE is specified, the date and time printed at the beginning of each message is meaningless, because they are in packed format. (For tape and disk, LOGDUMP unpacks the PDATE time and date.)

S2KDIAG DD statement The Diagnostic Log messages are written to the data set described by the S2KDIAG DD statement. If LOGLEVEL=NO, the S2KDIAG DD statement is not necessary. The S2KDIAG data set contains standard IBM variable length records. You may want a BLKSIZE of greater than 200 to improve efficiency.

The JCL describing the S2KDIAG data set must be set up at your site. Here are some sample DD statements.

```
disk      //S2KDIAG DD DSN=S2K.DIAG,DISP=(,CATLG),UNIT=3350,
(new)       SPACE=(200,(500,100))

disk      //S2KDIAG DD DSN=S2K.DIAG,DISP=SHR
(old)

tape      //S2KDIAG DD DSN=S2K.DIAG,DISP=(,CATLG),UNIT=TAPE

printer   //S2KDIAG DD SYSOUT=A
```

Generating Diagnostic Log Data: Execution Parameters

To activate the Diagnostic Log, use the LOGLEVEL and LOGCOUNT Multi-User execution parameters. Set these parameter values when you initialize Multi-User.

The LOGLEVEL variables determine which types of event data will be written to the Diagnostic Log. The LOGLEVEL variables can also be modified dynamically during a Multi-User session if the LOGLEVEL parameter is not set to NO.

The LOGCOUNT parameter sets the maximum number of records (event messages) to be written. The LOGCOUNT value cannot be dynamically modified after Multi-User has been initialized.

Setting the LOGLEVEL execution parameter Before you specify any LOGLEVEL variables, decide which events you want to log. If you are not familiar with the kinds of reports DIAG2000 produces, read **How to Use the DIAG2000 Utility** on page 3-139. If you write your own program, make sure you collect the appropriate data.

To activate the Diagnostic Log, specify one or more LOGLEVEL variables. Each variable causes messages for a different type of event to be written to the Diagnostic Log.

The format for the LOGLEVEL execution parameter is

LOGLEVEL=*variable*[/*variable*] . . .

where *variable* is one of the following keywords, indicating the type of data to be logged:

NO	no logging. If NO is specified, you cannot alter the LOGLEVEL parameter.
LOPEN	logical open or close of a database.
PDATE	date and time in packed format. Time only is the default if PDATE is not specified.
POPEN	physical open or close of a database.
SDATE	date and time in SAS TODSTAMP informat. SDATE is faster than the SVC-based time stamp.
TSPIO	thread suspended or dispatched I/O.
TSPND	thread suspended or dispatched (except for I/O).
TSTRT	thread start or stop.
UINIT	Multi-User initialization, USER start or stop. UINIT is the default if the LOGLEVEL execution parameter is not used.
USEGM	command segment start or stop; used for listings on the printer or with LOGDUMP; not used by DIAG2000..
USPND	user suspension or acquit.

When you are specifying LOGLEVEL variables, consider the following guidelines:

- Embedded blanks are not allowed.
- All LOGLEVEL variables must be specified on one execution parameter record. No continuation is allowed.
- If more than one LOGLEVEL statement is encountered, only the last LOGLEVEL statement is used.
- Each variable requests a specific type of information, and each type is independent of the others. UINIT is always logged unless LOGLEVEL equals NO (the default).
- Each message in the Diagnostic Log is always time-stamped. If you specify PDATE, the time and date are combined in packed format. SDATE writes the date and time in SAS TODSTAMP informat and runs faster than the SVC-based time stamp.

SDATE gives you an alternate method of storing the date/time field at displacement 5 into the Multi-User Diagnostic Log records. If you specify SDATE and the Diagnostic Log is disk or tape, a "Store Clock" instruction is issued. The results are stored in the rightmost eight bytes of the ten-byte date/time field, that is, bytes 5 through 14. (SDATE uses the fourteenth byte, which normally is a blank filler.) With any other device type, such as the printer, the "Store Clock" is placed in a work area and is converted to hh.mm.ss format.

SDATE offers two advantages:

- "Store Clock" runs much faster than the SVC-based time stamp.
- The disk and tape format of the SDATE date/time field is compatible with the SAS TODSTAMP informat, which is an 8-byte time-of-day stamp.

- Use SDATE if you want to read the Diagnostic Log with the SAS System.
- The SDATE option and the PDATE option are mutually exclusive. If you try to specify both of them in the LOGLEVEL execution parameter, WTO message S2K0111/00 is issued.

Note: The LOGDUMP and DIAG2000 utility programs do not recognize an SDATE time stamp. You must use the SAS System or your own program to read disk or tape Diagnostic Log records created with the SDATE format.

If you initialize Multi-User with LOGLEVEL set to any variables except NO, the console operator can dynamically modify and display the LOGLEVEL settings with the commands shown below.

```
F S2K,LOGLEVEL=variable [/variable] . . .
F S2K,DISPLAY LOGLEVEL
F S2K,D LOGLEVEL
```

The LOGLEVEL variables for the console operator are the same as those used in the LOGLEVEL execution parameter.

When modifying the variables, the operator must specify all variables that are to remain set. For example, if LOPEN, POPEN, and PDATE are set and you want to turn off PDATE, you must specify LOPEN and POPEN again with

```
F S2K,LOGLEVEL=LOPEN/POPEN
```

Setting the LOGCOUNT execution parameter Use the LOGCOUNT execution parameter to set the maximum number of messages that can be written to the Diagnostic Log, that is, the S2KDIAG data set.

The format for the LOGCOUNT execution parameter is:

```
LOGCOUNT=n
```

where *n* is a positive integer specifying the maximum number of messages that can be logged in units of 1000. The range is 1 - 32767. The default is 1, that is, a maximum of 1,000 messages.

If the S2KDIAG data set is allocated to disk and the LOGCOUNT limit is not large enough, the disk is reinitialized. New data will be written on top of the old data. If you want to do a complete analysis of an entire Multi-User session, you might want to use a tape for the Diagnostic Log to ensure that you have all the event data.

The LOGCOUNT parameter cannot be modified by the console operator.

How to Use the LOGDUMP Utility

The LOGDUMP utility produces a sequential, unformatted listing of the Diagnostic Log messages. You can display all messages or a selected subset of them, for example, activities for specific jobs, userids, threads, specific types of messages, time intervals, and so forth.

LOGDUMP reads the messages that were written on the disk or tape S2KDIAG data set and prints them to the SYSPRINT data set. The content of the LOGDUMP listings depends entirely upon what kinds of information you chose to write to the Diagnostic Log. The LOGLEVEL and LOGCOUNT execution parameters determine which events are logged and how many messages are written. (See **Generating Diagnostic Log Data: Execution Parameters** on page 3-133.) If you want to obtain summary or detail reports, use the DIAG2000 utility discussed in **How to Use the DIAG2000 Utility** on page 3-139.

If the S2KDIAG data set fills up, the disk is reinitialized and new messages are written on top of the old ones. You can run LOGDUMP during a Multi-User session.

LOGDUMP is a stand-alone Assembler language program. It reads the following two input files:

- the SYSIN data set containing LOGDUMP commands that specify which types of Diagnostic Log messages to display
- the disk or tape S2KDIAG data set containing the Diagnostic Log messages.

The output from LOGDUMP goes to the SYSPRINT data set.

Specifying LOGDUMP commands LOGDUMP reads commands from the SYSIN data set. With these commands, you can request a listing of the entire Diagnostic Log or a selected subset of its contents. To specify LOGDUMP commands, use either (or both) of the following methods:

- Place each command on a separate SYSIN record beginning in Column 1. Do not use a comma after the command.
- Place several commands on a SYSIN record beginning in Column 1, with a comma between commands, for example,

  command=(*parameter*)[,*command*=(*parameter*)]...

The commands and parameters are the same for either method.

Command	Parameter	Parameter Meaning
START	*yyddd,hh.mm.ss*	year,day,hour.minute.second
STOP	*yyddd,hh.mm.ss*	year,day,hour.minute.second
JOBNAME	*jjjjjjjj*[,*jjjjjjjj*]...	job name
USER	*uuuu*[,*uuuu*]...	userid
THREAD	*tt*[,*tt*]...	thread number
MESSAGE	*iiii*[,*iiii*]...	message ID

If you do not specify a command, it defaults to the following:

Command	Default
START	beginning of the log
STOP	end of the log
JOBNAME	all job names
USER	all userids
THREAD	all threads
MESSAGE	all messages

When you specify your LOGDUMP commands, consider these guidelines:

- You can give commands in any order.
- Embedded blanks are not allowed. If a blank is found, scanning for that record stops, and the program begins reading Column 1 of the next SYSIN record.
- If a LOGDUMP command appears more than once on the SYSIN data set, only the last parameter value for that command is used.
- The number of parameters that can be specified for a command is essentially unlimited. It depends only on the size of the region in which LOGDUMP is executed.
- To list the events for a specific time period across several days, enter zeros for *yyddd* in the START and STOP commands. However, you must set *hh.mm.ss* to the appropriate start and stop times.
- Syntax errors in commands are reported as they occur. The following message appears:

  ```
  * * *SYNTAX ERROR IN COMMAND * * *
  ```

  The letter C is printed below the command in error.

The following statements show sample JCL for executing LOGDUMP:

```
//*  ************************************************************
//*  *   EXECUTE LOGDUMP UTILITY                                *
//*  *                                                          *
//*  *   INDEX = SYSTEM 2000 LOAD LIBRARY HIGH-LEVEL INDEX      *
//*  *   RLSE  = SYSTEM 2000 RELEASE LEVEL                      *
//*  *   LOG   = DATA SET DEFINED BY THE MU S2KDIAG DD STATEMENT *
//*  *   DSP   = DISPOSITION FOR INPUT DATA SET                 *
//*  ************************************************************
//*
//LOGDUMP  PROC INDEX=S2K,RLSE=R120,LOG='USER.LOGDSN',DSP='SHR'
//DUMP     EXEC PGM=LOGDUMP
//STEPLIB  DD   DSN=&INDEX..&RLSE..LOAD,DISP=SHR
//S2KDIAG  DD   DSN=&LOG,DISP=(&DSP)
//SYSPRINT DD   SYSOUT=A
//SYSUDUMP DD   SYSOUT=A
//SYSIN    DD   DUMMY
//PEND
//LOGRUN   EXEC LOGDUMP
//DUMP.SYSIN DD  *
START=(00000,10.00.00)
STOP=(00000,12.00.00)
JOBNAME=($DMPS2K)
MESSAGE=(0200,0201,0211,0212)
```

Output from LOGDUMP LOGDUMP produces an unformatted listing of the Diagnostic Log messages selected by the LOGDUMP commands. The output is written to the SYSPRINT data set.

If you specified PDATE for the LOGLEVEL execution parameter, the date and time are packed within each message. LOGDUMP unpacks them. In the output, the date appears on a line by itself, and each log message begins with the time.

The first four characters of each record on the log are reserved for use by the operating system and are not printed. They contain the length of the record and two reserved bytes.

The Diagnostic Log event messages are given in **Format of Diagnostic Log Records** on page 3-144.

Warnings and error messages issued by LOGDUMP are given in *SYSTEM 2000 Messages and Codes, Version 12, First Edition*.

How to Use the DIAG2000 Utility

The Diagnostic Log captures events as they happen during a Multi-User session. You can use the DIAG2000 utility to organize this information into detailed and summary reports. These reports give you statistics for analyzing Multi-User resource contention, performance, and the interaction of thread, database, and job activities. You can select any or all of the following reports each time you execute DIAG2000:

- Database Activity Detail Report
- Database Activity Summary Report
- Thread Activity Detail Report
- Thread Activity Summary Report
- User Job Activity Report.

Each report is discussed in **Displaying the Diagnostic Log** on page 3-125. The discussions include the type of statistics in each report, sample reports, and some suggestions for analyzing the report results.

DIAG2000 reads whatever was written to the Diagnostic Log. The type and amount of data depend on the values you assigned to the LOGLEVEL and LOGCOUNT execution parameters for the Multi-User session. Plan ahead. Be sure that you set these parameters to gather adequate and appropriate data. Otherwise, the information in the reports may be inaccurate and insufficient when you try to analyze it. See **Generating Diagnostic Log Data: Execution Parameters** on page 3-133 for details about specifying the LOGLEVEL and LOGCOUNT execution parameters.

Each time you execute DIAG2000, the DIAG2000 commands are displayed along with any error messages or anomalies in the S2KDIAG data set. Also, DIAG2000 displays the count of the number of messages in the Diagnostic Log.

DIAG2000 is a stand-alone COBOL program. The source code is provided on the SYSTEM 2000 delivery tape. Sample JCL to compile, link-edit, and execute DIAG2000 is given in this section.

Specifying the input to DIAG2000 DIAG2000 requires the following input:

- the Diagnostic Log (S2KDIAG data set) on disk or tape
- an execution time parameter passed via the JCL
- a set of DIAG2000 commands given in 80-byte records for the INPUT DD statement.

S2KDIAG data set - DIAG2000 reads a disk or tape S2KDIAG data set containing whatever messages were written to the Diagnostic Log. Make sure you have assigned a large enough number to the LOGCOUNT parameter.

The table below provides a cross-reference between LOGLEVEL variables and DIAG2000 reports. Some variables are required to produce a particular report. For example, TSTRT is required to produce thread reports. If you do not specify the appropriate variables, the totals shown in the reports will probably equal zero, or the report will fail completely.

Report Type	LOGLEVEL Variables						
	LOPEN	POPEN	TSPIO	TSPND	TSTRT	UINIT	USPND
THREAD DETAIL			Z	Z	R	W	
THREAD SUMMARY			Z	Z	R	W	
DATA BASE DETAIL	Z	Z				W	Z
DATA BASE SUMMARY	Z	Z	Z			W	Z
JOB ACTIVITY	Z	Z	Z	Z	Z	W,I	Z

where

I means some totals may be incorrect if this variable is not specified

R means this variable is required to generate this report

W means a warning message is issued if this variable is not specified

Z means invalid zeros are reported if this variable is not specified.

Note: The USEGM variable is used only for listings, not by DIAG2000.

DIAG2000 execution parameter - The PARM='R120' execution parameter must be given in the JCL EXEC statement that invokes DIAG2000. This parameter is necessary because of changes in the format of Diagnostic Log messages that were introduced in Release 10.

DIAG2000 commands - These commands specify which reports you want to produce. DIAG2000 commands can be specified following the INPUT DD statement or in 80-byte records in the data set allocated in that statement.

Each DIAG2000 command must have a '1' in Column 1. Columns 2 through 5 specify a keyword (left-justified) that indicates the type of report(s) to be printed.

Possible values for the type field in Columns 2 through 5 are

Type	Meaning
THRD	requests a Thread Activity Report
DB	requests a Database Activity Report
JOB	requests a Job Activity Report
ALL	requests all reports

Column 6 designates either a detail report or a summary report if type is THRD or DB. This field has no meaning and is ignored if you specify JOB or ALL in the type field. Possible values for the detail/summary field in Column 6 are:

D for a detailed report

S for a summary report

Following are some sample DIAG2000 commands:

DIAG2000 Command	Report Requested
1THRDD	Thread Activity Detail Report
1DB S	Database Summary Report
1JOB	Job Activity Report
1ALL	all reports

Job Control Language DIAG2000 is provided as COBOL source code on the SYSTEM 2000 delivery tape. The following sample JCL illustrates compiling the DIAG2000 program:

```
//jobname JOB ...
//COB    EXEC       PGM=IKFCBL00,REGION=120K,
//                  PARM='BUF=10K,NONUM,LOAD,APOST'
//SYSIN         DD  DSN=S2K.R120.TEST(DIAG2000),DISP=SHR
//SYSLIN        DD  DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
//                  SPACE=(80,(500,100))
//SYSPRINT      DD  SYSOUT=A
//SYSUT1        DD  UNIT=SYSDA,SPACE=(460,(700,100))
//SYSUT2        DD  UNIT=SYSDA,SPACE=(460,(700,100))
//SYSUT3        DD  UNIT=SYSDA,SPACE=(460,(700,100))
//SYSUT4        DD  UNIT=SYSDA,SPACE=(460,(700,100))
//LKED   EXEC       PGM=IEWL,PARM='LIST,XREF,LET',
//                  REGION=96K,COND=(5,LT,COB)
//SYSLIN        DD  DSNAME=&LOADSET,DISP=(OLD,DELETE)
//              DD  *
 NAME DIAG2000(R)
//SYSLMOD       DD  DSN=user.load.library,DISP=SHR
//SYSLIB        DD  DSN=SYS1.COBLIB,DISP=SHR
//SYSUT1        DD  UNIT=SYSDA,SPACE=(1024,(50,20))
//SYSPRINT      DD  SYSOUT=A
//
```

After linking DIAG2000 and creating a disk or tape S2KDIAG data set, use the following JCL to run DIAG2000:

```
//jobname JOB ...
//stepname EXEC PGM=DIAG2000,PARM='R120'
//STEPLIB       DD  DSN=user.load.library,DISP=SHR
//DIAG          DD  DSN=S2KDIAG.dsn,DISP=SHR
//ERRORS        DD  SYSOUT=A
//INPUT         DD  *

   ...(DIAG2000 commands)

//PRINT         DD  SYSOUT=A
//SPOOL         DD  UNIT=SYSDA,DCB=(LRECL=52,BLKSIZE=5200),
//                  SPACE=(TRK,(2,1))
//SYSOUT        DD  SYSOUT=A
//SYSPRINT      DD  SYSOUT=A
//SYSUDUMP      DD  SYSOUT=A
//
```

Consider the following guidelines when specifying your JCL to execute DIAG2000:

- Use the SYSOUT and SYSPRINT DD statements only to capture COBOL errors or explanatory messages. They are not necessary for executing DIAG2000.
- The DIAG DD statement defines the input Multi-User Diagnostic Log file, that is, the S2KDIAG data set. See **Setting Up the Diagnostic Log File** on page 3-132.
- The INPUT DD statement must define a file for the DIAG2000 commands, either an in-line file ('DD *') or a file with LRECL=80 and RECFM=F or FB.
- The PRINT DD statement defines the major output file, which contains a listing of DIAG2000 commands, the reports, and a count of the S2KDIAG input messages. This file has fixed-length records with a record length of 133 bytes.
- The ERRORS DD statement defines a print file for any error messages. This file contains fixed-length records having a record length of 133 bytes.
- The SPOOL DD statement defines a scratch file used by DIAG2000. The file must have a record length of 52 and a block size that is a multiple of 52. To calculate the space necessary for the SPOOL file, assign one record for each user job on the Diagnostic Log. Usually the space requirements for the SPOOL file are one-fourth the space of the S2DIAG file.

Output from DIAG2000 If the PDATE variable is set in the LOGLEVEL execution parameter, the time and date are written in packed format for each message. DIAG2000 unpacks them, discards the date, and displays the time.

The reports produced by DIAG2000 are discussed in **Displaying the Diagnostic Log** on page 3-125. Also, each time you execute DIAG2000, the output includes a listing of your DIAG2000 commands, error messages (if any), and a count of the number of Diagnostic Log messages. Here is a sample Diagnostic Log record count showing the total number of messages (records) written to the log and the totals for messages grouped by message number, for example, messages 100 to 199.

```
TOTAL INPUT RECORDS. . . . . .      1,042
RECORDS IN RANGE 01-99 . . . .         20
RECORDS IN RANGE 100-199 . . .        318
RECORDS IN RANGE 200-299 . . .        704
RECORDS IN RANGE 300-399 . . .          0
RECORDS IN RANGE 400-499 . . .          0
```

Messages and codes DIAG2000 issues the following types of error messages:

- non-fatal warnings about a condition that probably should be investigated.
- messages about a processing error that is not serious enough to cause termination of DIAG2000. Usually these messages mean there are discrepancies in the S2KDIAG file input.
- messages for fatal errors, that is, errors that stop the execution of DIAG2000.

DIAG2000 error messages are listed in *SYSTEM 2000 Messages and Codes, Version 12, First Edition*.

Format of Diagnostic Log Records

The Diagnostic Log records are written to the DDname S2KDIAG. The records are variable length with a maximum record length of 126 characters. Each record contains a message regarding some type of event. The format of the messages is shown below. All messages are standard IBM variable length records containing 32 characters of information in a fixed format and up to (LRECL - 36) characters of variable data.

If the S2KDIAG data set has been reinitialized, the message LOG INIT. *xxxx* times is written in Columns 15 through 35 of the first log record. This record is only 35 characters long. The record format for all types of Diagnostic Log messages is shown below.

Column	Total Number of Characters	Contents	Format
1-2	2	Record length	Binary
3-4	2	Reserved	Reserved
5-8	4	Date	00YYDDDF, packed*
9-13	5	Time	0HHMMSSSSF, packed*
14	1	Filler	Blank*
15-22	8	Jobname	Character
23-26	4	Userid	Character
27-28	2	Thread ID	Character
29-32	4	Message ID	Character
33 to LRECL-4	(LRECL-36)	Variable data, if any	Character

* If PDATE is specified in the LOGLEVEL execution parameter, the date and time are combined in packed format. If the LOGLEVEL variable PDATE is not specified, only the time is written in Columns 5 through 14 in the format HH.MM.SSSS (hundredths). For SDATE, bytes 5 through 14 contain the date and time in SAS TODSTAMP informat.

Illus. 3.10 Messages For Each LOGLEVEL Variable

LOGLEVEL Parameter	Log Message Number	Variable Data (Length, if any)		Message Description
UINIT	0001	*LOGLEVEL settings**	(4 char)	Multi-User initialization started
	0002			Accounting initialized
(logged	0003			Multi-User initialization complete
when any	0004			Sign on non-TP user
level	0005			Sign on TP user
except NO	0006			Multi-User terminated during initialization
is specified	0007			TP user request canceled
or when	0008			TP user table full; request canceled
requested	0009			No memory for user control blocks
by itself)	0010			SCF session not allowed; not enough threads to start SCF session
	0011			USER died for POST
	0012			Error physical open of database
	0013			Terminate non-TP user
	0014			Terminate SCF TP user
	0015			Terminate Multi-User
	0016	*db name*	(12 char)	Database close on termination
	0017	*size* BYTES		Amount of memory GETMAIN gets for copy area
	0020-0030**			ESTAE work area messages
	0031	*LOGLEVEL settings**	(4 char)	Modified LOGLEVEL settings
POPEN	0100	*db name*	(12 char)	Physical database open
	0101	*db name*	(12 char)	Physical database close
	0213	*db name*	(12 char)	Reset Rollback Log
LOPEN	0102	*db name*	(12 char)	Logical database open
	0103	*db name*	(12 char)	Logical database close
TSTRT	0200	*xxx yyy****		Initialize thread
	0201			Terminate thread
TSPIO	0211	*ddnameInnnn* *ddnameOnnnn* followed by *bufage clock bufaddr*		Physical I/O thread Suspend thread I/O *ddname* identifies the file being accessed I,O indicate write, read *nnnn* is the page number, in hex, relative to zero (BDAM files only) *bufage* is the page age in hex, relative to X'-1'; (X'-1' indicates the oldest page and 0 indicates a freed page)

continued on next page

---

* The display format for LOGLEVEL settings is shown in **Format of LOGLEVEL settings display** on page 3-147.

** The ESTAE work area is formatted when an abend occurs. For details, see **Formatted ESTAE work area messages** on page 3-148.

*** *xxx* is the user priority and *yyy* is the number of dispatches since the user was added to the queue; that is, average wait time. The user priority may be an adjusted priority.

Illus. 3.10 *continued*

LOGLEVEL Parameter	Log Message Number	Variable Data (Length, if any)	Message Description
			clock is the buffer manager clock in hex, relative toX'-1'; (X'-1' indicates the earliest time) *bufaddr* is the page address in hex
	0212	*xxx yyy*	Dispatch thread I/O
USPND	0303	LOCK *db name* HOLD *db name* UPDATE *db name* RETR *db name* OPEN *db name* FRAME *db name* ROLBK *db name* NLSES *db name* NKUPD *db name* HOLDL *db name* LHSES *db name* RESETR*db name*	Suspend user on database contention The user was suspended while trying to obtain the specified status.
	0304	*user id*	DISPATCH user. The database status that caused the user to be suspended has been released.
TSPND	0202	*pool mask* (2 char)	Wait for buffer — Suspend thread
	0203	Database request (6 char)	Wait for database — until the resource is
	0204	(4 char)	Wait for overlay — released
	0205		Wait for tape
	0210	*ddname page #*	Wait for I/O
	0206	*xxx yyy**	Buffer or I/O — Dispatch thread
	0207	*xxx yyy**	Database — that was waiting for
	0208	*xxx yyy**	Overlay — the specified
	0209	*xxx yyy**	Tape — resource
	0214	*pad number*	User suspended in thread because there are no available allocation units.
	0215	*pad number xxx yyy*	Allocation units have been released by other users; the suspended user will now be dispatched to continue processing.
	0216		User suspended in thread attempting to lock an Update Log that has been loaded by another user.
	0217	*xxx yyy**	A locked Update Log has been released. The suspended user will now be dispatched.
USEGM	0300	*opcode dbn SSR*	PLEX segment start
	0301	*SCF command stream*	SCF segment start (TP)
	0302	*SCF command stream*	Read command file (non-TP)
	0305	*DYNAM where-clause text*	For GET1 or LOCATE DYNAMICALLY (follows message 300 for opcode 07 and 12)
	0399	*opcode dbn SSR*	Secondary database names on FRAME command

* *xxx* is the user priority and *yyy* is the number of dispatches since the user was added to the queue; that is, average wait time. The user priority may be an adjusted priority.

Format of LOGLEVEL settings display The Multi-User 0001 initialization record (initial LOGLEVEL settings) and message 0031 (modified LOGLEVEL settings) have a 4-byte variable data field starting in column 29. The first four bits are unused. The fifth and sixth bits indicate the type of date format; these bit settings are useful when displaying the date and time. The remaining bits indicate other LOGLEVEL settings as shown below.

```
NO          ....  ....  ....  ....
SDATE       ....  1...  ....  ....
PDATE       ....  .1..  ....  ....
DFLOW       ....  ..1.  ....  ....
LOPEN       ....  ...1  ....  ....
TSPIO       ....  ....  1...  ....
MFLOW       ....  ....  .1..  ....
TSPND       ....  ....  ..1.  ....
USPND       ....  ....  ...1  ....
USEGM       ....  ....  ....  1...
TSTRT       ....  ....  ....  .1..
POPEN       ....  ....  ....  ..1.
UINIT       ....  ....  ....  ...1
```

The date/time format depends on the LOGLEVEL setting and is initially set by S2KPARMS.

The length of the Multi-User initialization record (message 0001) indicates whether the 4-byte field is present. If the length is less than 30, message 0001 was written with Release 11.5, and the 4-byte LOGLEVEL field is not present. If the record length for message 0001 is 30 or longer, the record was written by Release 11.6 or a later release and contains the variable data.

Formatted ESTAE work area messages If a Multi-User session terminates abnormally, a SYSTEM 2000 Error Code or an IBM user abend is issued. SYSTEM 2000 software formats the ESTAE work area when an abend occurs. The formatted work area appears on the Diagnostic Log with message numbers 20 through 30. Messages 20 and 22 are:

```
20  -881-  SYSTEM 2000 ABEND PROCESSING INVOKED -
22  -822-  STAE WORK AREA FOLLOWS  -
```

The next lines contain the completion code, Problem Program PSW, program name, entry point address, abend PSW, and the contents of R0 through R15 at the entry to the abend. Each item is labeled.

Here is an example of a formatted ESTAE work area:

```
20  -881- SYSTEM 2000 ABEND PROCESSING INVOKED  -
22  -882- STAE WORK AREA FOLLOWS  -
23  COMP CODE 80000800 type = m
24  P/P PSW   FF65000D  06308028          PGM NAME SYS2K
25  ABEND PSW FF65000D  06308028          EP ADDR 002F8708
26  REGISTERS AT ENTRY TO ABEND
27     R0       R1       R2       R3       R4       R5       R6       R7
28  00000000 00000021 0031BE24 00327C34 00316570 003167D0 00313230 0030B5E7
29     R8       R9       R10      R11      R12      R13      R14      R15
30  00314668 00308A4F 00314010 002F8BAB 00307A50 00316520 56307FF8 00000000
```

For problem program state abends, the variables are as follows:

type equals USER.

m equals abend code.

nnn equals 820 (SYSTEM 2000 Error Code).

For system abends, the variables are as follows:

type equals SYSTEM.

m equals abend code.

nnn equals 826 (SYSTEM 2000 Error Code).

For both cases above, a complete snap dump is produced on the S2KSNAP file.

If the ESTAE work area is unavailable, Multi-User issues messages 20 and 21.

```
20  -881- SYSTEM 2000 ABEND PROCESSING INVOKED  -
21  -883-  STAE WORK AREA NOT AVAILABLE FOR FORMATTING -
```

Chapter 4

Specifying Files and Buffers

A SYSTEM 2000 database consists of files called database files. This chapter describes those files, other files used by SYSTEM 2000 software for processing, and the I/O buffers needed to handle the files.

The information is divided into the following sections:

- **SYSTEM 2000 Files: Overview** on page 4-3 presents a list of files used by SYSTEM 2000 software and outlines how a SYSTEM 2000 database is built.
- **Database File Sizes** on page 4-6 describes database files and their page sizes.
- **Allocating Database Files** on page 4-15 describes how to allocate database files.
- **Determining I/O Buffer Requirements** on page 4-23 presents the SYSTEM 2000 I/O buffer and describes how to determine its space requirements.
- **Allocating Work Files** on page 4-37 describes how to allocate space for work files.
- **Database Definition Block Sizes** on page 4-54 describes how to specify memory requirements for database definitions.
- **Gathering Timing Statistics: QAEXIT and QASTAT** on page 4-57 describes how to use the QAEXIT command and QASTAT call to gather CPU timing for commands and I/O counts for individual files.
- **Prototyping Databases** on page 4-62 describes how prototype databases can be used for testing file allocations.

SYSTEM 2000 FILES: OVERVIEW

Files Used by SYSTEM 2000 Software

SYSTEM 2000 software uses a number of files for processing. This list categorizes these files by usage, indicates whether they are required, and gives their DDnames.

- An optional file containing execution parameter specifications. The DDname is **S2KPARMS**. See **Setting SYSTEM 2000 Execution Parameters** on page 5-9.
- An optional file to hold a snap dump, if one is produced. The DDname is **S2KSNAP**. See **STAE Parameter: Enabling Error Trapping** on page 5-14.
- Several sets of work files described in **Allocating Work Files** on page 4-37.
  - Scratch files are work files used by SYSTEM 2000 software to store the results of intermediate processes.

    The Multi-User environment requires scratch pads to hold the scratch files. You can use up to 16 scratch pads with the DDnames **S2KPAD00** to **S2KPAD15**. For a single-user job, scratch pads must also be used to hold scratch files, except for the situation discussed in **Allocating Scratch Files Individually** on page 4-52.
  - Sort files are work files that the software uses when sorting scratch files. The Multi-User environment requires one set of sort files for each thread. The DDnames are **SFn1** to **SFn6**, where n = 0 to 62. For a single-user job, you need one set of sort files. The DDnames are **SF01** to **SF06**.

- Special files for the Multi-User environment:

  - A file used to hold information about all current TP run-units. The DDname is **S2KUSERS**. See **S2KUSERS File** on page 3-30 for details about this file.

  - A file where SYSTEM 2000 software writes the Diagnostic Log. The DDname is **S2KDIAG**. See **Analyzing Multi-User Events: the Diagnostic Log** on page 3-125 for details about the Diagnostic Log.

  - Two files where SYSTEM 2000 software writes the disk Accounting Log. The DDnames are **S2KMANX** and **S2KMANY**. See **Using the Accounting Log** on page 3-106 for details about the Accounting Log.

- Four user files. Two of these user files contain user input to the software, and two contain output as described below.

  - An optional Data File containing data to be loaded into a database. This file is used only in SCF jobs.

  - A Command File containing commands that you issue. The DDname is **S2KCOMD**. This file is used only in SCF jobs.

  - A Message File containing SYSTEM 2000 messages. The DDname is **S2KMSG**. This file is used in SCF and PLEX jobs.

  - A Report File containing output. Unless otherwise specified, SYSTEM 2000 software sends Report File and Message File output to the same data set. This file is used only in SCF jobs.

    For the S2KTPI interface, an intermediate output file (**S2KOUTP**) can be allocated to hold output destined for the terminal. For details about the file S2KOUTP, see **SYS2KTPI interface** on page 3-36.

  For details about user files, see *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition.*

- Six required database files (Files 1 through 6) and two optional files (Files 7 and 8) for each active database. For details about their file names, sizes, and allocations, see the following sections in this manual. Also, become familiar with the ALLOC command discussed in *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

- Two files are used for each archival database:

  - A file to save the database, called the Savefile. The DDname is the first seven characters of the database name plus a suffix of S. The Savefile is a QSAM file.

  - A file to keep Update Log recordings for a database, called the Keepfile. The DDname is the first seven characters of the database name plus a suffix of K.

  For details about saving databases and recording updates on the Keepfile, see related topics in *SYSTEM 2000 CONTROL Language, Version 12, First Edition.* For details about the buffers needed for saving a database, see **Using Pools** on page 4-29.

Building Active Databases

SYSTEM 2000 databases are either active or archival. An active database is online and immediately accessible. An archival database is offline, accessible only through the CONTROL language or a PLEX job.

SYSTEM 2000 software keeps active databases in tables stored in database files. These tables and their corresponding files are described in Illus. 4.1 on page 4-7. To help you understand how these files are used, here is an overview of the process for building a database. The process includes the following steps:

1. **Allocating the data sets for the database** - You can let the software dynamically allocate the data sets (using defaults) or you can issue the ALLOC command. Alternatively, you can physically allocate the files during the same job step in which SYSTEM 2000 software is executed or during an independently executed job step. If the Update Log (File 7) and the Rollback Log (File 8) are to be used, these files can also be dynamically allocated by the software, or you can provide DD statements to allocate them.

2. **Creating the database** - The first time the database is accessed, SYSTEM 2000 software automatically formats the allocated data sets to contain all binary zeros. If the software finds that the data sets have already been formatted (that is, by a user routine), this step is bypassed.

3. **Naming the database** - The user assigns the database name and its master password when the database is created. The database name must be associated with the DDname of the data sets. When the NEW DATA BASE IS command is issued, SYSTEM 2000 software opens the database files, starting with File 6 and ending with File 1.

4. **Defining the database** - After the database is named, it must be defined using the DEFINE language.

5. **Loading the database** - The last step involves loading the data, using either SCF or PLEX. When the DATA BASE NAME IS command is issued in daily use, SYSTEM 2000 software opens the database files starting with File 1 and ending with File 6, and, optionally, File 7 and File 8.

DATABASE FILE SIZES

This section discusses how to determine the page size and the number of pages for each database file.

Database Files: Overview on page 4-6 describes, in general terms, each of the eight database files.

Selecting Page Sizes for Database Files on page 4-8 describes general considerations for selecting page sizes for database files.

File 1 Size on page 4-9 through **File 8 Size** on page 4-14 describe how to determine the page size and the number of pages for each database file.

For database files, the *page size* and *block size* are identical. You specify block size in the BLKSIZE subparameter in the DD statement for each database file. (See **Specifying DD Statements for Database Files** on page 4-19.)

Database Files: Overview

SYSTEM 2000 databases are either active or archival. An active database is online and immediately accessible. An archival database is offline, accessible only by the CONTROL processor.

An active database consists of seven database tables: the Master Record, the Definition Table, the Distinct Values Table, the Extended Field Table, the Multiple Occurrence Table, the Hierarchical Table, and the Data Table.

The Master Record and the Definition Table both reside in File 1. Each of the other tables resides in a separate file, named File 2 through File 6, respectively.

A file for the Update Log (File 7) is also required if update logging is to be enabled. An additional file for the Rollback Log (File 8) is needed if the Coordinated Recovery feature is to be used for the database.

Illus. 4.1 on page 4-7 gives an overview of the database tables.

Illus. 4.1 Database Tables: Contents and Relationships

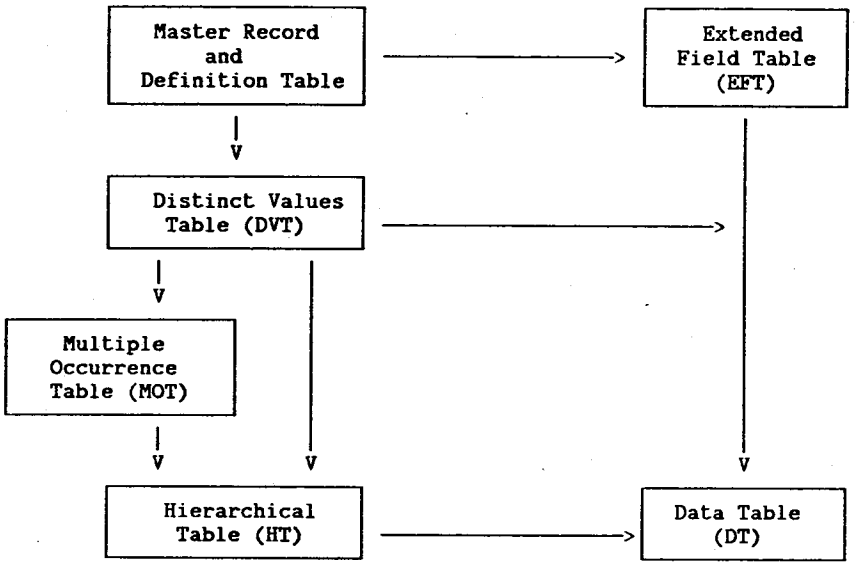


Table	File	Contents
Master Record	1	Database name, definition version number, cycle number, passwords, authorities, and backup and recovery information.
Definition Table	1	Component numbers, component names, and descriptions.
Distinct Values Table	2	The primary part of the index, containing the distinct values for each key item, with multi-level indexing.
Extended Field Table	3	Long values from the Definition Table, Distinct Values Table, and Data Table.
Multiple Occurrence Table	4	A secondary part of the index, containing an entry for each multiple pointer to key values.
Hierarchical Table	5	Internal pointers that relate each data record to its parent, next sibling, and first descendant, thus maintaining a hierarchical structure for the data.
Data Table	6	Representations of data values grouped by data record.
Update Log	7	Updates in processed form. This is an optional file used only to recover a database with Coordinated Recovery.
Rollback Log	8	Before-image records of modified database pages. This is an optional file used only to recover a database with Coordinated Recovery.

An *archival* database is a complete copy of the database stored in a file called the Savefile. The Update Log for the database is kept in a file called the Keepfile.

The Savefile holds a copy of Files 1 through 6 as of a given point in time. For Multi-User or single-user jobs, the Savefile is identified with a DDname of the first seven characters of the database name plus a suffix of S; the Savefile is a QSAM file. The Savefile can be allocated to disk or tape. SYSTEM 2000 software can dynamically allocate the Savefile.

The Keepfile is a permanent file for update logging. The Keepfile contains a processed form of successful updates to the database. Each record is identified by the database cycle number. Users issue the KEEP command to transfer the update records from the Update Log to the Keepfile.

For more information about saving and restoring databases see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

Selecting Page Sizes for Database Files

For a given disk device and environment, there are usually several choices of page size. See Illus. 4.3 on page 4-26 for information about these choices. For example, page size 6216 is an efficient, portable page size for 3330, 3350, 3380, and 3390 disk drives.

When determining page sizes, just as when designing a schema, consider the data to be stored, the transactions to be done, and the qualification criteria to be used. Also, consider carefully the application. Generally, I/O is minimized by assigning a large page size for any table expected to have many entries. On the other hand, large pages can waste space if the table has few entries. Since the size of a table depends on the actual values stored in the database, it is important to consider the distribution of key values before assigning index page sizes.

Some databases have many distinct values as well as many identical values. Updates and retrievals for such databases are processed most efficiently if Files 1, 2, and 4 all have large page sizes.

You can reduce I/Os for a specific database file by assigning a unique page size for that file and specifying a pool with the "exact fit only" attribute (that is, DE usage). DE usage prevents other files from using those buffers. This technique may be useful for database File 2 if you plan to use key processing.

For details about specifying page sizes (BLKSIZE value) for database files, see **Specifying DD Statements for Database Files** on page 4-19, and also read about the ALLOC command that allows you to specify block sizes within an SCF session (see *SYSTEM 2000 CONTROL Language, Version 12, First Edition*).

You can change the page size of a database file by executing the CNVRT12 program, which allows you to reblock a Version 12 database (see **The CNVRT12 Conversion Program** on page B-2). Or, you could unload the data, free the database, allocate a new one with the desired page size, redefine the database, and reload the data.

File 1 Size

As shown in Illus. 4.1 on page 4-7, File 1 contains the Master Record and the Definition Table. You can define a maximum of 10,000 components.

The Master Record contains: the database name, definition version number, cycle number, passwords and authorities, backup and recovery information.

The Definition Table has the following characteristics:

- It is a fixed-block type of table consisting of two parts: a component attribute part and component name part.
- Each entry contains the characteristics of a component.
- Entries representing key items point to the start of the index for that item.
- Entries representing schema records point to the first reusable Hierarchical Table entry for that schema record.

We recommend no special page size for File 1. The number of pages needed for File 1 depends on the maximum number of components allowed for the database.

File 2 Size

File 2 contains only the Distinct Values Table (DVT). The DVT and the Multiple Occurrence Table form the Index. SYSTEM 2000 software uses the Index to find data records containing values for key items.

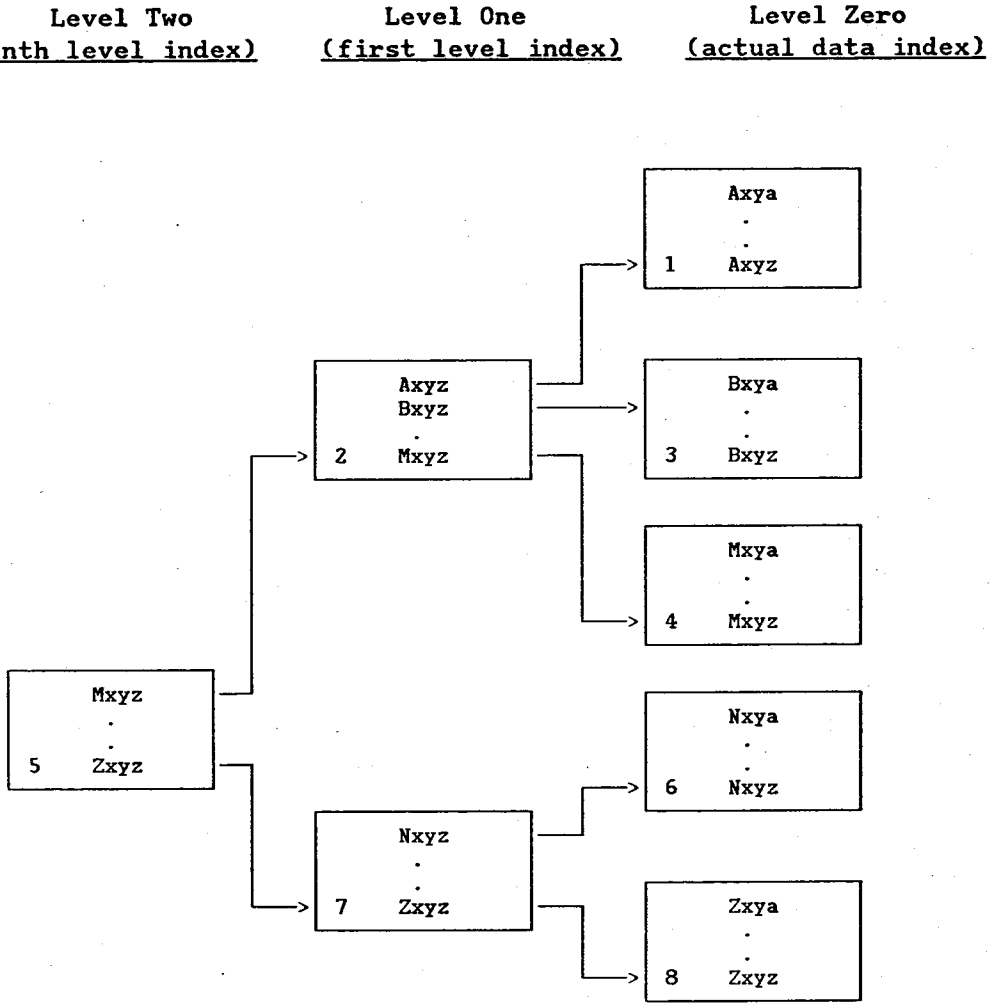
The DVT has the following structure:

- There is one logical table for each valued key item.
- Each logical table consists of one or more pages with multi-level indexing.

  Illus. 4.2 on page 4-10 shows multi-level indexing. In the example shown there, the system would find the entry for 'Bxya' by searching pages 5, 2, and 3.
- One entry exists for each distinct value of each key item.
- Entries on a page are maintained in collating sequence.
- Each entry has a pointer (to the Multiple Occurrence Table or to the Hierarchical Table) and its associated data.

Reorganization changes the order of the pages but does not change contents, padding, or packing, except for higher level index pages.

Illus. 4.2 DVT Multi-Level Index Structure



Page size for File 2 To determine the page size for File 2, examine the values of key items in the database.

The optimal page size is one that reduces the number of I/Os to File 2 without wasting space. When choosing a page size, consider

- the number of key items having many distinct values and how often these are accessed
- the number of key items having few distinct values and how often these are accessed

- that one I/O at each level in the DVT is required to access a given value
- that the number of pages needed to keep values for one item determines the number of levels needed in the higher level index.

If one item has many distinct values, the DVT needs many pages to store its values. If the page size is small, there are fewer values to a page, requiring more I/O to access a value and more higher level pages. On the other hand, a large page size allows more values to fit on a page, requires fewer higher level pages, and reduces I/O.

The I/O requirements can be estimated quite easily. Each access to a data record with a key item requires *n* I/Os on the DVT, where *n* is the number of levels in the index for that item. Actually, fewer I/Os may be needed if one or more pages are already in memory. Typically, there is one additional access to the Hierarchical Table and one to the Data Table. More than one I/O may be required in the Data Table, for a total of $(n+2)$ I/Os. If all the values for an item are distinct, there is no access needed to the Multiple Occurrence Table.

When you specify a range of many distinct values, the software first uses the higher level pages to establish the boundaries and then accesses the zero-level DVT pages.

I/O can be minimized by specifying a page size that is one larger than the default for the device.

Number of pages for File 2 The number of pages needed for File 2 depends on the number and size of valued key items, the number of distinct values for those items, and the option to enable values padding. The maximum size of File 2 is 536,870,912 bytes.

File 3 Size

The Extended Field Table (EFT) is the only database table residing on File 3. The file contains the following:

- component names exceeding the entry size in the Definition Table
- definitions for strings and functions
- the rightmost part of CHARACTER or TEXT values in the DVT and the values in the Data Table that exceed their picture sizes.

The space in the table is not reused except by a reload operation.

Page size for File 3 Values do not split across EFT pages; therefore, only one I/O is ever required for an EFT access. You may want to minimize (or completely eliminate) long values, thereby limiting the contents of the EFT to definition information. In that case, the maximum page size can be specified for the EFT, which would need very few pages. Also, a buffer pool configuration could be used to hold all EFT pages in memory (see **Using Pools** on page 4-29). Eliminating data overflow reduces EFT I/O, but it can increase the size of the Data Table and DVT records which, in turn, may increase I/O for those two tables.

Number of pages for File 3 The number of pages needed for File 3 depends on the number and length of CHARACTER and TEXT values that overflow. It also depends on the length of string and function definitions and on the number and length of item and schema record names. The maximum size of File 3 is 268,435,455 characters.

File 4 Size

The Multiple Occurrence Table (MOT) is the only database table residing on File 4. The DVT and the MOT form the Index.

The MOT has the following structure:

- One chain exists for each value of a key item occurring more than once.
- Each chain consists of one or more blocks containing pointers to the Hierarchical Table.
- Blocks of a chain need not be physically contiguous.
- Entry into a given chain is from a pointer in the DVT.
- You can request space (padding) for future use in a block, using the DEFINE language.

Reorganization rebuilds the blocks and makes them as large as the page size for File 4.

Page size for File 4 Large pages for File 4 offer more efficient processing of qualification criteria that involve many records containing identical values for an item. For example, the where-clause

```
. . . WHERE GENDER EQ MALE
```

selects about half of an average census database. I/O is minimized by having the entire chain of 3-byte pointers for a given value on the same MOT page. A large page size for File 4, therefore, minimizes I/O when many records have the same value for a key item.

You can quickly evaluate File 4 page size by comparing the allocated page size with

(estimated number of pointers per distinct value) x 3

If the product is larger than the page size allocated, a larger page would reduce I/O. Do this comparison for the key items that are accessed most frequently.

For example, consider a key item with several distinct values, each of which occurs an average of 4,000 times. File 4 would need about 12,000 bytes to store the pointer chain for each distinct value. This suggests a page size of 12992 for File 4. If each distinct value occurred 8,000 times or more, a larger page would be desirable. However, page size cannot exceed track size. The use of periodic REORGANIZE commands keeps updated pointer blocks for a given value contiguous, which also reduces I/O.

Number of pages for File 4 The size of File 4 is directly proportional both to the number of values occurring more than once for key items and to the number of times they occur. The maximum number of multiple occurrence pointers is 357,195,375.

File 5 Size

The Hierarchical Table (HT) is the only database table residing on File 5. The HT maintains the logical structure of the database. The table contains one entry for each data record in the database. The entries are fixed length (18 bytes).

The HT has the following structure:

- Each entry has
  - a pointer to the data record location in the Data Table
  - a pointer to the HT entry of the next sibling data record
  - a pointer to the HT entry of its parent
  - a pointer to the HT entry of its first child.
- Entries are reused by like data records on a last-in/first-out (LIFO) basis.
- Entries for similar records are not necessarily contiguous.

Page size for File 5 Traversal or navigation through the logical structure of a database occurs when more than one record in a data tree must be accessed. Such an application might use the following command:

```
PRINT EMPLOYEE NUMBER WHERE START DATE LT 01/01/1988 AT 1:
```

A large page size for File 5 increases the chance that all HT entries needed for a request will reside on the same page. For example, a large page size decreases the number of I/Os required to process a PRINT C0 WHERE... command that selects one or more records and retrieves their descendent records.

Number of pages for File 5 The size of File 5 is directly proportional to the number of data records in the database. The maximum number of data records is 119,304,647.

File 6 Size

The Data Table (DT) is the only database table residing on File 6. The table contains the data in data record format. The software accesses each record by using a pointer from the HT. Space is reused by like data records.

Page size for File 6 The DT is unique among the database files in that page boundaries are ignored. Use the guidelines in **Selecting Page Sizes for Database Files** on page 4-8 to determine the page size for File 6.

Number of pages for File 6 The size of this file depends on the number and size of the data records in the database. The maximum number of tracks that can be used for database File 6 is 65,535 (X'FFFF'). Thus, the maximum number of bytes for File 6 depends on the block size and disk device. For example, a 3380 disk using a block size of 23,464 (two blocks per track) could use up to 131,071 blocks or 3,075,426,480 bytes (non-numeric characters).

File 7 Size

All of File 7, which contains updates in processed form, is used for update logging. File 7 (the Update Log) is used with the Rollback Log when the Rollback Log is enabled and Coordinated Recovery becomes necessary.

Page size for File 7 File 7 is a BDAM file, and it is considered a database file. A large page size for File 7 produces fewer I/Os, but increases the amount of data left unwritten in memory in the event of a failure. A small page size increases I/O somewhat but is safer.

File 8 Size

File 8 is the Rollback Log, which contains before-images of modified database table pages. The Rollback Log is used with the Update Log when the Rollback Log is enabled and Coordinated Recovery becomes necessary. For information about Coordinated Recovery, allocating with the ALLOC command, and resetting the Rollback Log, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition*.

Page size for File 8 The page size for the Rollback Log depends upon the largest page size for database Files 1 through 6. Therefore, SYSTEM 2000 software automatically determines the page size for File 8.

ALLOCATING DATABASE FILES

Before you can define a database, all six database files must be allocated. You can dynamically allocate new database files within an SCF session with the default sizes set by SYSTEM 2000 software. Or you can dynamically alter the space and block sizes with the CONTROL language ALLOC command.

If either the Update Log (File 7) or the Rollback Log (File 8) is active, SYSTEM 2000 software can dynamically allocate this file also, or you can issue the CONTROL language ALLOC command.

The sections below discuss allocation strategies, how to dynamically allocate database files, and how to specify DD statements for database files.

Strategies for Allocating Space

You can let SYSTEM 2000 dynamically allocate database files, or you can plan for future applications requirements and allocate space according to those long-term needs. You could also simply fulfill initial requirements for storing the database definition and then gradually increase space allocation as needed. Or, you could create a prototype database to test an application (see **Prototyping Databases** on page 4-62).

Formatting the Files in the Database Definition Step

SYSTEM 2000 software automatically formats database files when a new database is created or restored.

- When a database is created (with the NEW DATA BASE IS command), SYSTEM 2000 software writes each database file (starting with File 6) by filling each data set with binary zero records until the first extent of the last physical volume is allocated. The length of the records equals the page size for the file.

- When a database is restored, the saved database (Savefile) is copied to the allocated files. Unused file space is formatted as described above.

A secondary extent provides a cushion for unexpected space requirements that would otherwise cause SYSTEM 2000 software to end processing and flag the database as damaged. A small amount of secondary extent space specified with each database file protects against such circumstances. If SYSTEM 2000 software runs out of space on a database file during processing, it attempts to format a secondary extent for that file (if a secondary extent was specified when the file was allocated).

Dynamically Allocating Database Files

With Version 12 and subsequent releases, you do not need to allocate any database files in the JCL, CLIST, or Multi-User initialization step. SYSTEM 2000 software checks for allocations specified in single-user jobs and in the Multi-User initialization step. Also, you can use EXIT03 to dynamically allocate the files.

If no allocations exist, the software looks for permanent files having the appropriate database data set names, that is, the first seven characters (with Xs substituted for

| embedded blanks), followed by an integer from 1 to 8. The software searches for Files 1 | through 8, but considers the database files to be allocated if it finds Files 1 through 6.

| If the software's search for allocated files is unsuccessful, an error occurs, and message | -766- DATABASE OPEN FAILED FOR *DDname* - appears.

| To have strict control over which databases are used in a Multi-User environment, you can | set up an S2KDBCNT table. This table contains the names of available databases and other | allocation specifications for existing databases. The software searches this table for | allocations, according to the value for the ALLOC execution parameter. ALLOC=YES allows | dynamic allocation with the ALLOC command or the S2KDBCNT table. ALLOC=NO | disallows dynamic allocation. ALLOC=TBL allows dynamic allocation of only the databases | specified in the S2KDBCNT table. For more details about the ALLOC execution parameter, | see the next topic. For information about the S2KDBCNT table, see **S2KDBCNT Table** on | page 4-17.

| ALLOC=YES allows users to allocate and deallocate database files within an SCF session. | The CONTROL language ALLOC command is not required for existing databases unless you | have used a different data set name than the standard one or you want DISP=SHR. To | create a new database, you must issue the ALLOC command with the minimum syntax of | the database name and DISP=NEW; optionally, you can also specify the space, block sizes, | and so on for the new files or use the defaults.

| SYSTEM 2000 software also dynamically allocates Savefiles and Keepfiles if they are not | already allocated in JCL or CLISTs.

| For more information about dynamic allocation and deallocation of database files, Savefiles, | and Keepfiles, see the NEW DATA BASE IS, DATA BASE NAME IS, ALLOC, FREE, SAVE, and | RESTORE commands in *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

| The following topics discuss three execution parameters that affect dynamic allocation and | deallocation of files, ALLOC, FREE, and PREFIX.

| **ALLOC execution parameter** The ALLOC execution parameter controls dynamic allocation | of database files and varying databases offline/online.

```
|    ALLOC=|YES              (default)
|          |NO
|          |TBL
```

| ALLOC=YES is the default, and it is the most flexible option. ALLOC=YES allows dynamic | allocation and deallocation of new or existing database files. Users can issue the CONTROL | language ALLOC and FREE commands, or they can dynamically access databases specified | in the S2KDBCNT table. For existing database files, SYSTEM 2000 software will search for | the files automatically if there is no ALLOC command or S2KDBCNT table. In order for you | to allocate new database files dynamically, the ALLOC execution parameter must equal YES, | since you must issue the ALLOC command naming the new database.

| ALLOC=YES also allows the console operator to vary databases offline/online in a | Multi-User environment (see **Varying Databases Offline/Online** on page 3-79).

ALLOC=NO disallows dynamic database file allocation and deallocation. The CONTROL language ALLOC and FREE commands are not honored, and the software does not search the S2KDBCNT table for allocations. NO means that all allocations for database files must be in the JCL or CLIST for the start-up job stream or that the files are allocated with user exits. (ALLOC=NO means that SYSTEM 2000 software performs as it did in releases prior to Version 12.)

ALLOC=TBL restricts dynamic allocation and deallocation to those databases defined in the S2KDBCNT table. TBL is useful for a Multi-User environment in which you want to have strict control of databases accessible. Note that if ALLOC=TBL, users cannot issue the CONTROL language ALLOC command to create new databases dynamically or to override the information in the S2KDBCNT table.

FREE execution parameter The FREE execution parameter controls whether SYSTEM 2000 software can deallocate database files when SYSTEM 2000 software terminates.

```
FREE=|NO          (default)
     |YES
```

YES allows the software to deallocate database files that were dynamically allocated. NO means that deallocation will not take place at exit time. NO is the default.

PREFIX execution parameter The PREFIX execution parameter must be one to 35 characters long, in a valid format for data set names. For single-user batch jobs as well as Multi-User software, the prefix precedes the database file names, unless you provide the database data set name (DSN option) in quotes in an SCF ALLOC command. The prefix also precedes the S2KPAD*nn*, SF*nn*, Savefile, Keepfile, S2KDBCNT, and S2KPARMS files when they are dynamically allocated.

```
PREFIX=prefix
```

Note: You must specify the PREFIX parameter in the EXEC statement if you want the software to dynamically allocate the S2KPARMS file.

For TSO users, the TSO user-prefix specified in the PROFILE is used if the PREFIX parameter is omitted. For non-TSO users, if the PREFIX parameter is not set, it is omitted from the data set name; there is no default.

S2KDBCNT Table The S2KDBCNT table enables you to have strict control over which databases users can access dynamically. You can set up an S2KDBCNT table for either single-user or Multi-User environments.

The ALLOC execution parameter controls usage of the S2KDBCNT table. You must set ALLOC=TBL in order to restrict dynamic allocation to only those databases specified in the S2KDBCNT table.

Note: ALLOC=YES is the default. This setting allows dynamic allocation by the software, by the CONTROL language ALLOC command, or by the S2KDBCNT table. ALLOC=NO disallows all dynamic allocation of database files.

To set up the S2KDBCNT table, you specify one or more entries, each containing a database name, its data set name if not the standard DSN, and, optionally, its disposition (DISP). By default, DISP=OLD for all specified databases unless you specify a disposition of SHR. The files for the databases must exist; that is, you cannot specify new databases in the table.

Savefiles and Keepfiles are not allocated by the S2KDBCNT table when the database files are allocated. They can be dynamically allocated when you issue the SAVE, RESTORE, APPLY, and KEEP commands. If so, the Savefile and Keepfile will have the same data set name with a suffix of S or K, respectively, unless you specify a data set name in the SAVE or RESTORE command.

The format of the S2KDBCNT table is:

Field	Beginning Column	Length	Description
DBN	1	16	Database name
OFFLINE	17	1	Flag showing the database offline
DSN	18	43	Data set name for the database files
DISP	62	3	Disposition to be used for dynamic allocation, that is, OLD or SHR. OLD is the default.

SYSTEM 2000 software dynamically allocates the S2KDBCNT data set if you have not specified it in the JCL or CLIST. The data set name must be

```
S2KDBCNT DSN=prefix.S2KDBCNT,DISP=SHR
```

The value of the PREFIX execution parameter is used if it exists. For TSO single-user jobs, the TSO user prefix is used if the PREFIX parameter is not specified. Notice that the disposition is SHR since the table can be shared by multiple users.

Here is a sample S2KDBCNT table with three entries. The LIBRARY database has a disposition of OLD by default. The disposition of SHR is specified for the EMPLOYEE database.

```
1               17                      61
LIBRARY         S2K.LIBRARY
EMPLOYEE        S2K.EMPLOYEE            SHR
FINANCE         MY.FINANCE              OLD
```

Increasing space allocation

By saving and then restoring a database, you can alter the space allocation for the database files as the database expands or contracts. After you save a database, you can scratch the database files and reallocate them with larger or smaller space requirements. You can then restore the database and use the new allocations. See *SYSTEM 2000 CONTROL Language, Version 12, First Edition* for a discussion of the SAVE and RESTORE commands.

Specifying DD Statements for Database Files

If you specify DD statements to allocate the files (instead of allocating them dynamically), they must conform to these rules:

- If the files do not exist, their DD statements must include a DDname and the following parameters:

  DSN, UNIT, SPACE, DCB=(BLKSIZE), VOL=SER, and DISP=(NEW,,)

- If the files exist, their DD statements must include a DDname, a DSN parameter, and DISP=OLD or DISP=SHR. If the files exist but are not cataloged, their DD statements must also include UNIT and VOL=SER parameters.

  DISP=SHR for database files allows all retrievals and updates in a Multi-User environment, and it allows long-running retrievals in single-user jobs, for example, REPORT language jobs or PLEX reports. DISP=SHR places no restrictions on Multi-User jobs, but it disallows updates of any type in single-user jobs and opens the files with read-only access.

  If DISP=SHR in a single-user job, SYSTEM 2000 Error Code 815 is issued for any attempt to update the shared database (even if Multi-User is not up). Disallowed updates include changes to data or the definition and any other commands that alter the status of the shared database, for example, CREATE INDEX, SEPARATOR IS, or SUSPEND.

  Note: Multi-User owns the shared database files. Therefore, if a shared database is being updated under Multi-User while a single-user job is retrieving data, the single-user output can contain out-of-date or conflicting information. The result could be incorrect data from File 6, or it could be SYSTEM 2000 Error Code 812 if the page was not found on disk. The error would affect only the single-user job execution; it would not damage the database.

  Updates made to databases that are shared under more than one Multi-User system are not protected.

The next three subsections discuss DDnames, UNIT assignments, and BLKSIZE specifications. The last subsection shows sample DD statements.

Assigning DDnames The DDname assigned to a database file has eight characters. It consists of the first seven characters of the database name, with any blanks being replaced by X and with the file number as the eighth character. If the database name has fewer than seven characters, X is used as a filler. Files 1 through 6 constitute the database; File 7 and File 8 are the optional Update Log and Rollback Log, respectively. Here are some sample DDnames for various database files:

- The DDname for File 1 of database PERSONNEL is PERSONN1.
- The DDname for File 3 of database JUNE SALE RECORDS is JUNEXSA3.
- The DDname for File 7 of database PARTS is PARTSXX7.

Allocating database files to different devices Before you specify the UNIT parameter, consider that you can minimize interference by allocating database files to different devices, as shown in the examples below. These examples are intended only as guidelines for allocating files to various devices. DDnames DATABAS1, DATABAS2. DATABAS3, DATABAS4, DATABAS5, and DATABAS6 denote the six database files. Each SPINDLEn denotes a separate online disk drive or pack.

Example 1

If there are many distinct values for key items, you could specify

```
SPINDLE 1 - DATABAS1, DATABAS4, DATABAS6
SPINDLE 2 - DATABAS3, DATABAS5
SPINDLE 3 - DATABAS2
```

Example 2

If there are many multiple occurrences of key item values, you could specify

```
SPINDLE 1 - DATABAS1, DATABAS4
SPINDLE 2 - DATABAS3, DATABAS5
SPINDLE 3 - DATABAS2, DATABAS6
```

Specifying the BLKSIZE subparameter As mentioned before, you should take time in assigning a page size (BLKSIZE value) to a database file. Read **Database File Sizes** on page 4-6 and determine the page size for each database file. Before you actually assign a block size, consider these factors:

- The page size and space requested for the file should fit the available resources. Consider changing the page size to make better use of a particular disk device.
- The page size should be compatible with the buffer size specified in at least one POOL*n* execution parameter (see **POOLn Parameter for Allocating I/O Buffer Pools** on page 4-24).
- Default block sizes are not necessarily optimal. (Their primary purpose is upward compatibility.) Particularly in a VS environment, consider specifying a buffer size of 4060 due to frame alignment. When SYSTEM 2000 software accesses data, it uses a 36-byte header; consequently, a 4060-block size allows an exact 4K frame alignment.

If you do not specify a value for BLKSIZE, SYSTEM 2000 software assigns a default block size that depends on the current value of the DISK execution parameter described in **DISK Execution Parameter** on page 4-21. By default, DISK is set to 3380; therefore, BLKSIZE=6216.

Example 1

Here are the DD statements for allocating a database named SUPPLIERS.

```
//SUPPLIE1 DD DSN=PROD.SUPPLIERS1,DCB=(BLKSIZE=4060),UNIT=SYSDA,
//            VOL=SER=STR01,DISP=(NEW,CATLG),SPACE=(CYL,(1,1))
//SUPPLIE2 DD DSN=PROD.SUPPLIERS2,DCB=(BLKSIZE=4060),UNIT=SYSDA,
//            VOL=SER=STR01,DISP=(NEW,CATLG),SPACE=(CYL,(1,1))
//SUPPLIE3 DD DSN=PROD.SUPPLIERS3,DCB=(BLKSIZE=4060),UNIT=SYSDA,
//            VOL=SER=STR01,DISP=(NEW,CATLG),SPACE=(CYL,(1,1))
//SUPPLIE4 DD DSN=PROD.SUPPLIERS4,DCB=(BLKSIZE=4060),UNIT=SYSDA,
//            VOL=SER=STR01,DISP=(NEW,CATLG),SPACE=(CYL,(1,1))
//SUPPLIE5 DD DSN=PROD.SUPPLIERS5,DCB=(BLKSIZE=4060),UNIT=SYSDA,
//            VOL=SER=STR01,DISP=(NEW,CATLG),SPACE=(CYL,(1,1))
//SUPPLIE6 DD DSN=PROD.SUPPLIERS6,DCB=(BLKSIZE=4060),UNIT=SYSDA,
//            VOL=SER=STR01,DISP=(NEW,CATLG),SPACE=(CYL,(1,1))
//SUPPLIE7 DD DSN=PROD.SUPPLIERS7,DCB=(BLKSIZE=4060),UNIT=SYSDA,
//            VOL=SER=STR01,DISP=(NEW,CATLG),SPACE=(CYL,(1,1))
//SUPPLIE8 DD DSN=PROD.SUPPLIERS8,DCB=(BLKSIZE=4060),UNIT=SYSDA,
//            VOL=SER=STR01,DISP=(NEW,CATLG),SPACE=(CYL,(1,1))
```

Example 2

This example shows the DD statements for the PARTS database. It assumes the files are cataloged.

```
//PARTSXX1 DD DSN=PROD.PARTS1,DISP=OLD
//PARTSXX2 DD DSN=PROD.PARTS2,DISP=OLD
//PARTSXX3 DD DSN=PROD.PARTS3,DISP=OLD
//PARTSXX4 DD DSN=PROD.PARTS4,DISP=OLD
//PARTSXX5 DD DSN=PROD.PARTS5,DISP=OLD
//PARTSXX6 DD DSN=PROD.PARTS6,DISP=OLD
//PARTSXX7 DD DSN=PROD.PARTS7,DISP=OLD
//PARTSXX8 DD DSN=PROD.PARTS8,DISP=OLD
```

DISK Execution Parameter

DISK=*device*

device is one of the following devices: 3310, 3350, 3370, 3375, 3380, or 3390. The default is 3380, which is an efficient block size (6216) that fits on all disk types.

SYSTEM 2000 software uses the DISK value to

- assign a default page size (BLKSIZE value) to database files (see **Specifying DD Statements for Database Files** on page 4-19).
- assign a default buffer size for POOL0 when no POOL execution parameter is specified (see **POOLn Parameter for Allocating I/O Buffer Pools** on page 4-24).

The default block sizes and buffer sizes for the various disks are as follows:

DISK	Default POOL0 buffer size and BLKSIZE
3310	4096
3350	4592
3370	4096
3375	4060
3380	6216
3390	6216

The OPT012 execution parameter is available to change the default block size. (Also, for R11.5 and R11.6, a zap is available to support 3390 disks with current block sizes.)

Note: If the DISK parameter indicates a page size that is larger than the largest pool size specified for database usage, the block size defaults to the page size of the largest pool that was specified for database use.

DETERMINING I/O BUFFER REQUIREMENTS

This section describes the SYSTEM 2000 I/O buffer and how to determine its space requirements.

- The next topic describes the I/O buffer and pools of buffers.
- **POOLn Parameter for Allocating I/O Buffer Pools** on page 4-24 describes the POOLn execution parameter.
- **Using Pools** on page 4-29 describes how to use the POOLn parameter.
- **Clearing Pages** on page 4-34 describes how SYSTEM 2000 software handles the process of clearing pages from memory to disk.

I/O Buffer Description

To allow more control of space resources and to reduce database and thread contention, SYSTEM 2000 software groups I/O buffers into pools. A *pool* is a group of buffers of identical size with identical use permission. Use the POOLn execution parameter to specify the three attributes of each pool:

- the size of the buffers
- the number of buffers
- the use of the buffers:
  - for database files only
  - for work files only
  - for both database files and work files.

SYSTEM 2000 software allows up to eight pools: POOL0 through POOL7.

When you initialize Multi-User or when you start a single-user job, SYSTEM 2000 software uses the information in the POOL parameter(s) to acquire (GETMAIN) space within the region for the I/O buffers. You must specify enough of these buffers to fit your requirements.

- For a single-user job, you must have enough buffers available for database and work file use.
- In a Multi-User environment, you must have enough buffers for database and work file use for each user.

POOLn Parameter for Allocating I/O Buffer Pools

Use the POOLn parameter to specify the attributes of each I/O buffer pool you need. The syntax of the parameter is shown here. **Setting SYSTEM 2000 Execution Parameters** on page 5-9 gives details about how to specify execution parameters.

POOL*n*=*buffer-size*/*buffers*[/*usage*]

n — a number designating the pool. The range is 0-7, that is, POOL0, POOL1, and so on. If you specify more than one POOL*n* parameter, the pool numbers must be consecutive, starting with zero.

| *buffer-size* — the number of bytes in each buffer within a pool. The range is 2016-27972. The buffer size for POOL*n* must be less than or equal to the buffer size for POOL*n*+1, and so on. The values for *buffer-size* are listed in Illus. 4.3 on page 4-26.

buffers — the number of I/O buffers in the specified pool. The range is 0-999.

usage — five alphabetic codes designating the use of the I/O buffers in the pool:

- D allows only database files whose page size is equal to or less than *buffer-size*. Work files cannot use buffers in this pool.
- DE allows only database files whose page size is equal to *buffer-size*. Work files cannot use buffers in this pool.
- S allows only work files whose page size is equal to or less than *buffer-size*. Database files cannot use buffers in this pool.
- B allows both database files and work files whose page size is equal to or less than *buffer-size*.
- BE allows both database files and work files whose page size is equal to *buffer-size*.

Note that SE is not listed; if you specify SE, the E is ignored.

If you do not specify POOL parameters, SYSTEM 2000 software sets POOL0 to be

POOL0=*buffer-size*/14/B

The default buffer size depends on the current value of the DISK execution parameter described in **DISK Execution Parameter** on page 4-21. By default, DISK is set to 3380; thus, POOL0=6216/14/B. If the usage option is omitted, B usage is the default.

If no POOL parameter is specified and the TPTHREADS parameter is greater than zero, SYSTEM 2000 software also sets up the following default pool:

```
POOL1=11592/n/B          (for the S2KUSERS file)
```

where *n* is the value of THREADS or TPTHREADS, whichever is smaller. See **THREADS Parameter** on page 3-62 and **TPTHREADS Parameter** on page 3-63 for details about those parameters.

If duplicate POOLn specifications are given, for example, two POOL0 specifications, the last one overrides all previous ones.

Because SYSTEM 2000 software scans the table of pools in ascending order, place pools with DE or BE usage before other pools of the same size. For example,

```
POOL1=2492/8/DE
POOL2=2492/10/D
```

When SYSTEM 2000 software initializes the buffers, it validates the requested buffer sizes. At the same time, SYSTEM 2000 software assigns the page size for scratch files or scratch pads. The page size is set equal to the smallest buffer size specified in pools with usage S, B, or BE. The page size for sort files is half the buffer size. See Illus. 4.3 on page 4-26. If the software cannot allocate buffer space from the region assigned, user abend code 115 is generated.

SYSTEM 2000 software issues a SYSTEM 2000 Error Code when it attempts to process a command or function and finds that the number of buffers specified was not enough. SYSTEM 2000 software does an automatic "clear" before issuing the Error Code to minimize the possibility of damaging the database.

Illus. 4.4 on page 4-27 shows the pages per track and percentage of space used for the various block sizes on 3390, 3380, and 3350 disks.

Illus. 4.3 Buffer and Page Size Values

Buffer Size	Database Page Size	Scratch File Page Size	Sort File Page Size	Actual Buffer Length*	Comments
2016	2016	2016	1008	2048	1/6-track-3330**
2048	2048	2048	1024	2080	FBA devices (3310, 3370)
2492	2492	2328	1164	2520	1/5-track-3330
3136	3136	3016	1508	3164	1/4-track-3330
3472	3472	3376	1690	3500	1/2-track-2314
4060	4060	3920	1960	4096	VS frame alignment** 1/3-track-3330 1/2-track-3340 1/8-track-3375
4096	4096	4096	2048	4132	FBA devices (3310, 3370)
4592	4592	4432	2216	4620	1/4-track-3350
5600	5600	5600	2800	5628	1/2-track-3340 CMS support
6216	6216	6184	3092	6244	Portable between 3330, 3350, and 3380, 3390: 1/2-track-3330 1/3-track-3350 1/7-track-3380 1/8-track-3390
6440	6440	6312	3156	6468	1/2-track-3330
7524	7524	7524	3762	7552	1/7-track-3390
8316	8316	8192	4096	8344	full-track-3340
9408	9408	9248	4624	9436	1/2-track-3350
11592	11592	11176	5588	11620	1/3-track-3375
12236	12236	12232	6118	12288	full-track-3330 (frame-aligned)
12992	12992	12888	6444	13020	full-track-3330
15456	15456	14952	7476	15484	1/3-track-3380
17584	17584	17192	8596	17612	1/2-track-3375
19040	19040	18880	9940	19068	full-track-3350
23464	23464	22920	11460	23492	1/2-track-3380
27972	27972	27360	13680	28000	1/2-track-3390

* The actual buffer length includes a header used only by SYSTEM 2000 software.

** Under virtual storage operating systems, these buffers take advantage of VS frame-alignment efficiencies.

Illus. 4.4 Percentage of Space Used by Block Size for 3390, 3380, and 3350 Disks

***** 3390 Disk ****** **Track Byte Capacity 56664**

Block Size	Pages Per Track	Percentage of Space Used
2016	21	74
2048	21	75
2492	18	79
3136	15	83
3472	13	79
4060	12	85
4096	12	86
4592	10	81
5600	09	88
6216	08	87
6440	08	90
7524	07	92
8316	06	88
9408	05	83
11592	04	81
12236	04	86
12992	04	91
15456	03	81
17584	03	93
19040	02	67
23464	02	82
27972	02	98

***** 3380 Disk ****** **Track Byte Capacity 47476**

Block Size	Pages Per Track	Percentage of Space Used
2016	18	76
2048	18	77
2492	15	78
3136	13	85
3472	12	87
4060	10	85
4096	10	86
4592	09	87
5600	07	82
6216	07	91
6440	06	81
7524	05	79
8316	05	87
9408	04	79
11592	03	73
12236	03	77
12992	03	82
15456	03	97
17584	02	74
19040	02	80
23464	02	98
27972	01	58

continued on next page

Illus. 4.4 *continued*

***** 3350 Disk ****** **Track Byte Capacity 19069**

Block Size	Pages Per Track	Percentage of Space Used
2016	08	84
2048	08	85
2492	07	91
3136	05	82
3472	05	91
4060	04	85
4096	04	85
4592	04	96
5600	03	88
6216	03	97
6440	02	67
7524	02	78
8316	02	87
9408	02	98
11592	01	60
12236	01	64
12992	01	68
15456	01	81
17584	01	92
19040	01	99
23464	00	00
27972	00	00

Using Pools

This section discusses some considerations for determining pool requirements. Basically, you want to provide enough I/O buffers to handle database file and work file use. For Multi-User environments you must also consider the number of users of the system.

In a Multi-User environment, you may want a variety of pools with different sizes and different uses. As shown in the examples at the end of this section, you can reduce buffer contention by having separate pools for work files and for database files, separate pools for individual databases, and even separate pools for individual database files.

When SYSTEM 2000 software needs an I/O buffer, it scans the table of pools in ascending order to find the appropriate buffer needed. If there is a choice of pools, a requested small page uses a larger buffer pool member only if smaller buffers are not available (that is, in a locked state).

Using pools for scratch files Scratch files can be allocated using scratch pads or, in some cases, individually. (See **Allocating Work Files** on page 4-37.) Regardless of how scratch files are allocated, a scratch file locks one buffer until processing of the data in the scratch file is complete.

You must provide at least four buffers for work file use. That is, you must specify at least one pool with four buffers and usage of S, B, or BE. Failure to provide at least four buffers for work files causes user abend 101 at initialization time.

Some commands can require more than four buffers. For example, most commands involving large scratch file operations (initial loads, incremental loads, or volume where-clause processing), need a minimum of five work file buffers. Also, when using the REPORT processor, each COMPOSE session can require as many as 10 work file buffers.

If you anticipate scratch file sorts, reloads, large batch updates, and so on, specify work file buffers with large buffer sizes to increase the scratch file page size.

Discarding PLEX Locate Files frees buffer space in memory and on disk. (See **Work Files: Overview** on page 4-37.) A Locate File retains one work file buffer from a pool for as long as the Locate File exists. A buffer is released immediately after a specific Locate File is discarded.

Using pools for database files The block size specified when database files are allocated (see **Specifying DD Statements for Database Files** on page 4-19) determines the page size for the database files. Therefore, the block size is significant when you are choosing the size and number of pools to support the database(s).

Usage of I/O buffers for database files depends on the processing that is involved.

Database definition buffers When you create or modify a database definition, the SYSTEM 2000 buffer manager obtains buffers from the pools that were specified by the POOLn execution parameters. The number of buffers required in a DEFINE session depends on three factors:

- MAXCOMP, which determines the maximum number of components that can be defined for the database. You specify MAXCOMP in the NEW DATA BASE IS

command. For example, NEW DATA BASE IS COMPANY/5000 specifies a maximum of 5,000 components. The number includes schema items, schema records, strings, and functions. The default is 430. For more details, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

- the LDBSIZE execution parameter, which determines the maximum number of components that can be processed in the current SYSTEM 2000 session. The default is 1000. LDBSIZE determines the largest DEFBLOCK that can be used during this session; it also determines the maximum MAXCOMP for any database that can be opened (or created) in this session.

- the buffer size and usage that was specified for each pool. For more information about setting up pools, see the discussion of the POOL*n* execution parameter.

DEFINE processing consists of two phases. During Phase 1, SYSTEM 2000 software scans the new component definitions (or modifications) and builds a temporary definition. Phase 2 finalizes the definition at MAP time and stores it in database Files 1 and 3.

Phase 1 uses buffers from the pool with the largest buffer size having D , DE. or B usage. The buffer manager uses the following formula to calculate the number of buffers needed:

```
(((MAXCOMP x 28)/bufsiz) + 4 + 2)
```

where *bufsiz* is the number of bytes in the largest buffer having D, DE, or B usage. The additional six buffers do not necessarily come from the same pool. Four of the buffers are for scratch files (S or B usage), and the other two are for database Files 1 and 3 (D, DE, or B usage).

Phase 2 requires the following number of bytes:

```
(LDBSIZE x 4)
```

If one or two buffers (from any pool with any usage) are adequate, the buffer manager uses that pool. If more than two buffers are needed, the buffer manager acquires them from the pool with the largest buffers that were specified for database usage (D, DE, or B).

Note: Buffers acquired during Phase 1 are freed before Phase 2. Therefore, in most situations, the buffers used in Phase 1 can be reused in Phase 2.

For example, if MAXCOMP equals 10,000 and LDBSIZE equals 10,000, the following POOL*n* specification should be sufficient:

```
POOL0=6216/52/B
```

This pool specification sets up fifty-two buffers having a block size of 6216 with B usage, that is, for either scratch files or database files.

You could also specify these pools:

```
POOL0=6216/4/S
POOL1=23464/14/D
```

POOL0 sets up a pool of four buffers having a block size of 6216 for scratch file usage. POOL1 contains fourteen buffers having a block size of 23464 for database usage.

Note: The example and formulas show absolute minimums. To prevent buffer steals, allocate ten or more buffers beyond the minimum number required for database usage. However, there is no advantage to allocating more scratch file buffers than required.

In addition, the number of I/Os to the scratch pad is proportional to the block size; the larger block sizes require fewer I/Os. For example, a scratch pad blocked at 6216 bytes experiences about four times the I/Os of a scratch pad blocked at 23464. The number of I/Os affects real time for the job, but it does not affect CPU time.

If the required number of buffers for LDBSIZE is not available when SYSTEM 2000 software is invoked, WTO message 0116 is issued, followed by user abend 101. See *SYSTEM 2000 Messages and Codes, Version 12, First Edition* for explanations of these error messages.

Adequate buffer space for File 3, the EFT, prevents excessive I/O during definition processing. Component definitions mapped prior to the current DEFINE/MAP session are stored in Files 1 and 3. File 1, which contains the Definition Table and Master Record, resides in memory during execution. File 1 and File 3 are described in **File 1 Size** on page 4-9 and **File 3 Size** on page 4-11, respectively.

When a component declaration is processed, SYSTEM 2000 software checks existing component names and numbers to ensure that new component labels are unique. While a particular database is in use, the component numbers from File 1 are stored in the memory-resident Definition Block (see **Database Definition Block Sizes** on page 4-54). SYSTEM 2000 software accesses component names longer than four characters through the buffer pools provided for File 3. Because most component names are longer than four characters, File 3 is usually accessed. If you conserve memory requirements for buffer pools by providing only one buffer for the File 3 page size, buffer thrashing might occur. Providing more buffers allows the software to check for duplicate names without excessive I/O for File 3.

Usually DEFINE processing in single-user jobs uses only Files 1 and 3. There is no reason to provide a generous buffer allocation for other files. All database files are used only if definition modifications force database restructuring, which needs additional buffers.

Database access During database access (DBN IS or PLEX OPEN), one buffer is locked in the pool with the largest buffers that can accommodate a database file page (D, DE, B, or BE usage). The buffer is released when the open processing completes.

This can cause problems in a Multi-User environment if this pool is also used by the S2KUSERS file. The S2KUSERS file uses one 11592-byte buffer for each active TP session (see **TP requirements for pools** on page 4-32).

SAVE/RESTORE operations SYSTEM 2000 software needs two buffers for restoring a database and two buffers for saving a database. If Update Logs are in use for one or more databases in any session, coordination and standardization of page size needs to be considered, especially for Multi-User environments.

The DBBUFN execution parameter allows you to specify the number of QSAM buffers to be used for database files while saving or restoring a database. The default is five. To obtain more than five buffers for the Savefile, you could also specify DCB=BUFNO=*n* in the JCL for xxxxxxxS. QSAM buffers are acquired when the database files (or Savefile) are opened QSAM for a SAVE or RESTORE operation. They are released when the files are closed. These QSAM buffers are separate from the buffer pools.

Rollback Log If the Rollback Log is enabled, it writes the before-images of the database pages from the pool buffer containing the database page. (See *SYSTEM 2000 CONTROL Language, Version 12, First Edition* for information about resetting the Rollback Log.)

If the database becomes damaged and Coordinated Recovery takes place, the recovery processors request use of one buffer for File 8. While reinstating the database, Coordinated Recovery reads and writes from that buffer. The File 8 block size is 12 bytes greater than the largest page size for Files 1 through 6 in that database. If that size were 6228, for example, a buffer would be requested from any pool that would accommodate 6216.

TP requirements for pools A single buffer pool with an 11592-byte block size is required for the S2KUSERS file described in **S2KUSERS File** on page 3-30. Also, for non-MVS TP users, one buffer with a 4060 block size is required for the output of each TP user. For MVS, the output buffers are obtained from the CSA (Common Service Area) at initialization for an SVC Multi-User system or from the Multi-User private region for a Cross Memory Services (XMS) Multi-User system.

If you specify any POOLn parameter at Multi-User initialization time and you request the TP facility by specifying a value greater than zero for the TPTHREADS execution parameter, you must specify one pool with a block size equal to 11592 and with D, DE, B, or BE usage. The S2KUSERS file can be accommodated in a pool with S usage, but extra I/O can result from contention with work files. If more than one pool has a block size of 11592, the highest pool number (n) is used for the S2KUSERS file regardless of its usage.

If an 11592 pool is specified and it is to be used by other database files or work files, ensure that the pool has enough buffers. There must be enough buffers for all possible concurrent TP sessions. If possible, specify the 11592 pool with D usage (for database file pages). Otherwise, if the S2KUSERS file shares with work file buffers (S, B, or BE usage), a SYSTEM 2000 Error Code can occur, because both work file buffers and S2KUSERS buffers are locked while in use. S2KUSERS buffers are not forced out when the TP session ends unless a buffer is needed by some other run-unit. Therefore, I/O to the S2KUSERS file can be reduced by allocating more buffers to the 11592 pool or even eliminated by allocating enough buffers to handle the maximum possible concurrent TP run-units. See **TPSCRUN Parameter** on page 3-62.

The rules for assigning S2KUSERS buffers provide minimum assignment for avoiding deadlock. If memory is not critical and if more than TPTHREADS terminals are signed on at any given time, turnaround can be improved by providing enough S2KUSERS buffers to minimize swapping.

Buffer assignment and eviction When a buffer is required, the software assigns a buffer that is adequate for both the usage and size required. The Buffer Manager software checks to see if there is an available buffer from any pool that matches the needed size and usage. If so, it assigns a buffer from that pool.

If all buffers of the proper size and usage are currently assigned, the Buffer Manger software must evict a current buffer and assign it to the latest request. The software's procedure consists of the following steps:

1. Find the "best fit" pool to evict a buffer.
2. Find the oldest buffer in that pool.
3. If the oldest buffer contains an update, write it out to disk before reassigning it.
4. Unlink that buffer from the previous owner.
5. Assign that buffer to the new owner.

Multiple Local Holds buffer PLEX programs can issue the HOLD option to hold specific records while updating. You can allow multiple local holds for a database by issuing the ENABLE MULTIPLE HOLDS command (see *SYSTEM 2000 CONTROL Language, Version 12, First Edition*). If multiple local holds are enabled, each database must have a buffer to hold the addresses of records that are being held at any given time.

For Version 12 and later releases, SYSTEM 2000 software does a GETMAIN to obtain a Multiple Local Holds (MLH) buffer. The default size is 23464, the same size as the maximum buffer size allowed by Release 11.6. User exit EXIT07 can be used to alter the default size with a range of 512 bytes to 50,000 bytes.

For MVS/XA and VM/XA, the buffer is obtained with a GETMAIN using LOC=ANY,ANY. For non-MVS/XA systems, the GETMAIN is done only if a size is provided with user exit EXIT07. For non-MVS/XA systems without user exit EXIT07 and for all non-VM/XA environments SYSTEM 2000 software uses a buffer from the largest pool. The software also uses a buffer from the largest pool any time the GETMAIN request fails.

Sample pool assignments

Example 1

Assume that the system is operating in a Multi-User environment with database files distributed between 3380 and 3390 devices. You might also want the VS frame alignment and reduced I/O associated with the 4K (4060) buffers for both system scratch files and some high priority database files. Only database files with 4060-byte page size can use POOL0. These pools cannot accommodate the TP facility.

POOL0=4060/10/BE	Available for database files with page size exactly 4060; available for work files.
POOL1=6216/8/D	Available for database files with page size ≤ 6216.
POOL2=23464/8/D	Available for database files with page size ≤ 23464.

Example 2

With the following pool assignments, POOL1 would be used to determine the page size for all work files. When scratch files use POOL3, one-half of each buffer is wasted. This can be a valid assignment if database files with pages of 12992 are to be processed. If the TP facility is active, there would be buffer contention in POOL3 for database files only if they were blocked at 11592.

POOL0=4060/10/D	Available for database files with page size ≤ 4060.
POOL1=6440/4/B	Available for database files with page size ≤ 6440; available for work files and establishes the work file block size because it is the first pool for work file usage.
POOL2=11592/4/DE	Available for S2KUSERS file; could be used for any database file blocked at 11592.
POOL3=12992/4/B	Available for database files with page size ≤ 12992; available for work files.

Clearing Pages

To alter a database, the pages of the database table that need changes must be brought from disk to memory. SYSTEM 2000 software writes an updated page back to the database disk files when any of the following events occur:

- a session terminates
- a PLEX program terminates
- a CLEAR command is executed
- SYSTEM 2000 software needs to use the buffer.

PLEX, as well as the QUEST processor, provides four CLEAR commands[1] to give some control over clearing updated database pages from memory to disk. These commands also control the clearing of buffers holding information to be written to an active Update Log. The various forms of the CLEAR commands not only clear pages from memory to disk but also set the mode of clearing.

SYSTEM 2000 software maintains an internal damage flag to indicate whether updated pages have been written back to disk. The flag shows a "damaged" status as long as the modified pages have not been written back to disk. The flag shows a "clean" status after the pages are written to disk. For an SCF session, a message is issued when a database becomes damaged, when a damaged database is opened or closed, and when updates are attempted.

[1] See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* and the *SYSTEM 2000 PLEX Manual, Version 12, First Edition* for details about these commands.

XBUF CACHING FEATURE

Introduction

The Extended Buffer (XBUF) Manager feature allows you to use several kinds of caching techniques. You can use XBUF caching for database files in single-user jobs and in a Multi-User environment under MVS. Scratch pads and other temporary work files cannot be cached.

XBUF console commands are available for displaying cache activity. Also, when XBUF is activated, you can use two of the MUSTATS console commands to display read, write, and drop rates for database files and pools. For details about the MUSTATS console commands, see **DBN= Command** on page 3-85 and **POOLS Command** on page 3-94.

Simple XBUF macros are available for

- dual logging of database files
- caching database files to secondary buffers in 31-bit addressable memory
- caching database files to a high-speed disk device
- modeling cache activity
- gathering statistics about SYSTEM 2000 buffer pool and XBUF cache activities.

The simple front-end macros rely on a basic CACHE macro inherent to XBUF caching. They are user-friendly in a way that lets you use XBUF caching without having to learn much about how XBUF caching works internally. For example, you can try dual logging or 31-bit memory quickly, using default settings for the basic XBUF options.

For detailed discussions on how to use the XBUF macros and XBUF console commands, see *Technical Report: S2-107 XBUF Caching Feature in SYSTEM 2000 Software, Release 11.6 under IBM OS*.

Dual Logging

XBUF allows you to specify one or more database files for dual logging. When dual logging is activated, XBUF maintains two identical copies of the selected files. You can request dual logging even if your computer site does not have the expensive high-speed devices usually associated with caching operations.

Dual logging is especially appropriate for database Files 1, 7, and 8. These three files are relatively small, but the information contained in them is absolutely essential for database recovery. If the primary copy of a file is lost, you can continue to process the database by modifying the JCL to point to the second copy.

31-Bit Addressable Buffers

XBUF allows you to use secondary buffers in 31-bit addressable memory. Again, XBUF caching techniques offer advantages that do not require any extra high-speed devices.

XBUF can expand the buffering of database files to memory that can be addressed by 31 bits. Thus, up to 2048 megabytes are theoretically available to hold data without the software having to reference disk data sets. The data that XBUF keeps in 31-bit addressable memory (above the 24-bit memory) cannot be used directly by SYSTEM 2000 software. However, the XBUF instructions that move data to the SYSTEM 2000 buffer area are significantly faster than the I/O operations necessary to bring data from a disk to a buffer pool.

Caching to High-Speed Disk Devices

If your computer site has high-speed disk devices, you can set up cache files on any selected device. For each cache file, you can specify one or more specific database files to participate in that cache.

Simulated Cache Areas

Dual logging, 31-bit addressable memory, and high-speed cache files are not applicable to every database. Since the mix of transactions is unique to each production environment, you may not notice any improvement when using XBUF.

An important XBUF option addresses the question of what XBUF caching would do at your site. Without using any physical resources, XBUF can simulate its own operation and give you statistics on how it would have performed if you had specified real 31-bit memory or high-speed disk caching.

ALLOCATING WORK FILES

This section describes how to allocate space for work files.

Work Files: Overview

The software uses two types of work files: sort files and scratch files. Sort files are used to sort and merge large scratch files. The software uses them when processing an ordering-clause, when sorting internal scratch files that hold addresses, and so on.

Scratch files are work files that store results of intermediate processes. For example, during ASSIGN command processing, SYSTEM 2000 software writes the addresses of selected data records on Scratch File 7. The Keepfile is treated as a work file and so are Locate Files used in PLEX programs.

You allocate scratch file space by defining *scratch pads*, which are direct access files using BDAM for read/write operations. Scratch pads allow SYSTEM 2000 software to associate scratch files within one physical direct access file. This, in turn, allows SYSTEM 2000 software to increase its control over scratch file I/O and to provide more efficient management of scratch file space. **PAD*nn* execution parameter** on page 4-41 describes how to define scratch pads. Also, SYSTEM 2000 software can dynamically allocate scratch pads.

The few situations in which scratch pads are not required are discussed in **Allocating Scratch Files Individually** on page 4-52.

Discarding PLEX Locate Files in a Multi-User environment An existing Locate File can be discarded in a PLEX program by issuing a LOCATE command that qualifies no data records. For example,

```
LOCATE BOOKS WHERE item EQ nonexistent-value;
```

Discarding Locate Files frees pages in scratch pads. This is significant in a Multi-User environment where several users have large Locate Files that are no longer being used. Discarding a Locate File involves little overhead and should be considered for all PLEX programs.

Virtual I/O for temporary data sets For MVS users, see the IBM MVS JCL manual for a discussion of virtual I/O (VIO) for temporary data sets. When you indicate (at system generation) that user-assigned group names, such as SYSDA, or device type names, such as 3330, are eligible for VIO, you can cause SYSTEM 2000 scratch files and sort files to reside with the virtual paging space. If frame-aligned page sizes are specified for the scratch and sort files, this can result in better performance by SYSTEM 2000 software. However, some contention with the operating system's use of the virtual paging space can occur. Carefully consider the restrictions in the IBM JCL manual.

Assigning a scratch pad to VIO is not recommended. When SYSTEM 2000 software attempts to format the primary space, it formats all primary and secondary allocation space. VIO does not distinguish between primary and secondary allocation.

Allocating Sort Files

Sort files are work files that the software uses to sort large scratch files. For example, they are used for an ORDER BY request with many data records.

SYSTEM 2000 software dynamically allocates all the required sort files if they are not allocated in the JCL or CLIST. Alternatively, you can allocate sort files by including appropriate DD statements in single-user jobs or in the job that initializes Multi-User software. These two methods are discussed next.

Dynamically allocating sort files The easiest way to allocate the SYSTEM 2000 sort files is to let SYSTEM 2000 software allocate them dynamically, using all the defaults. Dynamic allocation takes place automatically if SYSTEM 2000 software does not find any DD statements for the SF*nn* files in the JCL.

In addition, four execution parameters are available that enable you to alter the primary space, secondary space, unit, and space unit for sort files that are dynamically allocated. (See **Setting SYSTEM 2000 Execution Parameters** on page 5-9 for information about setting execution parameters.) Most jobs can use the defaults and temporary data sets. However, you can specify all the execution parameters or only the ones you need to alter for dynamic allocation.

Parameter	Default	Minimum	Maximum	Description
SFPRI=s	1	1	32767	Sort file primary space
SFSEC=s	1	0	32767	Sort file secondary space
SFUNIT=x	SYSDA	-	-	Sort file unit device
SFSPACE=y	CYL	(either TRACK or CYL)		Sort file space unit

Thus, the default is SF0*n* UNIT(SYSDA),SPACE(1,1),CYL, where *n* equals the integers 1 through 6 for the six sort files per job (or thread).

SYSTEM 2000 software goes through the following steps, in the order given, when determining whether to dynamically allocate the sort files:

1. If the files were allocated in the JCL or CLIST, the software uses them

2. If no files are allocated in the JCL or CLIST, the software looks for permanent data sets. If files exist and they can be allocated with DISP=OLD, the software uses them. The DDnames must be

   SF*tn* DSN=*prefix*.SF*tn*, DISP=OLD

   where *t* is a thread number minus 1 and *n* is an integer from 1 to 6. (For single-user jobs, *t* equals 0.) The integers for *t* and *n* identify each of the sort files. The *prefix* value equals the value for the PREFIX execution parameter if it was specified. For TSO single-user jobs, the TSO user prefix is used if the PREFIX parameter does not exist. For more information about the *prefix*, see **PREFIX execution parameter** on page 4-17.

3. If SYSTEM 2000 software does not find the sort files in Step 1 or 2, it allocates temporary sort files dynamically. Default values for primary space, secondary space, unit, and unit space are used unless you specify other values with the SFPRI, SFSEC, SFUNIT, and SFSPACE execution parameters.

Dynamic allocation of sort files does not create any new permanent data sets. All data sets allocated are temporary files.

If the sort files are dynamically allocated, they are dynamically deallocated at the end of the single-user job or the Multi-User session. Sort files allocated with JCL and permanent files are never deallocated dynamically.

Allocating sort files with JCL or CLISTs Dynamic allocation is the simplest method for allocating sort files, as discussed in the previous topic. However, if you want to include the sort file DD statements in the JCL or CLIST for a single-user job or in the Multi-User initialization step, consider the following guidelines:

- Sort files do not use scratch pads.
- Sort files can never span volumes.
- You can allocate sort files to disk or tape. But you must allocate at least one sort file to disk. For Multi-User, sort files must be allocated to disk.
- Sort files allocated to disk must all be allocated to the same disk device type. You can optimize processing time by assigning each sort file to a different device.
- SYSTEM 2000 software determines the page size of all sort files at job initialization time. The page size of sort files is exactly one-half the scratch file page size. (The scratch file page size is the smallest buffer size specified in pools having S, B, or BE usage. See **POOLn Parameter for Allocating I/O Buffer Pools** on page 4-24 for details.) Sort files need not be bigger than two-thirds of the largest scratch pad.
- Each thread requires one set of six sort files. The number of threads is set by the THREADS execution parameter.
- The sort file DDnames for thread *t* are: SF*n*1, SF*n*2, SF*n*3, SF*n*4, SF*n*5, and SF*n*6, where $n = t\text{-}1$.
- An example of JCL allocating a sort file to tape is

```
//SF01   DD=SF1,UNIT=(TAPE9,,DEFER),DISP=(NEW,KEEP),
//          VOL=SER=SP0012
```

  An example of JCL allocating a sort file to disk is

```
//SF02   UNIT=SYSDA,SPACE=(TRK,(1,1))
```

Allocating Scratch Pads

As discussed in **Work Files: Overview** on page 4-37, scratch pads are BDAM files used by SYSTEM 2000 software to hold scratch files. When scratch pad space is needed for a scratch file, SYSTEM 2000 software determines which pad has the lowest percentage of used space and assigns the scratch file to that pad. The scratch file remains there until it is closed. Disk space is automatically reclaimed when eviction occurs or the user terminates.

Several scratch files can use the same pad, but a scratch file cannot use more than one pad, even if more space is needed and none is available in the current scratch pad. No scratch file is ever associated with a particular pad.

Dynamically allocating scratch pads The simplest way to allocate scratch pads dynamically is to let SYSTEM 2000 software allocate one scratch pad (S2KPAD00) dynamically. SYSTEM 2000 software goes through the following steps, in the order given, when determining how to dynamically allocate a scratch pad:

1. If the scratch pads were allocated in the JCL or CLIST, the software uses them.

2. If no files were allocated in the JCL or CLIST, the software looks for permanent data sets with DSN=*prefix*.S2KPAD*nn*,DISP=OLD. If files exist, the software uses them.

3. If the scratch pad is already allocated with the PAD*nn* execution parameter, the software uses that space information specified for the pad.

4. If PAD*nn*=YES, the software uses the values specified for the PADPRI, PADSEC, PADUNIT, and PADSPACE execution parameters to determine the space for allocation. These execution parameters are not used if a PAD*nn* execution parameter exists for the file.

5. If you have not given any space allocations for any scratch pads, the software dynamically allocates one scratch pad as follows:

```
S2KPAD00 UNIT(SYSDA) SPACE(1,1) CYLINDER
```

   This default scratch pad is a temporary file.

When you dynamically allocate scratch pads, you can alter the default primary space, secondary space, unit device, and space unit with the following execution parameters:

Parameter	Default	Minimum	Maximum	Description
PADPRI=s	1	1	32767	Scratch pad primary space
PADSEC=s	1	0	32767	Scratch pad secondary space
PADUNIT=x	SYSDA	-	-	Scratch pad unit device
PADSPACE=y	CYL	(either TRACK or CYL)		Scratch pad space unit

Thus, the default is UNIT=(SYSDA), SPACE(1,1), CYL. The default DDname is *prefix*.S2KPAD00, where *prefix* equals the value for the PREFIX execution parameter if it was specified. For single-user jobs, the TSO user prefix is used if the PREFIX parameter does not exist. For more information about the PREFIX execution parameter, see **PREFIX execution parameter** on page 4-17.

Dynamic allocation of scratch pads does not create any permanent data sets; the data sets are temporary files. If the software dynamically allocates a scratch pad, it dynamically deallocates the file at the end of the single-user job or the Multi-User session.

Allocating scratch pads with the PADnn execution parameter If you decide to explicitly allocate scratch pads, read this entire discussion of the PAD*nn* execution parameter. You must define at least one scratch pad and no more than 16 scratch pads. Each scratch pad

can reside in either a temporary or a permanent data set with a DDname of S2KPAD*nn*, where *nn* ranges from 00 to 15.

The block size for all scratch pads is determined by using the pool which has both the smallest buffer size and S, B, or BE usage. For example, if POOL0=4592/14/S, the block size for all scratch pads is 4592. See **POOLn Parameter for Allocating I/O Buffer Pools** on page 4-24 for details about I/O buffer pools.

SYSTEM 2000 software maintains scratch pad space based on allocation units. That is, when scratch file space is needed during command processing, the software assigns one allocation unit of space at a time. Use the PAD*nn* execution parameter to specify the number and size of the allocation units in scratch pad S2KPAD*nn*.

Therefore, in the PAD*nn* parameter for a scratch pad you specify both the size of the allocation unit (in terms of number of blocks) and the total number of allocation units in the pad. The amount of space to be used in a scratch pad is the product of the number of allocation units times the number of blocks per allocation unit. This amount cannot exceed the amount of space specified for the data set. If the two amounts match, no data set space is wasted.

The next two topics describe how to specify the PAD*nn* parameter and how to define the data set for the scratch pad. The last two subsections give some examples and usage considerations.

PAD*nn* execution parameter The syntax of the parameter is shown here. **Setting SYSTEM 2000 Execution Parameters** on page 5-9 gives details about how to specify execution parameters.

```
PADnn= |number-of-allocation-units / blocks-per-allocation-unit
       |YES
       |NO
```

nn — is the pad number. The range is 00-15, that is, PAD00, PAD01, and so on. PAD*nn* must be defined consecutively, starting with PAD00.

number-of-allocation-units — is the total number of allocation units to be used for the pad. The minimum is 2, and the maximum is determined by the number of blocks that fit on a particular device.

blocks-per-allocation-unit — is the number of blocks in the allocation unit for the pad. As mentioned above, the block size is determined from the buffer size of pools used for work file pages.

YES — means that SYSTEM 2000 software calculates the number of allocation units and the number of blocks per allocation unit based on the size of the allocated file.

NO — means that the specified scratch pad parameter and any other subsequent scratch pad parameters that have higher pad numbers (*nn*) will be ignored.

Note: 32,760 is the limit number for both the allocation units and blocks per allocation unit.

By specifying PAD*nn*=NO, you can allocate only the scratch pads needed for a particular job. You do not have to delete PAD*nn* parameters or S2KPAD*nn* DDnames. PAD*nn* numbers must still be defined consecutively, starting with PAD00.

When defining data sets for scratch pads, consider the following guidelines:

- Scratch pads are BDAM files; therefore, they cannot be allocated to tape.
- You can allocate scratch pads to different disk devices to eliminate disk and channel contention.
- If you do not specify a particular PAD*nn* execution parameter but a corresponding S2KPAD*nn* DDname exists, that PAD*nn* parameter defaults to YES. **Note**: If an earlier PAD*nn*=NO exists, the PAD*nn* defaulting to YES will not be used.
- If you do not specify a particular PAD*nn* execution parameter and there is no corresponding S2KPAD*nn* DDname, that PAD*nn* parameter defaults to NO. **Note**: If PAD00 is NO, scratch files are used, not scratch pads.
- If PAD*nn*=YES, the number of allocation units is calculated on total available space, primary and fifteen secondary extents. The number of blocks per allocation unit is calculated on pages per track. If the number of allocation units is less than the number of blocks per allocation unit, the values will be transposed.
- A calculated PAD*nn* parameter cannot be allocated to multiple volumes. To have a multi-volume scratch pad, you must explicitly state the PAD*nn* parameters. (Calculations are not used.) The system assumes the scratch pad is capable of containing the amount of data specified in the PAD*nn* parameter. When you specify a multi-volume scratch pad, all 16 extents are formatted on all volumes except the last volume, on which only the first extent is formatted.
  - The data set specified can be temporary or permanent. A permanent scratch pad decreases initialization time for SYSTEM 2000 software. The initialization routine assigns scratch pad block sizes and formats the files. At initialization time, if scratch pads exist with proper block size and space and if they have been formatted, they are used without re-initialization. If you use an existing scratch pad that has either not been formatted or not been formatted for the current scratch file block size, the scratch pad file is reformatted. For temporary scratch pads, SYSTEM 2000 software always initializes the files. The presence or absence of a control record on each scratch pad and of the DISP field in the DD statement governs the initialization process.
  - The SPACE specification can be determined by using one of the following formulas. In these formulas, AU = *allocation-unit*.

    cylinders = AUs x (*blocks-per-AU*)/(*blocks-per-cylinder*[2])

    tracks = AUs x (*blocks-per-AU*)/(*blocks-per-track*[2])

[2] To determine the number of blocks per track, consult the documentation for your storage device.

- When additional allocation units are requested during processing, SYSTEM 2000 software acts as described below.

  - In a Multi-User environment, the software checks the count of active users. If the user needing the allocation unit is the only one active and no more extents are available or if the pad of space is already equal to the number of allocation units, the user is terminated with SYSTEM 2000 Error Code 75. If other users are active and the requesting user does not have all available allocation units, the user is suspended in the thread as waiting for resources. If a deadly embrace situation occurs, the user is terminated with SYSTEM 2000 Error Code 47.

  - For a single-user job, if there are no available allocation units and no secondary extents to be used, your job terminates with SYSTEM 2000 Error Code 75.

    Note that SYSTEM 2000 software extends the scratch pad if (1) there is a secondary quantity specified on the SPACE parameter, or (2) the space to be acquired does not exceed the control blocks[3] (Allocation Unit Directory Blocks) built at system initialization time.

  - The software formats secondary extents as needed. However, as discussed before, the space to be used in the scratch pad (specified in PAD*nn*) cannot exceed the space available in the data set. For single-user jobs, the primary extent must be less than the secondary extent in order to prevent I/O from being used to format physical pages that may never be used. Be careful when using secondary extents, because space abends (B37) can occur.

Coordinating pool and pad use Any time you change the buffer size specification for the pool that determines the block size of all work files, you are, in effect, also changing the scratch pad block size. This can require changes to the PAD*nn* specification, and, if the scratch pad is a permanent data set, SYSTEM 2000 software may have to reformat the data set.

Scratch pad space validation The software checks the SPACE parameter for scratch pads specified in your JCL to see whether enough space is allocated for each scratch pad specification in the S2KPARMS data set. The check for sufficient space is performed each time the SPACE parameter is encountered. (Releases prior to R11.6 checked the space only once, when the first SPACE parameter for each DDname was encountered in the JCL.)

Suppose the first step in your job contains SPACE (TRK,(30,150)) for PAD00 and the PAD00 execution parameter specifies PAD00=300/1. Also, assume that the block size for the scratch pad is 12992 and that the scratch pad is allocated on a 3350 disk. Specifying SPACE(TRK,(30,150)) allows 30 primary tracks and 2250 tracks (15 times 150) from secondary extents for a total of 2280 available tracks. The PAD00=300/1 requests 300 tracks because one block per track accommodates a block size of 12992. SYSTEM 2000 software finds that 2280 tracks is more than enough to satisfy the requested 300 tracks.

[3] SYSTEM 2000 software builds a 4-byte control block for each allocation unit. For example, if 1000 allocation units are specified, then 1000 4-byte blocks are built at initialization time, occupying 4000 bytes of memory.

Further, suppose that in job's second step, PAD00 is reused but the space changes to SPACE=(TRK,(30,0)). This could happen, for example, if you use the S2K procedure provided on the installation tape because the SPACE parameter provided there overrides the SPACE parameter in Step 1. Therefore, in Step 2, SYSTEM 2000 software finds that PAD00 space is 30 tracks of primary space and no secondary space. In this example, the check fails because 30 tracks is not enough to satisfy the 300 tracks requested by the PAD00=300/1 execution parameter. The user receives an abend.

Then Step 3 executes SYSTEM 2000 software and uses the same JCL as Step 1 and the same PAD00 data set. The result shows you that 2280 tracks are available.

Examples: user-specified scratch pads Examples 1 through 3 illustrate how you can specify scratch pads and space for them. (See Examples 4 through 8 for sample usage of the YES and No options.)

Example 1

Assume:

```
PAD00=90/5
POOL0=2016/5/D
POOL1=4060/12/S
UNIT=3380
```

The scratch pad block size is 4060 because POOL1 has the smallest buffer size of the pools with S, B, or BE usage.

In a 3380 device, there are ten 4060-byte blocks per track.

PAD00 is set to 90/5, so the scratch pad will use 450 (90 x 5) blocks or 45 tracks (using the formula for SPACE requirements: 90 x 5 / 10).

Three possible ways to define a DD statement for the scratch pad are

```
S2KPAD00 DD DSN=SCRPAD0,UNIT=3380,SPACE=(TRK,(45))

S2KPAD00 DD DSN=SCRPAD0,UNIT=3380,SPACE=(4060,(450))

S2KPAD00 DD DSN=SCRPAD0,UNIT=3380,SPACE=(TRK,(15,2))
```

In these cases the entire data set can be used by the scratch pad.

Example 2

Assume:

```
PAD00=1000/5
PAD01=5000/1
PAD02=1500/10
```

Assume that the smallest POOL*n* parameter for work files is 4060. Also assume that the space parameter on the DD statement specified only a primary allocation.

PAD00 has a total of 1000 allocation units with five physical blocks per unit. A total of 5000 4060-byte blocks are initialized. If the device is 3380, the space requirement is 500 tracks. (4060 is 1/10 track and 5000/10 = 500.)

PAD01 has a total of 5000 allocation units with one physical block per unit. In this case, PAD01 requires the same amount of space as PAD00 (5000 * 1 = 5000 4060-byte blocks).

PAD02 has 1500 allocation units with 10 physical blocks per unit. A total of 15000 4060-byte blocks is initialized. If the device is 3380, the space requirement is 1500 tracks.

Memory requirements for the pads are as follows:

PAD00 - 1000 4-byte control blocks, approximately 4K
PAD01 - 5000 4-byte control blocks, approximately 20K
PAD02 - 1500 4-byte control blocks, approximately 6K.

Example 3

Assume the device type is 3380, the block size is 4060 (1/10 track), and PAD00=100/6. SYSTEM 2000 software only formats a total of 600 blocks (100 x 6) or 60 tracks. The following space parameters show the initialization flexibility:

SPACE=(TRK,60) - All 600 blocks are written at system initialization.

SPACE=(TRK,(15,3)) - 150 blocks are written at initialization, and 30 blocks are written for each secondary extent.

SPACE=(TRK,(20,3)) - 300 blocks are written at initialization, and 30 blocks are written for each secondary extent. In this case only 10 extents are used, because only 600 blocks are needed.

Examples: calculated scratch pads Examples 4 through 8 illustrate some uses of the YES and NO options for the PAD*nn* execution parameter. Assume that these examples use the JCL shown here, with slight variations to illustrate different situations.

```
//SUSCF1 JOB
//SUSCF   EXEC   PGM=SYS2K,PARM='PAD00=YES,PAD02=570/5'
       .
       .
//S2KPAD00 DD  SPACE=(CYL,(5,1)),UNIT=SYSDA
//S2KPAD01 DD  SPACE=(CYL,(10,1)),UNIT=SYSDA
//S2KPAD02 DD  SPACE=(CYL,(9,1)),UNIT=SYSDA
//S2KPAD03 DD  SPACE=(CYL,(1,1)),UNIT=SYSDA
//S2KPAD04 DD  SPACE=(CYL,(1,1)),UNIT=SYSDA
       .
       .
//S2KPARMS DD  *
POOL0=6216/20/B
PAD01=570/2
PAD03=570/2
STAE=NO
XBUF=NO
       .
       .
DISK=3380
```

Example 4

This example shows how PAD04 defaults to YES because a DDname exists for PAD04 and no PAD04 parameter is specified. PAD00 is specified as YES.

```
PGM=SYS2K,PARM='PAD00=YES,PAD02=570/5'
```

```
$HASP601 SUSCF1 ...
      .
      .
      .
+PAD00=YES
+PAD01=570/2
+PAD02=570/5
+PAD03=570/2
+PAD04=YES
      .
      .
      .
+S2K0143/01- DEFAULT PAD PARM, PAD00=0300/0007 -
+S2K0143/01- DEFAULT PAD PARM, PAD04=0240/0007 -
+S2K0131/01- S2KPAD00 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD01 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD02 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD03 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD04 NOW BEING FORMATTED -
```

Example 5

This example illustrates ignoring a scratch pad and subsequent scratch pads. For this job, PAD03=NO means ignore scratch pad 03 and 04.

```
PGM=SYS2K,PARM='PAD00=YES,PAD02=570/5,PAD03=NO'
```

```
$HASP601 SUSCF1 ...
      .
      .
      .
+PAD00=YES
+PAD01=570/2
+PAD02=570/5
+PAD03=NO
      .
      .
      .
+S2K0143/01- DEFAULT PAD PARM, PAD00=0300/0007 -
+S2K0131/01- S2KPAD00 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD01 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD02 NOW BEING FORMATTED -
```

Example 6

In this example, PAD01 and PAD03 are specified in the S2KPARMS data set. PAD00, PAD02, and PAD04 default to YES because they have DDnames but no PAD*nn* parameters are specified.

```
PGM=SYS2K
```

```
$HASP601 SUSCF1 ...
      .
      .
      .
+PAD00=YES
+PAD01=570/2
+PAD02=YES
+PAD03=570/2
+PAD04=YES
      .
      .
      .
+S2K0143/01- DEFAULT PAD PARM, PAD00=0300/0007 -
+S2K0143/01- DEFAULT PAD PARM, PAD02=0360/0007 -
+S2K0143/01- DEFAULT PAD PARM, PAD04=0240/0007 -
+S2K0131/01- S2KPAD00 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD01 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD02 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD03 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD04 NOW BEING FORMATTED -
```

Example 7

In this example, the DD statement for PAD00 specifies multiple volumes and PAD00=YES is specified in the basic JCL. Fatal error message S2K0144/01 appears, because a default pad parameter cannot have multiple volumes.

```
//S2KPAD00 DD . . . VOL=SER=(SAS510,SAS508)
```

```
$HASP601 SUSCF1 ...
      .
      .
      .
+PAD00=YES
+PAD01=570/2
+PAD02=570/5
+PAD03=570/2
+PAD04=YES
      .
      .
      .
+S2K0144/01- MULTIPLE VOLUMES - DEFAULT PAD PARM, PAD00, NOT ALLOWED
```

Example 8

In this example, PAD03 in the S2KPARMS data set requests more tracks than are available, so fatal error message S2K0142/01 appears. The PGM=SYS2K statement does not specify any parameters, but DD names are specified for PAD00, PAD02, and PAD04; therefore, scratch pads 00, 02, and 04 default to YES.

```
PGM=SYS2K
      .
      .
      .
//S2KPAD03 DD  SPACE=(TRK,(9,4)),UNIT=SYSDA
```

```
$HASP601 SUSCF1 ...
      .
      .
      .
+PAD00=YES
+PAD01=570/2
+PAD02=YES
+PAD03=570/2
+PAD04=YES
      .
      .
      .
+S2K0143/01- DEFAULT PAD PARM, PAD00=0300/0007 -
+S2K0143/01- DEFAULT PAD PARM, PAD02=0360/0007 -
+S2K0143/01- DEFAULT PAD PARM, PAD04=0240/0007 -
+S2K0131/01- S2KPAD00 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD01 NOW BEING FORMATTED -
+S2K0131/01- S2KPAD02 NOW BEING FORMATTED -
+S2K0136/01- PAD03 HAS SPACE INCONSISTENCY -
+S2K0142/01- REQUESTED TRACKS 0285 > AVAILABLE TRACKS 0069 -
```

PADCALC Utility Program

PADCALC is a stand-alone utility program that displays how much space will be required for scratch pads. The program looks at the S2KPARMS file, the specified execution parameters, and the S2KPAD*nn* DD statements.

By executing the PADCALC program, you can see the defaults that will be used prior to running SYSTEM 2000 software with default scratch pad allocations. The program estimates the amount of space required for the various PAD*nn* specifications on specific DASD volumes. PADCALC looks for POOL*n* and DISK statements in order to arrive at the proper default block size for scratch pads.

After PADCALC determines the default scratch pad block size, the program evaluates each scratch pad for which a DD statement exists. Then it displays the information shown below on the SYSTEM 2000 Message File.

Sample PADCALC output Here is a sample report created by the PADCALC program.

```
PAD   DEFAULT    PRIM SECND TOTAL  BLKS  SPECIFIED    REQUIRED  TRACK ALLOC
NUM X-VAL Y-VAL  TRKS  TRKS  TRKS P/TRK X-VAL Y-VAL BLKS   TRKS AVAIL ERROR

 00    69     8    9     4     9     8   570     5  2850   357    65 *YES*
 01  2070     8  270   120   270     8                          2070
 02    18     8    3     1     3     8                            18
 03    69    18    9     4     9    18                            69
 04   975    18  135    60   135    18   300    40 12000   667   975
 05    18    16    1     1     1    18                            16
 06    69    18    9     4     9    18                            69
 07  1035    18  135    60   135    18                          1035
 08    18    16    1     1     1    18                            16
 09 MULTI-VOL-NO DEFAULTS ALLOWED    18    72     8   576    32
 10    18     8    3     1    18     8                            18

S2KPAD BLOCK SIZE WILL BE   2016
```

The information displayed in the various columns is described below.

PAD NUM	is the scratch pad number.
DEFAULT X-VAL	is the number of allocation units that will be the default if PAD*nn*=YES or if an S2KPAD*nn* DD statement exists. (X-VAL is the first value for the PAD*nn*=x/y parameter.)
DEFAULT Y-VAL	is the number of blocks per allocation unit that will be the default if PAD*nn*=YES or if an S2KPAD*nn* DD statement exists. (Y-VAL is the second value for the PAD*nn*=x/y parameter.)
PRIM TRKS	is the number of tracks requested for the primary allocation of the data set if the allocation is new. If the allocation is not new, PRIM TRKS is the number of tracks in the first extent of the data set.

SECND TRKS — is the amount of space requested for each secondary extent of the data set after the primary allocation has been satisfied.

TOTAL TRKS — is the total number of tracks currently allocated to this data set, not including any unused extents for the data set.

Note: For VIO data sets, TOTAL TRKS equals 15 times the secondary allocation plus the primary allocation. JES always computes the maximum amount of space required (primary plus 15 secondaries) and assigns one VIO with that much space.

BLKS P/TRK — is the number of scratch pad records (blocks) that will fit on a track of the device allocated to the scratch pad.

SPECIFIED X-VAL — is the number of allocation units specified by the user in a PAD*nn* parameter if present in either the S2KPARMS data set or the execution parameters. (X-VAL is the first value in the PAD*nn*=*x*/*y* parameter.)

SPECIFIED Y-VAL — is the number of blocks per allocation unit specified by the user in a PAD*nn* parameter if present in either the S2KPARMS data set or the execution parameters. (Y-VAL is the second value in the PAD*nn*=*x*/*y* parameter.)

REQUIRED BLKS — is the total number of blocks required to satisfy the user's PAD*nn* specification. This number is displayed only when SPECIFIED X-VAL and SPECIFIED Y-VAL are present, that is, the number is X times Y.

REQUIRED TRKS — is the total number of tracks required to satisfy the user's PAD*nn* specification. This number is displayed only when SPECIFIED X-VAL and SPECIFIED Y-VAL are present.

TRACK AVAIL — is the total number of tracks that can be obtained in one primary and up to fifteen secondary extents. **Note**: VIO data sets are allocated with one extent of space equivalent to the primary space request plus fifteen secondary extents.

ALLOC ERROR — is the word *YES* if you specify a PAD*nn* parameter that will not have enough available space. That is, REQUIRED TRKS is greater than TRACK AVAIL.

The PADCALC program and the SYSTEM 2000 routine for calculating default pad parameters do not consider multi-volume files. For multi-volume files, you must specify a PAD*nn* parameter in the S2KPARMS data set or in the execution parameters. SYSTEM 2000 software considers any properly specified PAD*nn* parameter as having the required space available. You must ensure that enough space exists to fulfill the PAD*nn* requirements for multi-volume scratch pads. SYSTEM 2000 software formats the space on all volumes until the first extent on the last volume is formatted.

When you specify a multi-volume scratch pad, PADCALC displays your specified parameters, the number of blocks per track for the determined block size and device type, and the number of blocks and tracks required to fulfill the PAD*nn* parameter. The message MULTI-VOL-NO DEFAULTS ALLOWED appears where the actual values for the data set would normally appear.

JCL for executing PADCALC The sample JCL below illustrates how to run the PADCALC program in batch mode.

```
//PADCALC  JOB
//STEP01  EXEC  PGM=PADCALC,PARM='PAD04=300/40,PAD09=72/8'
//STEPLIB  DD   DSN=S2K.R120.LOAD,DISP=SHR
//S2KMSG   DD   SYSOUT=$
//S2KPARMS DD *
PAD00=570/5
/*
//S2KPAD00 DD UNIT=SYSDA,VOL=SER=SAS504,SPACE=(TRK,(9,4))
//S2KPAD01 DD UNIT=SYSDA,VOL=SER=SAS504,SPACE=(CYL,(9,4))
//S2KPAD02 DD UNIT=SYSDA,VOL=SER=SAS504,SPACE=(4000,(9,4))
//S2KPAD03 DD UNIT=SYSDA,VOL=SER=SAS803,SPACE=(TRK,(9,4))
//S2KPAD04 DD UNIT=SYSDA,VOL=SER=SAS803,SPACE=(CYL,(9,4))
//S2KPAD05 DD UNIT=SYSDA,VOL=SER=SAS803,SPACE=(4000,(9,4))
//S2KPAD06 DD DSN=S2K.R120.PAD6,DISP=OLD
//S2KPAD07 DD DSN=S2K.R120.PAD7,DISP=OLD
//S2KPAD08 DD DSN=S2K.R120.PAD8,DISP=OLD
//S2KPAD09 DD UNIT=SYSDA,VOL=SER=(123456,125567),SPACE=(TRK,(1,1))
//S2KPAD10 DD UNIT=VIO,SPACE=(4000,(9,4))
```

Allocating Scratch Files Individually

As mentioned in **Work Files: Overview** on page 4-37, SYSTEM 2000 software requires that you use scratch pads to allocate scratch file space. Although scratch pads simplify scratch file allocation, there may be times when you need to have the scratch files on tape. In those cases, you must allocate scratch files individually, because you cannot allocate scratch pads to tape. Note, however, that scratch pads are required to execute SYSTEM 2000 software in a Multi-User environment. They are also required in a single-user environment when the REPORT processor or COLLECT command is used.

If you allocate scratch files individually, you also have to allocate Locate Files to use the LOCATE command in PLEX. You can allocate up to 15 Locate Files with DDnames LOCATE01 to LOCATE15.

When allocating scratch files individually, consider the following guidelines:

- SYSTEM 2000 software determines the page size of all scratch files at job initialization time by using the smallest buffer size specified in any pool with usage of S, B, or BE.
- Scratch files can be allocated to tape or disk, but you must allocate at least one scratch file to disk. It can be a scratch file that will not be used in a session.
- All scratch files allocated to disk must be allocated to the same type of disk device. Mixing device types among scratch files causes unpredictable results.
- The scratch file DDnames are: S2KSYS01, S2KSYS02, S2KSYS03, S2KSYS04, S2KSYS05, S2KSYS06, and S2KSYS07.

LOAD and RELOAD are examples of two procedures that may require scratch files on tape.

The scratch file requirements for an SCF or PLEX load operation can be considerable. If you are planning to do extensive loads, you can estimate the size requirements for scratch files 2, 5, 6, and 7 using the instructions provided in the next section.

Scratch File Sizes During Loading

Before you can estimate the scratch file sizes for a load operation, you need a complete database definition and a realistic knowledge of the data to be loaded. Then, use this list of scratch files to estimate the files sizes needed for your data.

PLEX Load Mode	SCF	Contents
5	2	contains one record for each good value of a key item. The file is sorted into Index order.
	5	contains one record for each data record defined in the loader stream and one record for each value included in the data record. In general, the file can contain records from logical entries that had errors at scan time. These records are later dropped. (Not used for PLEX load mode.)
	6	first Data Table pass; contains one record for each good value in the loader stream. (Not used for PLEX load mode.)
6	6	second Data Table pass; contains one record for each long value of a key item; that is, a value that exceeds the picture size (overflow). The file is sorted into Data Table order.
	7	contains one record for every continuous series of "good" records on Scratch File 5. A "good" record from Scratch File 5 is one that is not part of a logical entry that has been rejected. For instance, if no errors were detected during the scan, then Scratch File 7 has one record. (Not used for PLEX load mode.)

DATABASE DEFINITION BLOCK SIZES

During execution, SYSTEM 2000 software keeps in memory a Database Definition Block for each open database. The software keeps these blocks in two separate pools: one for small database definition (SDBS) blocks and another for large Database Definition Blocks (LDBS).

Four execution parameters determine how these two pools are used: LDBSIZE, LDBS, SDBSIZE, and SDBS. In SDBS and LDBS, you specify how many small and large Database Definition Blocks should be available. In SDBSIZE and LDBSIZE, you specify the maximum number of components that can be defined for a small database and for a large database. SYSTEM 2000 software uses these values to calculate the sizes of large and small Database Definition Blocks.

To determine whether a database is to be considered large or small, SYSTEM 2000 software uses the values of SDBSIZE, LDBSIZE, and MXCOMP. MXCOMP is the maximum number of components that can be defined for a database as specified in the CONTROL language NEW DATA BASE IS command.

NEW DATA BASE IS *database-name*[*/maxcomp*]

maxcomp is 430 by default.

See the *SYSTEM 2000 CONTROL Language, Version 12, First Edition* for details about the NEW DATA BASE IS command.

When a database is opened, SYSTEM 2000 software assigns its definition block to one of the two pools according to the following rules:

- If *maxcomp* $\leq$ SDBSIZE, the block goes into the SDBS pool.
- If SDBSIZE $<$ *maxcomp* $\leq$ LDBSIZE, the block goes into the LDBS pool.
- If *maxcomp* $>$ SDBSIZE and *maxcomp* $>$ LDBSIZE, the system issues an error message and the database is not opened.

Note that SYSTEM 2000 software does not check whether LDBSIZE is greater than SDBSIZE.

Determining Large Database Definition Blocks: LDBS and LDBSIZE

To determine the size of the LDBS pool, use the values specified for the LDBS and LDBSIZE execution parameters. See **Setting SYSTEM 2000 Execution Parameters** on page 5-9 for details about how to specify execution parameters.

LDBS=*blocks*

blocks is the initial number of Database Definition Blocks to be acquired at run time for large databases. That is, it specifies the number of large databases that SYSTEM 2000 software can have open at one time without having to do a GETMAIN to acquire memory for additional definition blocks. The range is 0-32. The default is 0.

If the number of definition blocks required for large databases exceeds the parameter specification and the GETMAIN fails, an appropriate message is issued and the database is not opened for use.

LDBSIZE=*comp*

comp is the maximum number of components that a large database can contain. The range is 0-10000. The default is 1000.

SYSTEM 2000 software uses the value for LDBSIZE to determine the amount of memory to allocate for each large Database Definition Block.

The size of an LDBS pool, in bytes, is given by

blocks x *block-size*

where

blocks is the LDBS value.

block-size is

Multi-User: 3240 + (18 x *comp*) + (USERS x 8)

single user: 3200 + (18 x *comp*)

USERS is the value of the USERS parameter (see **USERS Parameter** on page 3-64).

Note: The software uses the value of LDBSIZE to determine the number of buffers required for the SYSTEM 2000 session, where (LDBSIZE x 4) equals the required number of bytes. If one or two buffers (from any pool with any usage) are adequate, the buffer manager uses that pool. If more than two buffers are needed, the buffer manager acquires them from the pool with the largest buffers that were specified for database usage (D, DE, or B). If the required number of buffers for LDBSIZE is not available when SYSTEM 2000 software is invoked, WTO message 0116 is issued, followed by user abend 101.

Determining Small Database Definition Blocks: SDBS and SDBSIZE

To determine the size of the SDBS pool, use the values specified for the SDBS and SDBSIZE execution parameters. See **Setting SYSTEM 2000 Execution Parameters** on page 5-9 for details about how to specify execution parameters.

SDBS=*blocks*

blocks is the initial number of Database Definition Blocks to be acquired at run time for small databases. That is, it specifies the number of small databases that SYSTEM 2000 software can have open at one time without having to do a GETMAIN to acquire memory for additional definition blocks. The range is 0-32. The default is 1.

If the number of definition blocks required for small databases exceeds the parameter specification and the GETMAIN fails, an appropriate message is issued and the database is not opened for use.

SDBSIZE=*comp*

comp is the maximum number of components that a small database can contain. The range is 1-10000. The default is 430.

SYSTEM 2000 software uses the value for SDBSIZE to determine the amount of memory to allocate for each small Database Definition Block.

The size of an SDBS pool, in bytes, is given by

blocks **x** *block-size*

where

blocks is the SDBS value.

block-size is

Multi-User: 3240 + (18 x *comp*) + (USERS x 8)

single user: 3200 + (18 x *comp*)

USERS is the value of the USERS parameter (see **USERS Parameter** on page 3-64).

GATHERING TIMING STATISTICS: QAEXIT AND QASTAT

The QAEXIT module (QASTAT for PLEX programs) allows you to gather CPU timing for commands and I/O counts for individual files. This information is different than the timing data for overlays provided by the TIMING ON command.

To gather the timing statistics in an SCF job, give the system-wide QAEXIT command. For a PLEX program, issue a call to QASTAT. In either case, you must have a specially linked system.

When gathering these timing statistics, consider the following guidelines:

- You can use QAEXIT and QASTAT in either single-user or Multi-User environments.
- You can monitor up to 256 files for I/O counts in an SCF session or a PLEX program. If this maximum is exceeded, a message is issued during the first call for timing, and all subsequent calls produce I/O counts for the first 256 files (DD statements).
- The execution time for producing the timing statistics is not included in any of the CPU totals.
- In some SYSTEM 2000 environments, I/O activity is not reported on STEPLIB. When reported, I/O activity shown for STEPLIB indicates I/Os that were required for overlay swapping. For a single overlay swap, the IBM operating system can use as many as 40 to 60 I/Os.

QAEXIT/QASTAT Output

The format and type of information produced by QAEXIT and QASTAT are identical. The first request for timing initializes the timer and then displays a set of brief comments explaining the content of various fields in the output. These comments do not appear for subsequent timing requests in the same job. Each timing request is assigned a unique identification number, which is displayed in the output.

CUR is the CPU time used since the last call to QAEXIT or QASTAT.

SEQ is the total CPU time for a specific sequence of PLEX commands. The sequence is specified as an optional parameter in the PLEX call to QASTAT (see **Timing for a PLEX Job: QASTAT** on page 4-60). If you want several consecutive calls totaled together, you can code the same sequence number on all the applicable timing calls. Each time the sequence number changes (or is not specified), the SEQ total is reset. If you do not specify any sequence numbers in your timing requests, then SEQ equals TOT.

The optional sequence parameter is not available in the QAEXIT command. Therefore, SEQ equals TOT for all SCF jobs.

TOT is the total CPU time used since the first call to QAEXIT or QASTAT.

OVD is the CPU overhead time used by the QAEXIT module while it gathered and reported the timing information. The OVD time is not included in the CUR, TOT, or SEQ totals.

Note: CPU time is reported to the nearest thousandth of a second.

The subsequent lines of output show I/O counts for the first 256 files specified in the JCL DD statements. Two counts appear to the left of each file name (DDname). If blanks precede the file name, the file has not been accessed.

- The first count is the number of I/Os since the last QAEXIT command or PLEX call to QASTAT.
- The second count is the total number of I/Os since the start of the job step.
- The file names are displayed in three columns with as many rows as needed. The display fits on 80-character output devices.
- The order of the displayed file names depends on the order of the DD statements in the JCL. All files are displayed, including those for DD DUMMY statements and concatenated data sets. A concatenated data set has the same DDname as the file to which it belongs; it is not blank.

Illus. 4.5 on page 4-59 shows a sample display of timing statistics produced by giving QAEXIT commands. The timing and I/O counts are intermixed with the echoes of SCF commands on the Message File. If you want to suppress the echoes of your commands, use the ECHO OFF command.

The output from calling QASTAT in a PLEX program is similar except that it reflects PLEX commands that were executed and is written to the TIMEIO file (or the FT09F001 file).

QAEXIT/QASTAT Messages and Codes

For specific information relating to any SCF messages or PLEX return codes encountered while executing QAEXIT or QASTAT, see *SYSTEM 2000 Messages and Codes, Version 12, First Edition*.

Illus. 4.5 Format of QAEXIT and QASTAT Output

```
 05/03/91  16:14:47  BEGIN SYSTEM 2000 - RELEASE xx.x
---
USER,ABC: QAEXIT:
---TIMING AND I/O STATISTICS BEGIN---85.192---16:14:4829----------------------
    I/O IS LISTED BY DDNAME ACROSS PAGE WITH COUNTS THEN DDNAME
    1ST FIGURE IS EXCPS SINCE LAST CALL, 2ND FIGURE IS TOTAL EXCPS
    TIME IS LISTED WITH THE FOLLOWING MEANING:
    A = ELAPSED TIME SINCE LAST CALL UNLESS CALL NUMBER HAS BEEN CHANGED
    B = ELAPSED TIME IN THIS CALL NUMBER SEQUENCE
    C = ELAPSED TIME SINCE 1ST CALL (BEGIN)
    D = OVERHEAD FOR TOTAL RUN    NOTE: OVERHEAD IS NOT INCLUDED IN A, B OR C
--------------------CALL # 000/  A   -------- 16:14:482-----------------------
 TIME -   CUR=00000.000    SEQ=00000.000    TOT=00000.000    OVD=00000.000
 I/O  -      88    88 STEPLIB                  STEPLIB                  SYS00866
                      SF01                     SF02                     SF03
                      SF04                     SF05                     SF06
                      S2KSYS01                 S2KSYS02                 S2KSYS03
                      S2KSYS04                 S2KSYS05                 S2KSYS06
                      S2KSYS07                 S2KPAD00                 S2KMSG
                      S2KSNAP                  SYSUDUMP                 S2KCOMD
                      TESTXXX1                 TESTXXX2                 TESTXXX3
                      TESTXXX4                 TESTXXX5                 TESTXXX6

---
NDB IS TEST: ECHO OFF:
 -558- CREATED....TEST                  0      0  05/03/1991  16:15:44
---
MAP: QAEXIT:
--------------------CALL # 000/  B   -------- 16:15:519-----------------------
 TIME -   CUR=00004.532    SEQ=00004.532    TOT=00004.532    OVD=00000.006
 I/O  -      97   185 STEPLIB                 STEPLIB                  SYS00866
                      SF01                     SF02                     SF03
                      SF04                     SF05                     SF06
                      S2KSYS01                 S2KSYS02                 S2KSYS03
                      S2KSYS04                 S2KSYS05       9     9 S2KSYS06
                      S2KSYS07                 S2KPAD00                 S2KMSG
                      S2KSNAP                  SYSUDUMP                 S2KCOMD
             227   227 TESTXXX1       211   211 TESTXXX2       212   212 TESTXXX3
             211   211 TESTXXX4       211   211 TESTXXX5       211   211 TESTXXX6

---
IT C0 * 0 EQ 1*A*2* 1*10*11*AX1*12* 2*20*21*AY1*22* 3*END*:
 -342-       1 SELECTED RECORD(S) -
 ---
QAEXIT:
--------------------CALL # 000/  C   -------- 16:15:520-----------------------
 TIME -   CUR=00000.055    SEQ=00004.587    TOT=00004.587    OVD=00000.010
 I/O  -          185 STEPLIB                 STEPLIB                  SYS00866
                      SF01                     SF02                     SF03
                      SF04                     SF05                     SF06
                      S2KSYS01                 S2KSYS02                 S2KSYS03
                      S2KSYS04                 S2KSYS05       9     9 S2KSYS06
                      S2KSYS07                 S2KPAD00                 S2KMSG
                      S2KSNAP                  SYSUDUMP                 S2KCOMD
               1    228 TESTXXX1           211 TESTXXX2           212 TESTXXX3
                    211 TESTXXX4           211 TESTXXX5           211 TESTXXX6

 ---
 EXIT:
   16:15:52  05/03/91   END  SYSTEM 2000 - RELEASE xx.x
```

Timing for the Self-Contained Facility: QAEXIT Command

To gather timing statistics and I/O counts during an SCF job, give the following command:

```
QAEXIT:
```

QAEXIT is a system-wide command. Therefore, you can use it with any of the SYSTEM 2000 SCF processors. Specify the QAEXIT command each time you want to display the detailed statistics. You will probably want to use it repeatedly during an SCF session, for example, before or after each retrieval or update command.

The first QAEXIT command displays the I/Os accumulated for each file since the beginning of the session. However, because the first call serves to initialize the timer, all timing information is zero for this call. Subsequent QAEXIT calls provide timing information from this initial point.

For details about the content and format for displaying the statistics, see **QAEXIT/QASTAT Output** on page 4-57. Each time you give the QAEXIT command, SYSTEM 2000 software writes the statistics to the Message File for your individual SCF job.

Consider the following guidelines when using QAEXIT:

- When you use the QAEXIT command in a Multi-User environment, SYSTEM 2000 software disables the CPU timing statistics to prevent conflict with the Multi-User Accounting Log. A message is issued to that effect, and only I/O counts are displayed.

- Both the TIMING ON command and the QAEXIT command can be used in an SCF job; however, this is not recommended. The scratch file I/O counts could differ, because the QAEXIT output includes I/O to and from merge files during a sort. Also, in some environments, CPU time might be reported incorrectly.

Linking QAEXIT with SYSTEM 2000 software To request timing information for SCF jobs in either a single-user or a Multi-User environment, you must use a specially linked copy of SYSTEM 2000 software. The QAEXIT module on the load library must be linked with the root segment of the SYSTEM 2000 executable load module. Otherwise, the QAEXIT command is ignored. To link the QAEXIT module with SYSTEM 2000 software, specify the QA= parameter in the PRELNK macro. This means you want to include the QAEXIT module in the installation configuration of the software. For details about the QA= parameter, see **The PRELNK Macro Instruction** on page 7-9.

The QAEXIT module has two external references, S2KCVT and COMENT. These references are resolved when you include and link the QAEXIT module with SYSTEM 2000 software.

The QAEXIT module requires an additional 6244 bytes of memory for an SCF job.

Timing for a PLEX Job: QASTAT

For a PLEX program, issue a call to QASTAT each time you want to display the timing statistics. The QASTAT module must be linked into each PLEX program that calls QASTAT.

You can call the QASTAT module in any PLEX program written in COBOL, FORTRAN, PL/I, or Assembler. Also, because QASTAT is independent of SYSTEM 2000 software, any program written in one of these languages can use the QASTAT utility by linking it into the program. This allows you to gather I/O and timing information for programs that are not PLEX programs.

Use the following formats to call QASTAT for the respective programming languages:

PLEX Language	CALL Syntax
FORTRAN	CALL QASTAT [(*var*)]
COBOL	CALL 'QASTAT' [(*var*)].
PL/I	CALL QASTAT [(*var*)];
Assembler	CALL QASTAT [,(*var*)]

where *var* is a fullword binary number. This number is optional. It affects the CPU time displayed for SEQ (see **QAEXIT/QASTAT Output** on page 4-57). If you want the CPU time and the I/O counts totaled for several consecutive calls to QASTAT, code the same sequence number in all the applicable calls. Each time the sequence number changes (or is not specified), the SEQ total is reset. If you do not specify any sequence numbers in your timing requests, then SEQ equals TOT. TOT is the total CPU time used since the first call to QASTAT.

Output file QASTAT output is written to the TIMEIO file, not the Message File. To be compatible with earlier releases of QASTAT, the FT09F001 file is also a valid output file. If you do not include one of the following DD statements, QAEXIT issues a WTO message, and the routine is effectively disabled. This does not cause your program to abend.

```
//TIMEIO DD SYSOUT=A
//FT09F001 DD SYSOUT=A
```

The format and content of the statistics written on this file are the same as for QAEXIT. The output file is restarted from the beginning each time a PLEX job is executed and terminated.

Note: QASTAT opens and writes the output file. However, because it does not know which call is the last one, it does not issue a CLOSE for the output file. Instead, it expects the operating system to close it at the end of the step.

Linking QASTAT into a PLEX program You can link the QASTAT utility with any COBOL, FORTRAN, PL/I, or Assembly language program.

The QASTAT module is delivered with the SYSTEM 2000 load library. You must include QASTAT with your program via the linkage-editor INCLUDE statement. If you call QASTAT and it is not included, there will be an unresolved reference and an IBM abend will occur during execution of your program.

When you link QASTAT into your PLEX program, two references, S2KCVT and COMENT, are unresolved, but this does not cause any problem during the execution of your program.

Note: The QASTAT module requires an additional 6244 bytes of memory for the PLEX job.

PROTOTYPING DATABASES

Frequently, a prototype database containing few data is established to test an application. While a prototype is a useful way to test a schema and file allocation, it is not merely a scaled-down version of a production database. For example, size differences can result in relatively less I/O for a prototype than for a large database. With a small amount of data, the entire database can be read into memory, page by page, as required. No I/O is done beyond a single "read" per database page and perhaps a "write" if updates are successful. If the record needed is already in memory, no I/O is needed to retrieve it from disk. However, for a large database, only a few database pages are retained in memory. Therefore, a page needed for a certain request is unlikely to be in memory as a result of a previous request.

Actual performance statistics for a production database provide more reliable information for reevaluating space allocation, buffer pool configuration, and block sizes. Gather this information when the database has stabilized, and use it to adjust JCL specifications when you recreate the database.

Also, when a prototype database contains only token data, few disk tracks are required to test an application. Each file requires at least one full track of disk space, since the operating system does not assign partial tracks. Although you can define a database in as few as ten tracks of 3380 disk space, Files 1 and 3, which store the definition, can themselves require more than five tracks, depending on the maximum number of components allowed for the database. (File 1 size can be minimized by limiting the number of components.) You cannot change the maximum definable number of components (MXCOMP), like block size, without scratching and recreating the database. This is acceptable during the early test phase when only a small amount of data has been loaded and block size is not too important. However, before you define the production database, you need to study the applications and assign appropriate block sizes to the database files.

The Distinct Values Table (DVT) uses at least one page for each valued key item, even if there are only a few values. Each item in a test database is likely to occur, with few occurrences of each. For forty key items, forty pages of DVT space is probably required, even though most of each DVT page remains unused. Remember that small blocks use less space. If you specify BLKSIZE=2016 for the DVT of this database, 18 blocks fit on a 3380 track, and three tracks of 3380 space hold the file. For BLKSIZE=12992, three blocks fit on a 3380 track, and 14 tracks of 3380 space are required, most of it unused.

One track of any standard disk for each file is enough for Files 3 through 6 to hold a small amount of test data. Note that one track is the minimum that can be allocated to a file.

During testing, you need to maintain a Command File of DEFINE declarations for the entire schema. If the database is damaged, you can recover it by scratching the old database, then reallocating, redefining, and reloading it. For a small amount of data, this is a reasonable alternative to the SAVE/RESTORE method.

Chapter 5

Selecting and Setting SYSTEM 2000 Options

SYSTEM 2000 software has options for a wide variety of tasks, such as setting up buffers for database file and scratch file I/O, gathering Multi-User statistics, collecting job activity and database activity data, requesting Coordinated Recovery, using the REPORT processor, setting various SCF and PLEX options, and so forth. Each option has a default setting. However, the defaults might not satisfy requirements of your site environment.

To set the options for a Multi-User session or for a single-user job, give appropriate specifications for the SYSTEM 2000 execution parameters when the session or job is initialized. Some parameter settings can also be changed dynamically by the console operator during a Multi-User session, for example, to reset the priority of user jobs.

Summary of Execution Parameters on page 5-3 summarizes the SYSTEM 2000 execution parameters available. Use this table to become familiar with the many components that affect the SYSTEM 2000 processing environment. The table also includes the range of values and default for each parameter, as well as the page on which the parameter is discussed.

Setting SYSTEM 2000 Execution Parameters on page 5-9 shows you four methods of setting the execution parameters. The section gives general formats and rules, along with examples of syntax. It also shows the priorities the software uses to establish parameter settings, depending on the method(s) you choose for setting the values.

You can set up an S2KPARMS file containing the parameter settings, and this file can be dynamically allocated when used in subsequent job sessions.

The remaining sections in this chapter describe the general-purpose execution parameters not discussed elsewhere in this manual. These parameters can be used in single-user jobs or for a Multi-User session, for example, to display parameter settings, to convert databases, to enable error trapping, to enable the REPORT processor, and to change the default settings used by many of the SCF and PLEX commands.

SUMMARY OF EXECUTION PARAMETERS

This table summarizes the SYSTEM 2000 execution parameters. It shows the ranges of values, the defaults, which parameters can be used in a single-user environment, and the pages that contain the details about the parameters.

Illus. 5.1 Summary of Execution Parameters

Name and Description	Range	Default	SU	Page
ALLOC=*option* allows dynamic allocation of database files.	YES, NO, TBL	YES	yes	4-16
ACCT=*option* activates/deactivates the Accounting Log.	YES, NO	NO		3-110
CONVERT=*option* allows conversion of databases to new releases; ignored for Version 12 conversions.	YES, NO	NO	yes	5-15
COPYAREAn=*sizeK/m* assigns an interregion copy area for systems requiring the S2KCOPY routine in the Multi-User SVC.	*n*=1-6 *size*=1-65 *m*=1-4096	*n*=1 *size*=1 *m*=4		3-54
CORECOV=*option* determines whether only the primary or all databases are recovered by Coordinated Recovery.	YES, NO	NO	yes	5-15
DBBUFN=*n* alters the number of QSAM buffers for database files during save/restore processing.	*n*=1-255	*n*=5	yes	5-13
DISK=*device* determines default block size for database files and the default buffer size for POOL0.	2314,3310,3330, 3340,3344,3350, 3370,3375,3380, 3390	3380	yes	4-21
DITTO=*option* controls the use of the DITTO operator across SCF TP segments.	YES, NO	YES		3-57
EXITS=*option* controls the overall execution of all user exits.	YES, NO	NO	yes	6-13

continued on next page

Illus. 5.1 *continued*

Name and Description	Range	Default	SU	Page
FREE=*option* controls deallocation of database files.	YES, NO	NO	yes	4-17
EXT reserved for future use.				
FPTPSYS=*option* determines whether system areas are to be obtained for fetch-protected systems to allow TP access from different address spaces.	YES, NO	YES		3-57
LDBS=*blocks* specifies the initial number of Database Definition Blocks to be used for large databases.	*blocks*=0-32	0	yes	4-55
LDBSIZE=*n* specifies the maximum number of components that can be defined in a large database.	*n*=0-9999	9999	yes	4-55
LHOLD=*option* controls the use of the PLEX /HOLD option.	YES, NO	YES		3-58
LIST=*option* controls the display of the execution parameter settings.	YES, NO	YES	yes	5-13
LOGCOUNT=*n* specifies the maximum number of records that can be written to the Diagnostic Log.	*n*=1-32767 (in thousands)	1 (=1000)		3-133
LOGLEVEL=*option*[/*option*]... specifies what types of messages are to be written to the Diagnostic Log.	LOPEN, PDATE, POPEN, TSPIO, TSPND, TSTRT, UINIT, USEGM, USPND, NO	NO		3-133
MAN reserved for future use.				
NLSEG=*hsec*/*nio*[/DELAY] controls what SCF batch job data, if any, are to be written to the Accounting Log.	*hsec*=NO, 1-999999 *nio*=NO, 1-999999	NO NO		3-110
OPI=*option* determines whether the console operator can override or specify execution parameter settings.	YES, NO	NO		3-58

continued on next page

Illus. 5.1 *continued*

Name and Description	Range	Default	SU	Page
OPT000=*option* checks the system separator when unloading CHAR, TEXT, and UNDEFINED values.	YES, NO	NO	yes	5-17
OPT001=*option* determines whether SCF commands are echoed (ECHO OFF/ON).	YES, NO	NO	yes	5-18
OPT002=*option* determines whether you can save a damaged database.	YES, NO	NO	yes	5-18
OPT003=*option* disables or enables clearing for QUEUE sessions.	YES, NO	NO	yes	5-18
OPT004=*option* includes system-related time in Type 5, 6, and 8 Accounting Log records.	YES, NO	NO		3-112
OPT005=*option* sets the default for the ZERO/ZERO SUPPRESS format option.	YES, NO	NO	yes	5-19
OPT006=*option* sets the default for the NULL/NULL SUPPRESS format option.	YES, NO	NO	yes	5-19
OPT007=*option* sets the default for the REPEAT/REPEAT SUPPRESS format option.	YES, NO	NO	yes	5-19
OPT008=*option* sets the default for the NAME/NUMBER format option.	YES, NO	NO	yes	5-19
OPT009=*option* determines whether database Files 2 through 6 are cleared for RELOAD and RELEASE.	YES, NO	NO	yes	5-20
OPT010=*n* sets a limit on selected records for updates and retrievals.	*n* = 0 to 32767	0,0	yes	5-20
OPT011=*n* specifies the maximum number of pages to be written to the primary scratch file for where-clause processing.	*n* = 0 to 4095	0	yes	5-20
OPT012=*n* sets the default block size for database files.	*n* = 2492 to 32767	6216	yes	5-20

continued on next page

Illus. 5.1 *continued*

Name and Description	Range	Default	SU	Page
OPT013=*n* changes the default date format. A default of 0 means mm/dd/yyyy.	*n* = 0 to 5	0	yes	5-21
OPT014 through OPT029 are reserved for future use.				
OPT030=*option* disables clearing of PLEX stacks and Locate Files when LUW is committed.	YES, NO	NO	yes	5-21
OPT031=*option* places the number of user calls in Type 4 Accounting Log records.	YES, NO	NO		3-112
OPT032=*option* sets the TALLY command option to ALL or EACH. NO means EACH.	YES, NO	NO	yes	5-21
OPT033=*option* sets the mode for clearing updated pages.	YES, NO	NO	yes	5-21
OPT034=*option* allows QueX software to update a record without requiring a hold.	YES, NO	NO		3-58
OPT035=*option* determines the type of display from the LIST execution parameter.	YES, NO, ALL	NO	yes	5-22
OPT036=*option* places the step and program names in Type 6 Accounting Log records.	YES, NO	NO		3-112
OPT037=*option* places the synchpoint ID for an LUW in the ACCTALT field of the user's URB.	YES, NO	NO		3-112
OPT038= is reserved for future use.				
OPT039=*option* affects floating point precision in PLEX data if padding is needed.	YES, NO, OFF	OFF	yes	5-23
OPT040=*option* affects floating point precision in PLEX data if truncation is needed.	YES, NO, OFF	OFF	yes	5-23
OPT041=*option* disables or enables the CMS FINIS command for all users.	YES, NO	NO	yes	5-24

continued on next page

Illus. 5.1 *continued*

Name and Description	Range	Default	SU	Page
OPT042=*option* disables the setting of condition codes for the COND parameter on the JCL execute statement.	YES, NO	NO	SU only	2-8
OPT043=*option* disables uppercase translation.	YES, NO	NO	yes	5-24
OPT044=*option* enables or disables the processor prompt feature.	YES, NO	NO	yes	5-24
OPT045=*option* enables or disables the processor lookup feature.	YES, NO	NO	yes	5-24
OPT046=*option* sets the default key/non-key status when defining new items in a definition. NO means KEY.	YES, NO	NO	yes	5-25
OPT047=*option* disables validity checking of user-specified packed decimal data.	YES, NO	NO	yes	5-25
OPT048 through OPT999 are reserved for future use.				
PAD*nn*=*x*/*y* |YES |NO defines one or more scratch pads or asks software to assign them.	*nn*=00-15 *x*=allocation units *y*=blocks per unit	See Text	yes	4-41
PADPRI=*n* alters scratch pad primary space.	*n*=1-32767	1	yes	4-41
PADSEC=*n* alters scratch pad secondary space.	*n*=0-32767	1	yes	4-41
PADSPACE=*option* alters SZKPAD space unit.	CYL, TRK	CYL	yes	4-41
PADUNIT=*unit* specifies SZKPAD UNIT value.	See Text	SYSDA	yes	4-41
PLSEG=*hsec*/*nio*[/DELAY] controls what PLEX job data, if any, is written to the Accounting Log.	*hsec*=NO, 1-999999 *nio*=NO, 1-999999	NO NO		3-110
POOL*n*=*bufsz*/*nbufs*[/*usage*) specifies the buffer size, number of buffers, and use of the buffer pools for database files and for scratch files.	See Text	See Text	yes	4-24

continued on next page

Illus. 5.1 *continued*

Name and Description	Range	Default	SU	Page
PQA=FIFO or PQA=PRTY[/*n*/*y*] determines the priority queuing algorithm.	*n*=0-999 *y*=0-225	FIFO 0/0		3-59
PREFIX=*prefix* specifies the default user-prefix, up to 35 chars.	See Text	See Text	yes	4-17
RW=*option* determines whether users can access the REPORT processor.	YES, NO	NO	yes	5-13
SAME=*option* controls the use of the SAME operator across SCF TP segments.	YES, NO	YES		3-61
SDBS=*blocks* specifies the initial number of DataBase Definition Blocks to be used for small databases.	*blocks*=0-32	1	yes	4-56
SDBSIZE=*n* specifies the maximum number of components that can be defined in a small database.	*n*=1-9999	430	yes	4-56
SFPRI=*n* alters sort file primary space.	*n*=1-32767	1	yes	4-38
SFSEC=*n* alters sort file secondary space.	*n*=0-32767	1	yes	4-38
SFSPACE=*option* specifies the sort file space unit.	CYL/TRK	CYL	yes	4-38
SFUNIT=*unit* specifies the sort file UNIT value.	See Text	SYSDA	yes	4-38
SID=*n* establishes the System ID.	*n*=0-99	1		3-61
STAE=*option* specifies how error trapping is to be handled.	YES, NO, NODUMP	YES	yes	5-14
STARTTP=*option* initiates the SCF TP facility.	YES, NO	YES		3-61
THREADS=*n* specifies the number of threads that can be used simultaneously.	*n*=1-63	1		3-62

continued on next page

Illus. 5.1 *continued*

Name and Description	Range	Default	SU	Page
TPSCRUN=*n* specifies the maximum number of active and inactive SCF TP users allowed.	*n*=1-10000	180		3-62
TPSEG=*hsec*/*nio*[/DELAY] controls what SCF TP job data, if any, are to be written to the Accounting Log.	*hsec*=NO, 1-999999 *nio*=NO, 1-999999	NO NO		3-110
TPTHREADS=*n* specifies the number of SCF TP users that can be queued.	*n*=0-230	5		3-63
TSO=*option* controls whether Multi-User can interface with TSO.	YES, NO	NO		3-63
USERS=*n* specifies the maximum number of batch SCF, batch PLEX, and PLEX TP users that can be executing concurrently.	*n*=0-230	16		3-64
XBUF=*option* enables the XBUF caching feature.	YES, NO, STAT	NO	yes	5-16
XBUFSUF=*suffix* identifies a specific XBUF load module if several exist on the load library.	*suffix*=one character		yes	5-16

SETTING SYSTEM 2000 EXECUTION PARAMETERS

You can set the SYSTEM 2000 execution parameters with the following methods:

- operator replies - if allowed by the Multi-User OPI execution parameter
- EXEC statement parameters
- S2KPARMS data set records
- SYSTEM 2000 defaults.

This list shows the priority of acquiring parameter values. In a Multi-User environment, the console operator can change parameter values if the OPI parameter equals YES. (See **OPI Parameter** on page 3-58.) Console operator values take precedence over any previous values.

The EXEC statement overrides S2KPARMS data set values. If the S2KPARMS data set is not present or fails to open, default values are used for all parameters not set in the EXEC statement and not overridden by the operator. SYSTEM 2000 default values are lowest priority and remain in effect if a specific value is not specified by one of the other methods.

Formats

This discussion shows the formats for setting the parameters with the JCL EXEC statement and with the S2KPARMS data set. See **Multi-User Console Operator Commands** on page 3-65 for details about changing some parameters with operator console commands.

Format for the JCL EXEC statement The format for setting SYSTEM 2000 execution parameters in the PARM parameter of a JCL EXEC statement is

parameter=value[*/value*] ... [,*parameter=value*[*/value*]

parameter the keyword name of the parameter.

value an alphabetic or numeric value.

Format for the S2KPARMS data set The syntax for setting SYSTEM 2000 execution parameters with the S2KPARMS data set is

parameter=value[*/value*]...[*bcomment*]

parameter the keyword name of the parameter.

value an alphabetic or numeric value.

bcomment comment text. After specifying a parameter and its value(s), you can give an optional comment; a blank separates the comment from the actual parameter specifications. All characters following the blank become part of the comment.

The S2KPARMS data set consists of 80-byte records readable by QSAM. The records can be blocked or unblocked. If the S2KPARMS data set fails to open, a non-fatal, informative message is issued.

Rules for setting execution parameters

- Each execution parameter must start in Column 1, which means that only one parameter can be given per line.
- Embedded blanks are not allowed in any parameter specification. Also, a blank line causes a user abend code, and the session ends.
- If a parameter is specified twice, the last value specified overrides the previous value. Only parameter values from the highest priority are effective during execution.
- With the COBOL/VS compiler, you can pass run-time parameters to the COBOL debugging option. These parameters are preceded by a slash (/) in the EXEC statement PARM field and must be the rightmost parameters. However, this can conflict with SYSTEM 2000 parameters being passed to COBOL PLEX programs.

For example, a COBOL program that is passed the PAD00=570/5 parameter on the EXEC statement gets a user abend 101. With the COBOL/VS compiler the '/5' is treated as a run-time debug option, and only PAD00=570 is passed to SYSTEM 2000 software. To avoid this problem, either specify SYSTEM 2000 parameters in the S2KPARMS data set or suffix the SYSTEM 2000 parameters with a COBOL/VS run time parameter, for example, PAD00=570/5/STATE=NO.

Dynamically Allocating the S2KPARMS File

You can let SYSTEM 2000 software dynamically allocate the S2KPARMS file, or you can allocate it in your JCL or CLIST. The software performs the following steps:

1. If the S2KPARMS file is already allocated, the software uses that file, and dynamic allocation does not take place.

2. The software looks for a permanent S2KPARMS data set. If it exists and can be allocated with DISP=SHR, the software uses it.

   The *prefix* for the data set name is the value of the PREFIX parameter in the EXEC statement if it exists. For TSO single-user jobs, the TSO user prefix is used if the PREFIX parameter was not specified. You can easily change the TSO prefix in the PROFILE with the PROFILE command before entering SYSTEM 2000 software.

Note: For the Multi-User initialization step and for batch jobs, you must specify the PREFIX execution parameter in the PARMS= portion of the EXEC statement so that the software can find the S2KPARMS file.

Examples of Setting Execution Parameters

Example 1

This example illustrates using the JCL EXEC statement to specify the POOL0 parameter and to request the REPORT processor:

```
//GO  EXEC  S2K, PARM='POOL0=6440/14/B,RW=YES'
```

Example 2

This example shows the JCL for specifying S2KPARMS as an in-line statement data set:

```
//S2KPARMS  DD  *
POOL0=6440/14/B
RW=YES
/*
```

Example 3

This example shows the JCL for specifying S2KPARMS as a member of a partitioned data set (PDS):

```
//S2KPARMS  DD  DSN=SYS1.PARMLIB(S2KPARMS),DISP=SHR
```

Example 4

This example shows the JCL for specifying S2KPARMS as a sequential data set:

```
//S2KPARMS  DD  DSN=S2K.EXEC.PARMS,DISP=SHR
```

Example 5

In this example, dynamic allocation occurs if you have an existing data set with the name 'S2K.S2KPARMS'.

```
//GO   EXEC S2K,PARM='PREFIX=S2K'
```

LIST PARAMETER: DISPLAYING THE EXECUTION PARAMETERS

For single-user and Multi-User environments, the LIST execution parameter enables or disables a display of SYSTEM 2000 execution parameter settings.

```
LIST=|YES     (default)
     |NO
```

YES (the default) produces a listing of parameter values specified in the S2KPARMS file and of default values that apply to the session. Only the first 21 characters of each parameter setting are displayed. The listing is written to both the job log and the operator console.

LIST=NO produces an abbreviated list showing only numeric parameters where the default is non-zero. A setting of NO is ignored if the software finds an unrecognizable parameter. Processing continues as though LIST=YES.

In general, a parameter is displayed only when it is required in the given environment. For example, a listing for a single-user job does not show the Multi-User parameter values.

A special zap is available that suppresses all operator warning messages pertaining to execution parameters. The zap suppresses those messages regardless of the LIST parameter specification or the occurrence of an unrecognizable parameter. See the Early Warning System listing for a description of the zap.

RW PARAMETER: ENABLING THE REPORT PROCESSOR

For single-user and Multi-User environments, the RW execution parameter determines whether users can access the REPORT processor.

```
RW=|NO     (default)
   |YES
```

YES makes the REPORT processor available. The work area for an SCF job requires 52K bytes (9K more than usual). In a Multi-User environment, each Thread Work Area requires 52K bytes. (See also THREADS parameter in **THREADS Parameter** on page 3-62.) NO (the default) means the REPORT processor is not available, and 9K fewer bytes are needed per work area.

DBBUFN PARAMETER: ALTERING QSAM BUFFERS FOR SAVE/RESTORE

The DBBUFN execution parameter enables you to alter the number of QSAM buffers to be used for database files while saving and restoring databases.

```
DBBUFN=n
```

The default for *n* is 5. The allowable range of values is 1 to 255. These QSAM buffers are acquired for the duration of the save or restore process only, then released. To control the QSAM buffers for the Savefile, use the BUFNO option on the DD statement.

STAE PARAMETER: ENABLING ERROR TRAPPING

The STAE execution parameter specifies how error trapping should be handled. STAE=YES enables error trapping.

```
STAE=|YES       (default)
     |NO
     |NODUMP
```

For single-user SCF and PLEX jobs with STAE=YES (the default), SYSTEM 2000 software formats the Extended Specify Task Abnormal Exit (ESTAE) snapshot when an unrecoverable error occurs. The formatted ESTAE work area appears on the user's Message File (S2KMSG). A full snapshot is written to the S2KSNAP file.

If a user job abends under Multi-User, the ESTAE work area is not formatted but is provided as the first snapshot, followed by a full snapshot. Both snapshots are written to the S2KSNAP file. For a Multi-User abend, the formatted ESTAE work area is written to the Diagnostic Log.

The ID of the full snapshot is 50 for SYSTEM 2000 Error Code 820, or 51 for SYSTEM 2000 Error Code 826. If an MVS abend has not occurred, but SYSTEM 2000 software detects an unrecoverable error, then some SYSTEM 2000 Error Code (not 820 or 826) is issued. A snapshot may or may not be issued. If a snapshot is issued, its ID is 60.

The DD name is hard-coded in the snapshot data control block as S2KSNAP. Some operating systems require that BLKSIZE=882 be specified in the DCB parameter of the S2KSNAP DD statement in the JCL. All operating systems permit such a specification.

One of the following SYSTEM 2000 Error Codes is issued if an attempt to take a snapshot fails: 821, 823, 825, 827, 828, 829, 830.

If STAE=NODUMP, ESTAEs are in effect during SYSTEM 2000 execution, but no dump is taken if an abend occurs. STAE=NODUMP is especially good for COBOL II and PL/I PLEX programs that utilize debugging aids and, therefore, do not need a SYSTEM 2000 dump. Normally, you would not set STAE=NODUMP for production Multi-User jobs; that is, for Multi-User abends, you would want to take a dump.

If STAE=NO, SYSTEM 2000 error trapping does not occur and unpredictable results can follow, for example, damaged databases, Multi-User software brought down, and so forth. Never use STAE=NO when running in a production environment. STAE=NO can be used in a test environment when developing and debugging programs against test databases. If an abend occurs in such a test environment, any dumps resulting from a user or system abend are sent to the SYSUDUMP or SYSABEND files. STAE=NO does not disable any ESTAE trapping done for dependent region PLEX sessions in a Multi-User environment.

For more information about ESTAEs and abends, see **Abends in Single-User Jobs** on page 2-9 and **Abends in the Multi-User Environment** on page 3-46.

Note: SYSTEM 2000 software will dynamically allocate an S2KSNAP file if necessary and if it is not already allocated in the JCL or CLIST. The file default values are SYSOUT=A and HOLD.

CORECOV PARAMETER: ENABLING COORDINATED RECOVERY

The CORECOV execution parameter determines the scope of Coordinated Recovery in single-user and Multi-User environments. The main discussion of Coordinated Recovery is in *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

```
CORECOV=|NO          (default)
        |YES
```

The CORECOV parameter is effective only when you are attempting to open a database that was damaged by a system failure in a previous session. The Rollback Log must have been active before the failure. This parameter is not used for recovering databases damaged by transaction failures, job cancellation, and so forth.

The database that is requested for opening (and that needs recovery) is called the primary database. Secondary databases are those that were updated and involved in logical units of work to be used for recovering the primary database. These databases may or may not be available during recovery of the primary database.

If CORECOV=NO (default), SYSTEM 2000 software recovers only the primary database and includes committed updates from all users of the database. The system does not check availability of secondary database files or their Update Logs.

If CORECOV=YES, SYSTEM 2000 software attempts a full Coordinated Recovery on the secondary databases as well as the primary database. This recovery includes all user transactions on all databases in the logical units of work. An error message is issued if the appropriate secondary databases, Update Logs, and Rollback Logs are not available for the full recovery of all databases.

CONVERT PARAMETER: CONVERTING DATABASES

CONVERT pertains only to earlier versions of SYSTEM 2000 software. It does not apply to converting databases to Version 12. Databases must be reconstructed for Version 12, which provides larger database files. You can use the CONVRT12 conversion program described in Appendix B. Or you can load your database using SCF commands or PLEX load mode. If you do not have existing PLEX programs that unload and load your databases, see Appendix B.

XBUF PARAMETER: ENABLING THE XBUF CACHING FEATURE

The XBUF execution parameter enables or disables XBUF caching software in a single-user job or in a Multi-User environment under MVS.

XBUF=|NO *(default)*
|YES
|STAT

YES means the XBUF software is activated and real caching takes place. To activate XBUF software, you must specify either XBUF=YES or XBUF=STAT. The delivery tape contains a default XBUFTBL module, which you can use to try XBUF=STAT. However, for XBUF=YES you must allocate real files, and you must code one or more cache macros. The XBUF execution parameter affects all cache areas that you have defined. However, when you code your XBUF macros, you will find they offer many options for designating individual cache areas and specific database files for real or simulated caching. For details, see *Technical Report: S2-107 XBUF Caching Feature in SYSTEM 2000 Software, Release 11.6 under IBM OS*.

STAT means the XBUF software is activated but only for simulated caching. The STAT option does not acquire cache space; it turns on the XBUF I/O monitor so you can display caching statistics with XBUF console commands. STAT allows you to model cache areas without performing any actual I/Os.

NO (the default) means the XBUF software is not activated.

XBUFSUF PARAMETER: IDENTIFYING A SPECIFIC XBUF LOAD MODULE

The XBUFSUF execution parameter identifies (with a suffix) a specific XBUF load module if more than one exists in the load library. You do not need the suffix if you generate only one XBUF load module.

XBUFSUF=*suffix*

suffix is a single character appended to the XBUFTBL load module name. The suffix is any national character that is valid for a PDS member name.

When you assemble and link-edit a set of XBUF macros, the resulting load module name is XBUFTBL. Optionally, you can create several configurations of XBUFTBL, each represented by a different module in the load library. In this situation, the load module names must be in the format of XBUFTBL[*suffix*], where *suffix* is an optional character that distinguishes one XBUFTBL configuration from another, for example, XBUFTBLA, XBUFTBL1, XBUFTBL2.

If several configurations of XBUFTBL exist, you must specify the appropriate suffix by giving the XBUFSUF execution parameter. The suffix indicates which configuration to activate. For example, to activate the load module named XBUFTBLA, specify XBUFSUF=A in the S2KPARMS file.

OPTNNN PARAMETERS

The OPTnnn execution parameters described in this section enable you to change many of the options used by SCF and PLEX commands. They can be set for Multi-User and single-user jobs. Each OPTnnn parameter has a default, which can be reset to satisfy requirements at your site. Many of these parameters replace special zaps that were offered prior to Release 11.6. See **Summary of Execution Parameters** on page 5-3 for an overview of other OPTnnn parameters that pertain either to Multi-User environments or to single-user jobs, but not to both.

OPT000 Parameter: Checking the System Separator

The OPT000 execution parameter determines how extensively SYSTEM 2000 software checks for the system separator when unloading CHARACTER, TEXT, or UNDEFINED item values.

```
OPT000=|NO      (default)
       |YES
       |OFF
```

NO (the default) means SYSTEM 2000 software checks for the current system separator only. Also, if the HEX ON format option is in effect, the check for the system separator is not performed for UNDEFINED items since those values are in HEX notation in the output.

OFF tells the software not to perform any checks for the system separator character in unloaded data. OFF saves time during the unload process since no checking is done, but users take the risk of unloading values that contain the current system separator. Problems will occur later if the software reads the unloaded data.

YES tells the software to check whether any characters that are allowed as the system separator occur in any of the CHARACTER, TEXT, or UNDEFINED values being unloaded. YES takes more time during the unload process.

UNDEFINED values might frequently contain system separator characters because all 256 EBCDIC characters are allowed. CHARACTER and TEXT values might contain the system separator characters because the current system separator is different from the character used when the data were loaded or because the data were loaded with a PLEX program, which does not involve the system separator.

If one or more values contain the current system separator, message -290- appears at the completion of the unload processing. YES produces message -291- after message -290-. The characters displayed in message -291- did not occur in any unloaded value and, therefore, can be used as the system separator during the unload. Lines 2 and 3 in message -291- show the hexadecimal values of the characters. You can safely change the current system separator to any of the displayed characters and execute the UNLOAD command again to produce acceptable UNLOAD output.

The following example assumes that OPT000 equals YES and the current system separator is the asterisk. Suppose a value being unloaded for C1 produces the following value stream:

```
1*AAA!<+*)"*
```

AAA is acceptable. The remaining characters in the value contain the asterisk and other characters that could create problems if they become the system separator. The following messages appear:

```
-290- DATA VALUE CONTAINS CURRENT SYSTEM SEPARATOR -
-291- THESE SYSTEM SEPARATORS ARE AVAILABLE ¢. ( |&  $  ; -/%_>?#@'= -
-291-                                      44 4 45  5  5 6666667777 -
-291-                                      AB D F0  B  E 01CDEFBCDE _
```

OPT001 Parameter: Setting the ECHO ON/OFF Default

The OPT001 execution parameter sets the default for echoing SCF commands.

```
OPT001=|NO      (default)
       |YES
```

YES sets ECHO OFF for your site; that is, SCF commands are not automatically echoed back to the user. The user must issue ECHO ON to see echoes during an SCF session; NO (the default) means ECHO ON. (OPT001 replaces Special Zap #002.)

OPT002 Parameter: Preventing Users from Saving Damaged Databases

The OPT002 execution parameter prevents users from saving damaged databases.

```
OPT002=|NO      (default)
       |YES
```

YES causes an 0C1 to occur if an attempt is made to save a damaged database. NO (the default) means that the database will be saved regardless of its condition. (OPT002 replaces Special Zap #011.)

OPT003 Parameter: Disabling Clears in a QUEUE Session

The OPT003 execution parameter disables clears in a QUEUE/TERMINATE session.

```
OPT003=|NO      (default)
       |YES
```

YES disables clears for a QUEUE/TERMINATE session in order to increase performance. NO (the default) enables clears. (OPT003 replaces Special Zap #034.)

OPT005 Parameter: Setting the ZERO/ZERO SUPPRESS Default

The OPT005 execution parameter sets the default for the ZERO/ZERO SUPPRESS format option, which is used in SCF commands.

```
OPT005=|NO      (default)
       |YES
```

YES specifies ZERO. NO (the default) means ZERO SUPPRESS. (OPT005 replaces Special Zap #103.)

OPT006 Parameter: Setting the NULL/NULL SUPPRESS Default

The OPT006 execution parameter sets the default for the NULL/NULL SUPPRESS format option, which is used in SCF commands.

```
OPT006=|NO      (default)
       |YES
```

YES specifies NULL. NO (the default) means NULL SUPPRESS. (OPT006 replaces Special Zap #109.)

OPT007 Parameter: Setting the REPEAT/REPEAT SUPPRESS Default

The OPT007 execution parameter sets the default for the REPEAT/REPEAT SUPPRESS format option, which is used in SCF commands.

```
OPT007=|NO      (default)
       |YES
```

YES specifies REPEAT SUPPRESS. NO (the default) means REPEAT. (OPT007 replaces Special Zap #110.)

OPT008 Parameter: Setting the NAME/NUMBER Default

The OPT008 execution parameter sets the default for the NAME/NUMBER format option, which is used in SCF commands.

```
OPT008=|NO      (default)
       |YES
```

YES specifies NAME. NO (the default) means NUMBER. (OPT008 replaces Special Zap #111.)

OPT009 Parameter: Clearing Files 2 through 6

The OPT009 execution parameter allows clearing of all database files for a RELOAD or a RELEASE command.

```
OPT009=|NO      (default)
       |YES
```

YES means that database Files 2 through 6 (in addition to File 1) will be cleared when the database is reloaded or released. NO (the default) means that only database File 1 will be cleared for RELOAD or RELEASE. (OPT009 replaces Special Zap #121.)

OPT010 Parameter: Setting a Limit on Selected Records

The OPT010 execution parameter sets a limit on the number of records that can be selected by a where-clause. This number determines the maximum number of records that can be updated or retrieved by an SCF command.

OPT010=*nnnnn*

OPT010 turns on the LIMIT command and sets the maximum to the value specified for your site. The value of *nnnnn* can be any number in the range of 0 to 32767. For example, if you specify 1200, an error message will appear when the number of selected records for any user's retrieval or update exceeds 1200.

If you do not specify OPT010, the default limits are 0,0, which means no limits are set. (OPT010 replaces Special Zap #003.)

OPT011 Parameter: Limiting the Where-Clause Scratch File Size

The OPT011 execution parameter indicates the maximum number of pages written to the primary scratch file for where-clause processing.

OPT011=*nnnn*

The value of *nnnn* is a number in the range of 0 through 4095. The default is 0, which means no limit. (OPT011 replaces Special Zap #020.)

OPT012 Parameter: Setting the Database File Block Size

The OPT012 execution parameter provides the default block size for preallocated database files with no block size and for dynamically allocated new database files, regardless of the DISK parameter setting. The value for OPT012 must be a valid block size and must not be greater than the largest block size specified for any pool, or error messages appear.

OPT012=*nnnnn*

The minimum value of *nnnnn* is 2492, and the maximum is 32767. (OPT012 replaces Special Zap #050.)

OPT013 Parameter: Changing the Default Date Format

The OPT013 execution parameter changes the default date format.

OPT013=*n*

The value *n* must be an integer between 0 and 5, which sets the following date formats:

0	mm/dd/yyyy (the default)
1	mm/dd/yy
2	dd/mm/yyyy
3	dd/mm/yy
4	yyyy/mm/dd
5	yy/mm/dd

(OPT013 replaces Special Zap #004.)

OPT030 Parameter: Clearing PLEX Stacks and Locate Files

The OPT030 execution parameter disables clearing of PLEX stacks and Locate Files.

```
OPT030=|NO      (default)
       |YES
```

YES disables clearing of all PLEX stacks and eviction of all Locate Files, which are normally performed when a logical unit of work (LUW) is committed. NO (the default) clears the PLEX stacks and Locate Files. (OPT030 replaces Special Zap #130.)

OPT032 Parameter: Setting the TALLY Option

The OPT030 execution parameter changes the EACH/ALL option for the TALLY command.

```
OPT032=|NO      (default)
       |YES
```

YES specifies ALL. NO (the default) means EACH. (OPT032 replaces Special Zap #113.)

OPT033 Parameter: Setting Clearing Mode

The OPT033 execution parameter alters the default mode for clearing updated pages at your site.

```
OPT033=|NO      (default)
       |YES
```

YES sets the mode to CLEAR AUTOMATICALLY, which clears the modified pages automatically for each update command. NO (the default) means modified pages are cleared only when the system needs a buffer or when the user issues a CLEAR command. (OPT033 replaces Special Zap #185.)

OPT035 Parameter: Setting the Type of Execution Parameter Listing

The OPT035 execution parameter affects the display of SYSTEM 2000 execution parameters.

```
OPT035=|NO      (default)
       |YES
       |ALL
```

OPT035 always works in conjunction with the LIST execution parameter. If LIST=YES, the value of OPT035 (or the default of NO) determines the type of listing that you will see. If LIST=NO, the execution parameters are not displayed, and OPT035 is ignored.

The first character in each display line indicates whether the value is a default or a specified value.

b	means a universal default. (The indicator is a blank.)
@	means the default is specific to your site.
*	means the value was changed by the S2KPARMS file.
=	means the value was changed by the EXEC statement.
-	means the value was changed by the operator (OPI=YES).

For example, +*RW=YES means that the RW execution parameter was changed to YES by the S2KPARMS file.

OPT035=NO and LIST=YES displays only the execution parameters that you have specified in the S2KPARMS file, that is, the parameters that were modified. Also, the display shows only those parameters that are relative either to the single-user job or to the Multi-User session. NO is the default.

OPT035=YES and LIST=YES displays all execution parameters relative to either the single-user job or to the Multi-User session. That is, you will see the default values as well as the modified values.

OPT035=ALL and LIST=YES displays all execution parameters. That is, the listing shows both single-user and Multi-User execution parameter settings and includes defaults as well as modified values.

OPT039 Parameter: Floating Point Precision Padding in PLEX Programs

The OPT039 execution parameter affects how you want floating point data precision to be handled in PLEX programs if padding is needed.

```
OPT039=|OFF     (default)
       |YES
       |NO
```

YES means you want data precision for single and double precision numbers. Therefore, specifying YES means that Return Code 17 will be issued if the subschema definition does not match the database definition.

NO means you are not concerned with data precision, but you want to see whether padding occurred. Return Code 98 is a warning that padding occurred when the software was transferring single and double precision numbers to and from the database. If both padding and truncation take place in the same subschema, Return Code 99 is issued instead of Return Code 98. (OPT040 determines the type of checking for truncation and issues Return Code 99.)

OFF (the default) means you are not concerned with data precision, and Return Code 98 will not be issued as a warning. Padding occurs if needed, and Return Code 00 is issued.

OPT040 Parameter: Floating Point Precision Truncation in PLEX Programs

The OPT040 execution parameter affects how you want floating point data precision to be handled in PLEX programs if truncation is needed.

```
OPT040=|OFF     (default)
       |YES
       |NO
```

YES means you want data precision for single and double precision numbers. Therefore, specifying YES means that Return Code 17 will be issued if the subschema definition does not match the database definition.

NO means you are not concerned with data precision, but you want to see whether truncation occurred. Return Code 99 is a warning that truncation occurred when the software was transferring single and double precision numbers to and from the database. If both truncation and padding take place in the same subschema, Return Code 99 is issued instead of Return Code 98. (OPT039 determines the type of checking for padding and issues Return Code 98.)

OFF (the default) means you are not concerned with data precision, and Return Code 99 will not be issued as a warning. Truncation occurs if needed, and Return Code 00 is issued.

OPT041 Parameter: Disabling the CMS FINIS Command

The OPT041 execution parameter determines whether the CMS FINIS command is disabled.

```
OPT041=|NO      (default)
       |YES
```

YES disables the CMS FINIS command for all users. NO (the default) means the FINIS command will be executed. (OPT041 replaces Special Zap #267.)

OPT043 Parameter: Disabling Uppercase Translation

The OPT043 execution parameter affects uppercase translation.

```
OPT043=|NO      (default)
       |YES
```

YES disables uppercase translation in single-user jobs and in the Multi-User TPI and TSO interfaces. For Multi-User, this option is specified on the dependent user side as a parameter in the JCL EXEC statement. NO (the default) enables uppercase translation. (OPT043 replaces Special Zap #188.)

OPT044 Parameter: Enabling the Processor Prompt Feature

The OPT044 execution parameter enables or disables the processor prompt feature.

```
OPT044=|NO      (default)
       |YES
```

NO (the default) means the system prompt is three dashes. YES means the system will display the current processor, for example,

```
ACCESS>
CONTROL>
DEFINE>
REPORT>
```

For more details, see *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition.*

OPT045 Parameter: Enabling the Processor Lookup Feature

The OPT045 execution parameter enables or disables the processor lookup feature.

```
OPT045=|NO      (default)
       |YES
```

YES means that if you are using the QUEST or CONTROL processor and you issue a command that is valid only in another processor, the software will attempt to transfer you to the appropriate processor. NO (the default) means that no lookup will be done, and a syntax error occurs. For more details, see *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition.*

OPT046 Parameter: Changing the Key/Non-Key Status of New Items

The OPT046 execution parameter changes the default key/non-key status of new items that the user is defining without specifying KEY or NON-KEY.

```
OPT046=|NO      (default)
       |YES
```

YES means the default key/non-key status is NON-KEY. NO (the default) means the default status is KEY. (OPT046 replaces Special Zap #277.)

OPT047 Parameter: Specifying Validity Checking for Packed Decimal Data

The OPT047 execution parameter disables or enables validity checking for user-specified packed decimal data.

```
OPT047=|NO      (default)
       |YES
```

YES means SYSTEM 2000 software will not validate user-specified packed decimal data. NO (the default) means validity checking will occur. (OPT047 replaces Special Zap #164.)

Chapter 6

User Exits

USER-EXITS: OVERVIEW

User exits allow you to do specialized user processing in conjunction with SYSTEM 2000 processing. User-provided routines can be executed against selected SYSTEM 2000 data areas during SYSTEM 2000 execution, at logical times that are predetermined within SYSTEM 2000 software.

User exits offer the following capabilities:

- enhanced or specialized security processing
- dynamic device allocation
- dynamic data value encoding
- creation of user-specific 'dialects' for the Self-Contained Facility
- PLEX data manipulation languages
- direct SYSTEM 2000 interface to site-developed software, such as editing and encryption routines
- SYSTEM 2000 interface with proprietary software packages, such as financial accounting, manufacturing, or statistical systems.

User-exit routines can be executed only at predefined logical exit points that reside in SYSTEM 2000 software. A user-exit interface routine, S2EXIT, is supplied to establish communication between the logical SYSTEM 2000 exits and the user-exit routines. The S2EXIT interface routine ensures the standardization and control of user-exit execution.

S2EXIT determines which SYSTEM 2000 logical exit is being taken and updates the exit parameter list accordingly. The exit parameter list is controlled by S2EXIT. It contains the addresses of the SYSTEM 2000 data areas that user-exit routines can interrogate or modify. Because of the dynamic nature of positions for the parameters, try to refer to the parameters in the list by labels rather than by displacement into the list. After S2EXIT accomplishes its tasks for a SYSTEM 2000 exit, control is given to the user-exit routine specified in the user-exit load list.

The user-exit load list is a control table of entry point addresses. The list is maintained by the user and specifies which user-exit routine is to be executed for the logical SYSTEM 2000 exit being taken. The S2EXIT interface routine calls the appropriate user-exit routine. The address of the exit parameter list is passed according to standard IBM calling conventions.

The user-exit routine can examine or modify those data areas available to it by obtaining their addresses from the exit parameter list. Before the user-exit routine returns control to S2EXIT, messages (if any) should be posted in the appropriate data area. Also, the action code must be set to indicate what action was taken and whether further SYSTEM 2000 processing is required. S2EXIT examines the action code in the user control block data area and takes the appropriate action.

Here is a summary of the available exits.

Illus. 6.1 Summary of Available Exits

Exit Number	Logical SYSTEM 2000 Exit Point	Possible Applications	Allowable Action Codes
00	During SYSTEM 2000 single-user or Multi-User initialization, if EXITS=YES is specified or if an external reference is resolved by link edit.	Calling S2KEXIN to allow exit to user-exit routine mapping.	0, 4
01	After parsing an SCF ENABLE/ DISABLE EXITS command or after a PLEX call resulting from an ENABLE/DISABLE EXITS command.	Modifying mapping of exit to user-exit routine.	0, 4, 8, 12, 40, 44, 48, 52
02	During SYSTEM 2000 single-user or Multi-User initialization (that is, first EXIT00, then EXIT02 if it was enabled by EXIT00).	Running initialization.	0, 4
03	Prior to physical opens for database files.	Dynamic allocation of database files, password security.	0, 4, 8, 12, 40, 44, 48, 52
04	After physical closes for database files.	Deallocating database files.	0, 4, 8, 12, 40, 44, 48, 52
05	After retrieval of any database page.	Decrypting database pages.	0, 4, 8, 12, 40, 44, 48, 52
06	Prior to writing any database page to disk.	Encrypting database pages.	0, 4, 8, 12, 40, 44, 48, 52
07	After physical opens of database files.	Updating allowable buffers per database file table.	0, 4, 8, 12, 40, 44, 48, 52
08	After scanning an SCF syntactic unit.	Security by command.	0, 4, 8, 12, 24, 36, 40, 44, 48, 52
09	Prior to tape mounts for keep/ apply Keepfile.	Dynamically allocating tape units.	0, 4, 8, 12, 40, 44, 48, 52
10	Prior to tape mounts for save/ restore operations.	Dynamically allocating tape units.	0, 4, 8, 12, 40, 44, 48, 52
11	After physical close Keepfile.	Deallocating tape units.	0, 4, 8, 12, 40, 44, 48, 52
12	After physical close Savefile.	Deallocating tape units.	0, 4, 8, 12, 40,

continued on next page

Illus. 6.1 *continued*

Exit Number	Logical SYSTEM 2000 Exit Point	Possible Applications	Allowable Action Codes
13	SCF (SCF TP) input line received before parsing.	Modifying input line or replacing with alternate input line.	0, 4, 8, 12, 16, 28, 40, 44, 48, 52
14	SCF line of output is ready.	Modifying output line or replacing with alternate input line.	0, 4, 8, 12, 20, 32, 40, 44, 48, 52
15	Parsed SCF input.	Verifying value range to user requirement.	0, 4, 8, 12, 24, 36, 40, 44, 48, 52
16	Formatted SCF output.	Modifying current value through table lookup.	0, 4, 8, 12, 20, 32, 40, 44, 48, 52
17	PLEX subschema record received.	Verifying data fields, encode.	0, 4, 8, 12, 24, 36, 40, 44, 48, 52
18	PLEX subschema record ready for output.	Modifying values through table look-up.	0, 4, 8, 12, 40, 44, 48, 52
19	Input variable in PLEX subschema record.	Modifying/verifying data value.	0, 4, 8, 12, 40, 44, 48, 52
20	Formatted variable for output to PLEX subschema record.	Modifying value through table look-up.	0, 4, 8, 12, 40, 44, 48, 52
21	Upon entry to S2KWAIT.	Monitoring resource utilization of user.	0, 4, 8, 12, 40, 44, 48, 52
22	End of input (that is, SCF batch EOF or SCF TP EOS).	Switching files and continuing input stream.	0, 4, 8, 12, 40, 44, 48, 52
23	Prior to physical open of each database file.	Inspecting or modifying each database DCB prior to physical open.	0, 4, 8, 12, 40, 44, 48, 52
24	Prior to physical close of each database file.	Inspecting or modifying each database DCB prior to physical close.	0, 4, 8, 12, 40, 44, 48, 52
25	Prior to a logical open of a database. (Multi-User only).	Preparing for dynamic deallocation.	0, 4, 8, 12, 16, 28, 40, 44, 48, 52
26-35	(Reserved for future use.)		
36	Prior to a physical close of a database. (Multi-User only).	Preventing Multi-User from physically closing a database.	0, 4, 8, 12, 16, 28
37	After a logical close of a database. (Multi-User only).	Companion to EXIT03 or EXIT25.	0, 4, 8, 12
38	All authority checkpoints.	Detecting security violations.	0

continued on next page

Illus. 6.1 *continued*

Exit Number	Logical SYSTEM 2000 Exit Point	Possible Applications	Allowable Action Codes
39	SCF input record read.	Replacing SCF input buffer with a new buffer.	0, 4, 8, 12, 16 28, 40, 44
40	(Reserved for future use.)		
41	After Coordinated Recovery completes.	Writing your own log when recovery occurs.	0, 8, 44
42	At SYSTEM 2000 termination time.	Allows clean-up tasks; last user-exit that is called.	0, 4
43	Prior to processing a Multi-User console command.	Inspecting or modifying a Multi-User console command.	0, 4, 8, 12, 16, 28
44-49	(Reserved for future use.)		
50§	Request for new SCF input.	Direct reading of input from various files.	0, 4, 8, 12, 40, 44, 48, 52
51§	After reading a data record.	Verifying current input record or replacing with alternate input.	0, 4, 8, 12, 16, 28, 40, 44, 48, 52
52§	Prior to output of a line.	Modifying current output line or replacing with alternate line.	0, 4, 8, 12, 20, 32, 40, 44, 48, 52
53§	End of input (that is, SCF batch EOF or SCF TP EOS).	Switching files and continuing input stream.	0, 4, 8, 12, 40, 44, 48, 52
54-63§	(Reserved for future use.)		

§ Multi-User dependent region exits, batch SCF jobs.

TERMS AND TABLES USED BY USER EXITS

Logical SYSTEM 2000 Exit Points

Logical SYSTEM 2000 exit points occur at predefined logical processing points within SYSTEM 2000 software. Illus. 6.1 on page 6-4 summarizes the available exits. At these exit points, control is transferred by SYSTEM 2000 software to the S2EXIT interface routine if the user has enabled the exit. The detailed specifications for EXIT00 through EXIT63 are given in **User-Exit Descriptions** on page 6-17.

When a logical exit point occurs, SYSTEM 2000 software checks the available exit mask. See the AVALEXIT field in the exit parameter list specifications, Illus. 6.2 on page 6-8. If the exit is allowed by the mask, control is transferred to the S2EXIT interface routine. S2EXIT then clears and builds the exit parameter list containing addresses of the SYSTEM 2000 data areas that are available to the specific user exit being taken. Then S2EXIT transfers control to the user-exit routine indicated in the user-exit load list. See AUSRLOAD in Illus. 6.2 on page 6-8.

The logical exits EXIT00 through EXIT49 originate in SYSTEM 2000 software and are available in both single-user and Multi-User environments. The logical exits EXIT00 and EXIT50 through EXIT63 originate in the Multi-User dependent region SYS2KJOB and are available for Multi-User batch SCF jobs, but not for Multi-User dependent region PLEX jobs.

Note: EXIT00 is available in both single-user and Multi-User environments. The routine executed for EXIT00 can be the same or different in each environment.

Illus. 6.2 Exit Parameter Specifications

Parameter Label	Address of	Field Description	Exits
AUSRCTL	user control block	29 fullwords	
		(1) - exit number (binary)	All
		(1) - execution mode (binary)	All
		0 - SCF	
		1 - PLEX	
		0 - master console command	43
		1 - SZOP console command	43
		(1) - environment (binary)	All
		0 - single user	
		≠ - Multi-User	
		(1) - action code (binary)	All
		(1) - PLEX return code set by user (binary)	All
		(1) - flag for ENABLE/DISABLE EXITS command (binary)	01
		0 - disable	
		1 - enable	
		(2) - bit mask for ENABLE/DISABLE EXITS command	01
		(20) - names for ENABLE/DISABLE EXITS command	01
		SCF: 10 fields, 8 characters each (characters are left-aligned and blank-filled)	
		PLEX: data from an 80-byte array	
		(1) - address of user work buffer (set by user exit) or	9-14,22-24, 50-53
		SCF tape indicator:	
		=1 if KEEP or RESTORE	
		>1 if SAVE or APPLY	
ATMPSTOR§	temporary storage	100 fullwords available to user routines	All
AEXTWT§§	exit wait routine	Entry point address	01,03-25,39
APASSWRD§§	password	1 fullword (character)	01,03-25,39
ADBNAME§§	database name	4 fullwords (character)	01,03-25,39
ACCTINFO§§	ACCT information (Multi-User only)	8 fullwords (character)	03-22,25,37, 39
		(2) - jobname	
		(2) - stepname	
		(2) - program name	
		(2) - terminal ID (for TP only)	
ACOMBLK§§	COMMBLOCK control block (PLEX only)	Address not given if program does not use COMMBLOCK	03-07,09-12, 17-20,25, 36-38
ASCHEMA§§	SSR control block (PLEX only)	Address not given if program does not use SSR.	05,06,17-20, 38

continued on next page

Illus. 6.2 *continued*

Parameter Label	Address of	Field Description	Exits
AS2KDUM§§	S2KDUM control block (PLEX only)	See PLEX manual for S2KDUM format.	01,03-07,09-12, 17-20,25,37,38
ABUFSIZ§§§	database page buffer size	1 fullword containing byte count (binary)	05,06
ADBBUF§§§	current database page buffer	Buffer page size is specified in parameter ABUFSIZ	05,06
AERMSGLN§	user-message length	1 fullword containing byte count Initially set to max size of 118 characters (binary)	01,03-63
AERMSGBF§	user-message buffer	30 fullwords (message length in AERMGSLN parameter)	01,03-63
ABUFTBL	maximum allowable buffer table	6 fullwords 1 word for each database file (binary)	05-07
ACOMPTYP§§§	component type	1 fullword (binary) 1 - CHARACTER 2 - TEXT 3 - DATE 4 - INTEGER 5 - DECIMAL 6 - MONEY 7 - unused 8 - REAL 9 - DOUBLE 10 - UNDEFINED	15,16,19,20
ACOMPLN§§	component length	1 fullword Number of characters (binary) SCF: actual value length for parsed input SCF: field size to contain output PLEX: defined length from database definition	15,16,19,20,26
ACOMPKY§§	component key indicator	1 fullword (binary) 0 - key 1 - non-key	15,16,19,20
ACOMPFLT§§	component decimal specification	1 fullword Number of positions to right of decimal point for DECIMAL and MONEY values (binary)	15,16,19,20
ACOMPVAL§§	component value	Variable size 250-byte max SCF - character PLEX - as declared in SSR	15,16,19,20,26
AIBUFPOS	input buffer position	Address of the input buffer position	39

continued on next page

Illus. 6.2 *continued*

Parameter Label	Address of	Field Description	Exits
AIBUFSIZ§§§	size of input buffer or syntactic unit size	1 fullword For input buffer size is LRECL limit (binary) For syntactic unit size is character count	08,13,22,39,43, 50,51,53
AIBUF§§§	input buffer or current syntactic unit	Address of input buffer whose size is specified in AIBUFSIZ	08,13,15,22,39, 43,50,51,53
AOBUFPOS	reserved		
AOBUFSIZ§§	output buffer size	1 fullword LRECL limit for output buffer size (binary)	14,52
AOBUF§§§	output buffer	Variable size (4K max) size is specified in AOBUFSIZ	14,52
AVALEXIT	available exit mask	5 fullwords one bit per exit (only first two words are used) 0 - disabled 1 - enabled	00,01
AUSRLOAD§§	user-exit load list	64 fullwords one word per user-exit entry point, address	00,01
ADCB	DCB address	BDAM DCB	23,24
AEXTGM	reserved		
ADBIO§§	database I/O count	2 fullwords when segment statistics active (packed)	13,14,21
ATIM§§	elapsed CPU time	1 fullword when segment statistics active (packed)	13,14,21
ASEGIO§§	database I/O count for Multi-User segments	2 fullwords when segment statistics active (packed)	13,14,21
ASEGTIM§§	segment elapsed CPU time	1 fullword when segment statistics active (binary)	15,16,19,20
ACOMPNO§§	user component number	1 fullword (extended binary)	15,19,38
APTYPE	processing type or security violations	1 fullword (binary) 0 - where-clause value 1 - update value security violations if EXIT38	

continued on next page

Illus. 6.2 *continued*

Parameter Label	Address of	Field Description	Exits
AFILENO§§	database file number	1 character-file number (binary)	05,06,23,24
AECB§§	ECB	14 fullwords (for field description, see Illus. A.7 on page A-10 starting with the ECBNAM field description)	19,20
AWAITCDE	WAIT code	1 fullword (binary) 0 - I/O wait 4 - (DOS/VS only) I/O wait for use of data file 8 - buffer wait 12 - database wait 16 - overlay wait 20 - tape wait	21
AHASHFN	CALC information	Not used	
AMHBUFSZ	MLH buffer size	1 fullword (binary) Range 512 to 50000	07

§ The temporary storage buffer and the user-message buffer addresses are used for one exit execution, but the contents are available for subsequent exit calls.

§§ These fields are used by SYSTEM 2000 DBMS and are altered regardless of action code.

§§§ Buffer sizes must conform to the provided buffer sizes to avoid writing outside of the buffer area.

The S2EXIT Interface Routine

The S2EXIT interface routine is supplied by SAS Institute Inc. S2EXIT creates the exit parameter list and then transfers control to the appropriate user-exit routine indicated in the user-exit load list. You can load S2EXIT dynamically, or you can link-edit it with SYSTEM 2000 software for single-user jobs, at Multi-User initialization, or with SYS2KJOB for dependent region Multi-User batch SCF jobs.

User-Exit Routines

A user-exit routine is the user-supplied code that is executed at a logical SYSTEM 2000 exit point (if that exit is enabled). A user-exit routine can be coded for each logical exit point, or the routine can apply to several (or all) logical exit points. The routine receives control from the S2EXIT interface routine.

After the user-exit routine processes the available SYSTEM 2000 data areas indicated in the exit parameter list, any messages or return codes should be posted in the proper data areas. User messages returned from a user-exit routine are written to the user's Message File for both SCF and PLEX jobs. A PLEX return code can be set in some user-exit routines by putting the desired return code in the PLEX return code field in the user control block and setting the action code to a 8 or 12. For details, see Exit Detail Specifications in Illus. 6.2 on page 6-8.

The action code field in the user control block must be set before control is returned to the S2EXIT interface routine. If the action code specifies the termination of a PLEX user, a SYSTEM 2000 Error Code occurs. The valid action codes are shown below. Not all the action codes are available to all user exits. See the detailed description for each user exit.

Illus. 6.3 User-Exit Action Codes

Action Code	Description
0	Good return; no modifications
4	Good return; data areas modified
8	Good return; print message
12	Good return; data areas modified, print message
16	Reject input; continue processing
20	Reject output line; continue processing
24	Reject command; continue processing
28	Reject input; print user message; continue processing
32	Reject output line; print user message; continue processing
36	Reject command; print user message; continue processing
40	Terminate user; can damage a database
44	Terminate user; print user message; can a damage database
48	Terminate user only if database would be left undamaged
52	Terminate and print message if database left undamaged

Exit Parameter List

The exit parameter list, Illus. 6.2 on page 6-8, forms the standardized control block communication area that is the communication link between the S2EXIT interface routine and the user-exit routine. It contains the addresses of the SYSTEM 2000 data areas that are available for examination or modification by the user-exit routine. S2EXIT initializes and rebuilds this list before it passes control to the user-exit routine indicated in the user-exit load list.

According to standard IBM linkage conventions, the high-order byte of the last usable address in the exit parameter list is set to X'80' by the S2EXIT interface routine. The names of the fields in this list (supplied by SAS Institute Inc.) are provided by the EXTPARM macro when it is expanded into the DSECT for the exit parameter list. (See also Illus. 6.8 on page 6-65.) Always refer to these addresses by the names provided in the EXTPARM macro when using Assembler language and not by offsets from the beginning. This allows you to incorporate future changes more easily.

User-Exit Load List

The user-exit load list is a control table of addresses for the available user-exit routines. The address of the user-exit load list is in AUSRLOAD in Illus. 6.2 on page 6-8. This table consists of 152 fullword addresses corresponding to the available exit points within SYSTEM 2000 software. The fullword addresses have a one-to-one relationship to each exit number, that is, Word 1 corresponds to the entry address for the EXIT00 user routine, Word 2 corresponds to the entry address for the EXIT01 user routine, and so forth.

All address words are initialized to binary zeros at SYSTEM 2000 initialization time except the first word, which is set to the address of the S2KEXIN entry point, with the high order byte of the last word set to X'80'. The user-exit routine EXIT00 (S2KEXIN module) or EXIT01 must fill in all necessary entry point addresses in the load list for the user-exit routines that are called when logical SYSTEM 2000 exits are taken. The addresses in the user-exit load list can all be different (one for each exit), all the same (one user-exit routine for all exits), or a mixture thereof.

ENABLING USER EXITS

EXITS Parameter

```
EXITS= |NO      (default)
       |YES
```

The EXITS parameter controls the overall execution of all user exits. EXITS=YES must be specified on the EXEC statement for all Multi-User dependent region jobs requiring user exits 50 through 63. EXITS=YES directs the dynamic loading of the S2EXIT interface routine for that particular execution of SYSTEM 2000 software, unless S2EXIT has been linked with SYSTEM 2000 code.

If EXITS=NO (the default), user exits cannot be used unless the S2EXIT interface routine has been linked with SYSTEM 2000 software. EXITS=NO or the absence of this parameter disables the dynamic loading of the S2EXIT interface routine.

Note: If the S2EXIT interface routine has been link-edited with SYSTEM 2000 software, the EXITS parameter is ignored. That is, the user-exits capability is enabled regardless of the EXITS parameter setting.

ENABLE/DISABLE ROUTINE Command

The ENABLE/DISABLE ROUTINE command for SCF and PLEX jobs controls the execution of specific user exits. This command is documented in *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

The ENABLE/DISABLE ROUTINE command does not actually enable or disable the specified user exits. It calls the user-supplied EXIT01 routine. Then the EXIT01 routine checks the parameters in the command and creates appropriate settings in the available exit mask and exit parameter list. The *names* and *exits* parameter values from the command are reflected in the enable/disable names and bit mask fields of the user control block. See AUSRCTL in Illus. 6.2 on page 6-8.

The ENABLE/DISABLE ROUTINE command cannot be used to dynamically enable or disable the Multi-User dependent region user exits EXIT50 through EXIT63. These exits can be enabled or disabled only at SYS2KJOB initialization time (EXIT00), which occurs in the dependent region SYS2KJOB when a Multi-User batch SCF job is started.

SETTING UP AND EXECUTING USER EXITS

Although only 36 exits are presently implemented, SYSTEM 2000 software can support up to 152 logical exits.

The logical exit points for EXIT00 through EXIT49 reside in SYSTEM 2000 software. The implementation of these exits is shown in Illus. 6.1 on page 6-4. The logical exit points for EXIT00 and EXIT50 through EXIT63 reside in the dependent region SYS2KJOB, which is used for all Multi-User batch SCF jobs. EXIT64 through EXIT151 are reserved for future use.

A summary chart of the available user exits, their logical SYSTEM 2000 exit points, possible applications, and allowable action codes is provided in Illus. 6.1 on page 6-4.

Any user-exit routines to be executed in a Multi-User environment can use the temporary working storage provided by SYSTEM 2000 software for any modifiable data areas, in order to ensure reentrancy. See Illus. 6.2 on page 6-8 for exit parameter list specifications. If this storage area is insufficient, users can provide their own dynamic storage area.

The S2EXIT interface routine supports any reentrant user-supplied exit routine in a single-user or Multi-User environment. Non-reentrant routines can be used only in a single-user environment or in the Multi-User dependent region exits (EXIT50 through EXIT63) for batch SCF jobs. We suggest that you code all user-exit routines in Assembler language and that you make them reentrant.

You must supply the user-exit routine for any exit that is allowed, that is, EXIT00 through EXIT63. There is no restriction on the usable user-exit module names, except for EXIT00. The user-exit routines for EXIT00 must have the module name S2KEXIN.

EXIT00 and EXIT01 are the only user exits that are permitted access to the available exit mask (AVALEXIT) and the user-exit load list (AUSRLOAD), shown in Illus. 6.2 on page 6-8. For loading and linking S2EXIT and user-exit routines, see **Load/Link Considerations** on page 6-62. All execution of user-exit routines is controlled by the user. EXIT00 is always taken if user exits are enabled with the execution parameter EXITS=YES or if the S2EXIT interface routine has been linked with SYSTEM 2000 software. The routine for EXIT00 must always be provided. EXIT00 is taken when SYSTEM 2000 software is initialized or when a Multi-User batch SCF job is started. EXIT00 always establishes the mapping of the logical exits to user-exit routines in the user-exit load list. The supplied EXTWAIT macro should be used for I/O waits done in the user-exit routine (see **I/O Wait Constraints** on page 6-63). Also, use the EXTPARM macro, which expands into the DSECT for the user-exit parameter list.

The S2EXIT interface routine external reference to S2KEXIN can be resolved by linking it with SYSTEM 2000 software or the dependent region executable code, or it can reside on the STEPLIB library where it is available for dynamic loading. When the execution parameter EXITS=YES is specified, the reference is checked for a resolution via link-edit. If no reference is found, an attempt is made to load S2KEXIN from the STEPLIB library. If EXITS=NO is specified and the reference was resolved via link-edit, the execution parameter is ignored. Otherwise, no attempt is made to load S2EXIT.

The user-exit routine S2KEXIN for EXIT00 can either be link-edited with S2EXIT, or it can reside on the STEPLIB library. Control is passed to the S2KEXIN routine by the S2EXIT interface routine, along with the address of the user-exit load list. User-exit routines can either be link-edited or dynamically loaded, but the user is responsible for updating the bit mask and user-exit load list entries for applicable exits.

31-Bit Mode under XA Systems

SYSTEM 2000 software operates in 31-bit mode under XA (Extended Architecture) systems. (Non-XA systems use 24-bit addressing mode.) Your user exits are entered in the same mode that SYSTEM 2000 software is using. In 31-bit mode, do not clutter the high order byte of a register or word with flags if that register or word contains an address; 31 bits are required for the address. In addition, the data areas that you want to access (and address) are probably above the 16 megabyte line.

The 31-bit mode provides full 31-bit addressing, which can have an impact on your user-exit code. When SYSTEM 2000 software is running in 31-bit mode, your user-exit routine will be entered in 31-bit mode. Some of the addresses that are passed to your user exits require your exits to be in 31-bit mode (AMODE=31). Unless your user exit is performing I/O processing, minimal change should be needed to run your user-exit routines in 31-bit mode.

EXIT00 Execution

EXIT00 occurs whenever SYSTEM 2000 software is initialized or whenever a Multi-User batch SCF job is started, but only if the EXITS execution parameter is YES or the S2EXIT interface routine has been linked with SYSTEM 2000 software. If EXIT00 does not enable EXIT01 or any other exit, no exits can be taken.

EXIT01 Execution

EXIT01 occurs any time an ENABLE/DISABLE ROUTINE command is issued in either SCF or PLEX jobs. Whenever this command occurs and EXIT01 is allowed by the available exit mask, the S2EXIT interface routine sets up the exit parameter list for use by EXIT01. The EXIT01 routine should examine the enable/disable flag, the enable/disable bit-mask, and enable/disable names fields in the user control block (exit parameter label AUSRCTL) and determine what action is to be taken. EXIT01 must then set the address of the requested user-exit routine in the user-exit load list if the request is permitted.

The enable/disable flag (in the user control block of the exit parameter list) indicates whether the command issued was an ENABLE (equal to 1) or DISABLE (equal to 0) ROUTINE command. The enable/disable names and bit-mask fields in the user control block contain whatever parameter data are passed by the *names* and *exits* parameters, respectively, of the ENABLE/DISABLE ROUTINE command. The names field can contain parameter data designated by the person who codes the user-exit routines.

The bit-mask field contains a 1-bit in the position that corresponds to the exits indicated in the *exits* parameter of the ENABLE/DISABLE ROUTINE command. For example, if EXIT05, EXIT09, and EXIT11 are specified in the *exits* parameter, then bits 5, 9, and 11 have a 1 in the bit mask, counting from bit zero for EXIT00. The EXIT01 routine uses the information to modify the available exit mask and user-exit load list. Only EXIT00 and EXIT01 have access to these fields.

In summary, the user-exit routine for EXIT01 can dynamically alter (during execution) the enabled or disabled exit in the available exit mask and any user-exit-to-routine relationships in the user-exit load list that were initially established by the EXIT00 routine S2KEXIN at SYSTEM 2000 initialization time. This capability allows dynamic alteration for tailoring SYSTEM 2000 software to satisfy unique end user, system-wide, and installation-specific requirements.

Dependent Region Exit Execution

The dependent region exits EXIT00 and EXIT50 through EXIT63 for Multi-User batch SCF jobs cannot be dynamically enabled or disabled using the ENABLE/DISABLE ROUTINE command via EXIT01. They can only be enabled or disabled by EXIT00 in the dependent region whenever a Multi-User batch SCF job is initialized.

User-Exit Load List Mapping

EXIT01 for a Multi-User environment should try to ensure that the ENABLE/DISABLE ROUTINE command does not alter any exit-to-routine mapping in the user-exit load list that a currently active user relies on.

Installation Security

The EXIT00 and EXIT01 routines must determine and ensure that any security requirements of the installation, regarding usage of SYSTEM 2000 software and the user exits, are enforced. The exit parameter list provides access to any data necessary for enforcement of such security.

USER-EXIT DESCRIPTIONS

Exits 00 through 49 are the logical exits residing within SYSTEM 2000 software. These exits are available in single-user jobs and in the Multi-User environment, except where excluded. In a single-user or Multi-User environment, the S2EXIT interface routine resides in the same region as SYSTEM 2000 software and the user-exit routines for EXIT00 through EXIT49.

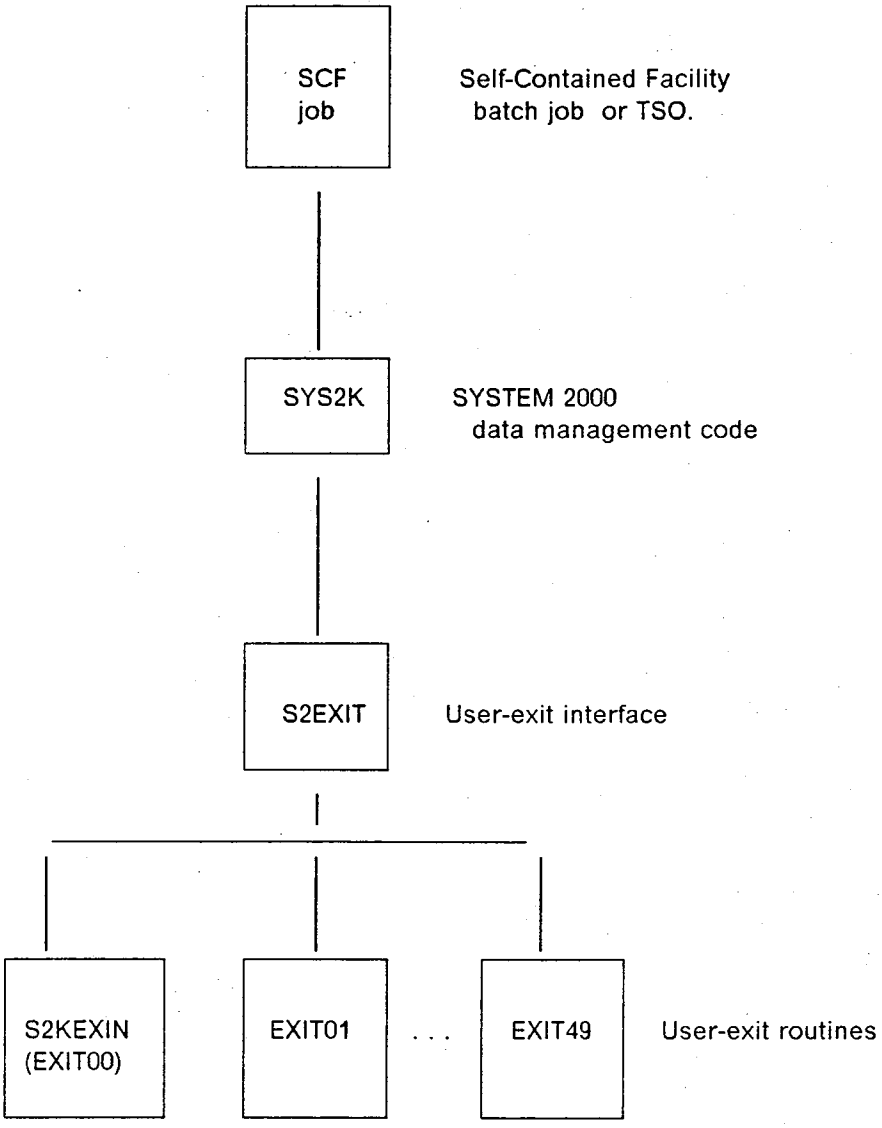
Make allowances for the increase in memory size and execution time for user-exit routines where the exits are used. Also, in a Multi-User environment, a thread is not released while executing any exit taken while in that thread.

Illus. 6.4 on page 6-18 shows how user exits 00 through 49 occur for SCF batch jobs running in a single-user environment. There is no SCF TP (interactive processing) in a single-user environment.

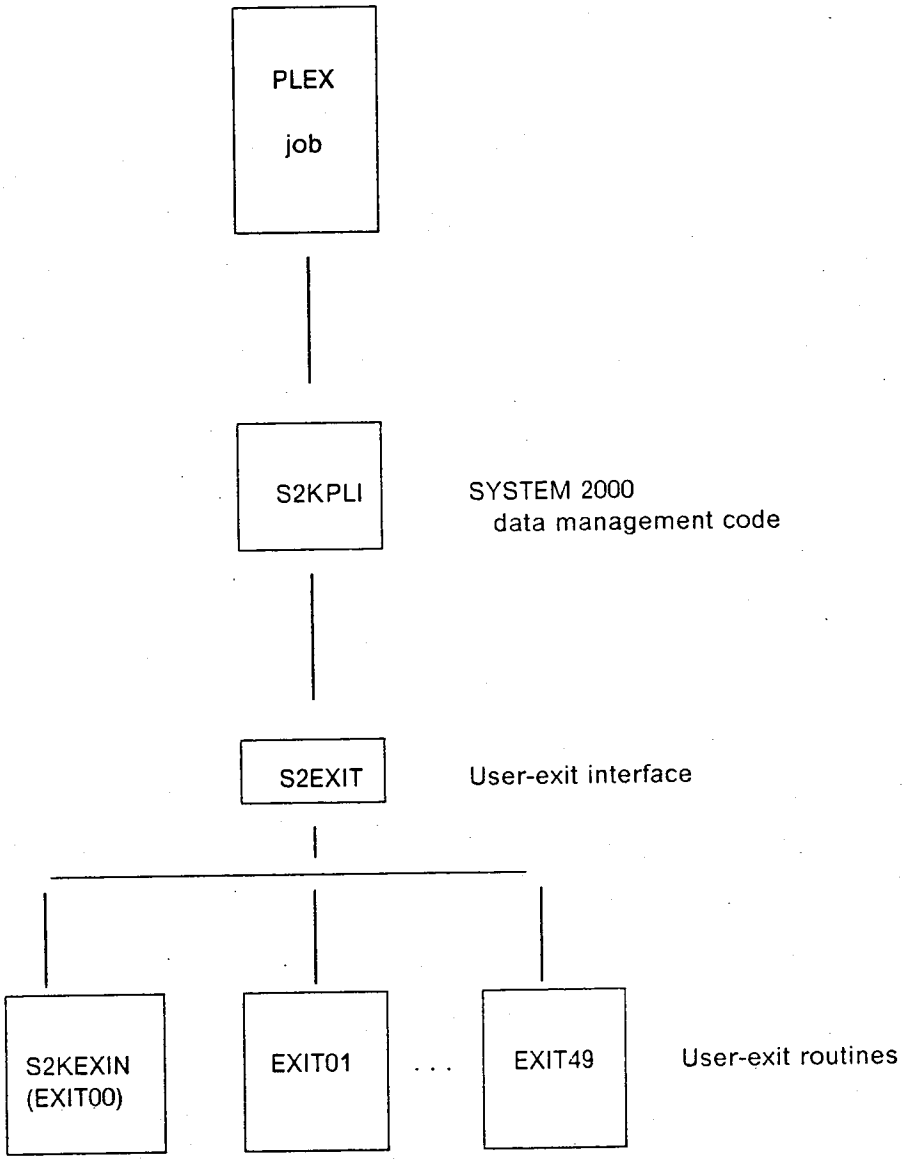
Illus. 6.5 on page 6-19 shows how user exits 00 through 49 occur for PLEX jobs executing in a single-user environment.

Illus. 6.6 on page 6-20 shows how user exits 00 through 49 occur in a Multi-User environment for batch SCF jobs, SCF TP jobs, and PLEX jobs.

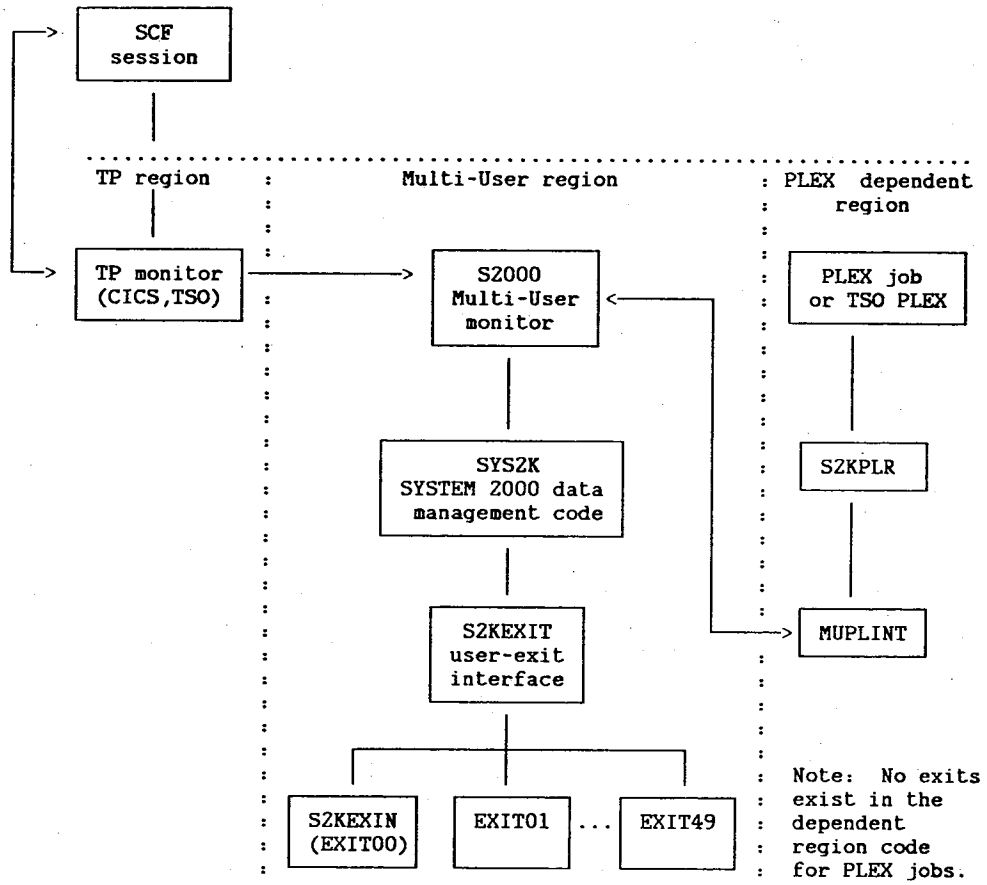
Illus. 6.4 Exits 00 through 49 for Single-User SCF Batch Jobs



Illus. 6.5 Exits 00 through 49 for Single-User PLEX Jobs



Illus. 6.6 Exits 00 through 49 for Multi-User Software



EXIT00

EXIT00 occurs during SYSTEM 2000 single-user or Multi-User initialization. For the dependent region user exits, EXIT00 occurs in the dependent region SYS2KJOB whenever a Multi-User batch SCF job is initialized. EXIT00 establishes user-exit availability and controls user-exit routine execution.

If the S2EXIT interface routine was not link-edited, it is dynamically loaded. The user-exit load list and available exit mask are initialized by S2EXIT. The user-exit routine for EXIT00 (S2KEXIN) is loaded if it was not link-edited. Control is then given to the S2EXIT interface routine. S2KEXIN should establish the user-exit load list and available exit mask to reflect the desired user-exit requirements during this execution of SYSTEM 2000 software. Because these data areas have been modified, the action code in the user control block should be set to 4 before returning control to the S2EXIT interface routine.

EXIT00 must be coded if the execution parameter EXITS = YES is specified at initialization time or if the S2EXIT interface routine was link-edited with either SYSTEM 2000 software or SYS2KJOB. If EXIT00 is taken and the S2KEXIN routine is not available, an IBM MVS operation exception occurs. EXIT00 must set up the available exit mask and user-exit load list addressed by AVALEXIT and AUSRLOAD, respectively, in the exit parameter list. These must be set to enable the necessary exits and to provide the entry point addresses of the routines to be called for any enabled exits.

EXIT00 and EXIT01 must ensure any security of SYSTEM 2000 data areas required by the installation.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AVALEXIT	available exit mask
AUSRLOAD	user-exit load list

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4

EXIT01

EXIT01 controls the availability and execution of user exits. EXIT01 occurs following the execution of the ENABLE/DISABLE ROUTINE command in either SCF or PLEX. The ENABLE/DISABLE ROUTINE command is documented in *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

The enable/disable flag in the user control block of the exit parameter list indicates whether the command issued was an ENABLE (equal to 1) or DISABLE (equal to 0) ROUTINE command. The *names* and *exits* parameter data from the ENABLE/DISABLE ROUTINE command are available in the user control block. You can use these data to modify the user-exit availability and user-exit routine execution control.

The enable/disable bit mask in the user control block indicates which exits are requested by the *exits* parameter. The *names* data are in the enable/disable names fields. The EXIT01 user-exit routine uses these data in any way required at your site.

For SCF, each name is 8 characters, left-aligned with blank-fill. Up to 10 names can be specified. For PLEX, the name data are an 80-byte array. The EXIT01 routine must ensure the availability of the user-exit routines by updating the user-exit load list and available exit mask. If these data areas are modified, the action code should be set to 4 or 12.

In a Multi-User environment, the user-supplied EXIT01 routine is responsible for ensuring that any ENABLE/DISABLE ROUTINE command issued does not alter the exit-to-routine mapping in the user-exit load list relied on by a currently active end user.

Note: EXIT01 cannot be used to enable/disable the dependent region user exits in SYS2KJOB, EXIT50 through EXIT63. Also, the use of EXIT01 to disable EXIT01 causes subsequent ENABLE/DISABLE ROUTINE commands to have no effect.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name, if available
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer
AVALEXIT	available exit mask
AUSRLOAD	user-exit load list

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT02

EXIT02 occurs at SYSTEM 2000 initialization time after EXIT00 has been taken. This exit can be used for any processing not logically connected with EXIT00. For example, you can dynamically allocate the database files, scratch pads, Locate Files, and the sort, S2KUSxxx, and S2KUSERS file at this time.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4

EXIT03

EXIT03 can be used to allocate database files dynamically, thereby reducing JCL and CLIST requirements. EXIT03 occurs prior to the database files being physically opened. Password security checks have not been done at this time. A PLEX return code can be set by this user exit.

Note: You can also use the ALLOC command to allocate the files dynamically in an SCF session. For details, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 8, 40, 44, 48, 52

EXIT04

EXIT04 can be used to deallocate database files dynamically, thereby freeing resources not needed for an entire job. EXIT04 occurs after SYSTEM 2000 physically closes the database files.

A PLEX return code can be set by this user exit.

Note: You can also dynamically deallocate database files with the FREE command in SCF sessions. For details, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 8, 40, 44, 48, 52

EXIT05

EXIT05 allows you to develop and use decryption routines for database file pages. (See EXIT06 for encryption routines.) EXIT05 occurs after the physical retrieval of any database file page except the Master Record (the first table of database File 1) or File 8 pages. A physical retrieval is an I/O to the database files, not a request for a page residing in memory.

The page retrieved should be decrypted in place. If decrypted in a work buffer provided by the user, the page must be moved into the buffer whose address is provided. Terminating a user at this time can result in a damaged database.

Providing this user exit after physical I/Os for database files allows the user-exit routine to decrypt database pages existing on disk in an encrypted format. Changes should be accompanied by action code 4 or 12.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
ASCHEMA	subschema record, PLEX only
AS2KDUM	S2KDUM, PLEX only
ABUFSIZ	database page buffer size
ADBBUF	current database page buffer
AERMSGLN	user message length
AERMSGBF	user message buffer
ABUFTBL	maximum allowable buffer table
AFILENO	database file number

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT06

EXIT06 allows you to develop and use encryption routines for database file pages. (See EXIT05 for decryption routines.) EXIT06 occurs prior to any database pages being written to disk, except for the Master Record (the first table of database File 1) or File 8 pages.

Do not use the current database page buffer for encryption. A work buffer must be provided by the user exit to hold the encrypted page. The new buffer address must be placed in the user work buffer field of the user control block (exit parameter AUSRCTL). Also, the action code should be set to 4 or 12 prior to returning control to the S2EXIT interface routine. If the page needs no encryption, set the action code to zero. Terminating a user at this time can result in a damaged database.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
ASCHEMA	subschema record, PLEX only
AS2KDUM	S2KDUM, PLEX only
ABUFSIZ	database page buffer size
ADBBUF	current database page buffer
AERMSGLN	user message length
AERMSGBF	user message buffer
ABUFTBL	maximum allowable buffer table
AFILENO	database file number

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT07

EXIT07 occurs after the physical opening of a database. This user exit permits modification of the table defining the maximum allowable buffers per database file that can be in memory concurrently. Reducing the number of buffers can increase I/Os, and increasing the number of buffers can decrease I/Os. Changes must be accompanied by action code 4 or 12.

For XA systems (MVS/XA and VM/XA), if you specify a buffer size in AMHBUFSZ for the Multiple Local Holds buffer, the buffer will be allocated with a GETMAIN, rather than from a pool. If a size is not specified, the default size is 23464. For non-XA MVS systems, the buffer is allocated with a GETMAIN only if you specify a size; if no size is specified, the buffer is allocated from a pool. (Non-XA VM systems always allocate the buffer from one of the buffer pools.)

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer
ABUFTBL	maximum allowable buffer table
AMHBUFSZ	MLH buffer size

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT08

EXIT08 can be used for command security or syntax substitution for SCF jobs. This exit occurs after a syntactic unit has been parsed. The SCF Command File is scanned and broken into syntactic units. A syntactic unit is either a special character or a string of alphanumeric characters without embedded blanks. For example, the partial command PRINT/NAME/EMPLOYEE NAME WHERE... would cause seven EXIT08 calls, one for each of PRINT, /, NAME, /, EMPLOYEE, NAME, and WHERE.

If a syntactic unit is modified, the new size (maximum of 250 characters) must also be set. The new syntactic unit must replace the current unit and must be blank-filled to a fullword boundary. Changes should be accompanied by action code 4 or 12.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password, where available
ADBNAME	database name, where available
ACCTINFO	ACCT information, Multi-User only
AERMSGLN	user message length
AERMSGBF	user message buffer
AIBUFSIZ	syntactic unit size
AIBUF	current syntactic unit

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 24, 36, 40, 44, 48, 52

EXIT09

EXIT09 occurs prior to the physical opening of a Keepfile for an APPLY or KEEP command. You can use this user exit to allocate the Keepfile dynamically in order to acquire needed resources and reduce JCL requirements. The Keepfile needs to be allocated dynamically only when a request for its use is generated.

Note: You can also let SYSTEM 2000 software dynamically allocate the Keepfile by not allocating it in the JCL or CLIST. For more details about the Keepfile, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

If the Keepfile is a tape, the last word of AUSRCTL equals 1 for a KEEP command and is greater than 1 for an APPLY command.

Note: When the operating system provides for dynamic allocation, the allocation is made if the resource is immediately available. Otherwise, the request is rejected, because dynamic allocation does not wait for a resource to be made available unless the program making the request is an authorized program. Authorization allows a program to modify the appropriate dynamic allocation request blocks, which enables the program to wait for the resource. For specific details, see dynamic allocation in the MVS Systems Programming Library, Job Management manual.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT10

EXIT10 occurs before a Savefile is physically opened for saving or restoring a database. This user exit can be used to allocate the Savefile dynamically in order to acquire needed resources and reduce JCL requirements. The DDname for the Savefile must be the first seven characters of the database name and a suffix of S. The Savefile is a QSAM file. Savefiles need to be allocated dynamically only when a request for their use is generated.

Note: You can also let SYSTEM 2000 software dynamically allocate the Savefile by not allocating it in the JCL or CLIST. For more details about the Savefile, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

If the Savefile file is a tape, the last word of AUSRCTL equals 1 for a RESTORE command and is greater than 1 for a SAVE command.

Note: When the operating system provides for dynamic allocation, the allocation is made if the resource is immediately available. Otherwise, the request is rejected, because dynamic allocation does not wait for a resource to be made available unless the program making the request is an authorized program. Authorization allows a program to modify the appropriate dynamic allocation request blocks, which enables the program to wait for the resource.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT11

EXIT11 occurs after the physical closing of a Keepfile following an APPLY command or a KEEP command. You can use this user exit to deallocate the Keepfile dynamically in order to release unneeded resources and reduce JCL requirements.

If SYSTEM 2000 software dynamically allocated the Keepfile (not through JCL or EXIT09), it deallocates the Keepfile dynamically after the APPLY or KEEP operation.

If the Keepfile is a tape, the last word of AUSRCTL equals 1 for a KEEP command and is greater than 1 for an APPLY command.

Note: When the operating system provides for dynamic allocation, the allocation is made if the resource is immediately available. Otherwise, the request is rejected, because dynamic allocation does not wait for a resource to be made available unless the program making the request is an authorized program. Authorization allows a program to modify the appropriate dynamic allocation request blocks, which enables the program to wait for the resource.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT12

EXIT12 occurs after the physical close of a Savefile, which occurs after a database has been saved or restored. This user exit can be used to do dynamic deallocation for the Savefile data set in order to release unneeded resources and reduce JCL requirements.

If SYSTEM 2000 software dynamically allocated the Savefile (not through JCL or EXIT10), it deallocates the Savefile dynamically after saving or restoring the database.

If the Savefile file is tape, the last word of AUSRCTL equals 1 for a RESTORE command and is greater than 1 for a SAVE command.

Note: When the operating system provides for dynamic allocation, the allocation is made if the resource is immediately available. Otherwise, the request is rejected, because dynamic allocation does not wait for a resource to be made available unless the program making the request is an authorized program. Authorization allows a program to modify the appropriate dynamic allocation request blocks, which enables the program to wait for the resource.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT13

EXIT13 occurs after SYSTEM 2000 software reads an SCF input record from the Command File or the Data File. EXIT13 can be used to trap non-SYSTEM 2000 input interspersed with SYSTEM 2000 input when you are interfacing with other software packages. For example, you can cancel the execution of any SCF command if certain values appear in an input record. EXIT13 can also be used to substitute SYSTEM 2000 syntax for installation-specific syntax, enhanced security, and command accounting by password.

With EXIT13, you can preprocess an SCF input line before SYSTEM 2000 software scans it. Records coming to EXIT13 from the user must have binary zeros at the end of the record, and the record size must be specified in AIBUFSIZ. The input line can be modified in place or in a work area. A work area must be used if the input buffer size will increase beyond the LRECL limit. If a work area is used, its address must be placed in the user work buffer field of the user control block (exit parameter AUSRCTL). If necessary, update the input buffer size. Return an action code of 4 or 12 if any modifications were made.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
AERMSGLN	user message length
AERMSGBF	user message buffer
AIBUFSIZ	input buffer size
AIBUF	input buffer
ADBIO	database I/O count
ATIM	elapsed CPU time
ASEGIO	segment database I/O count
ASEGTIM	segment elapsed CPU time

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 16, 28, 40, 44, 48, 52

EXIT14

EXIT14 occurs just before SYSTEM 2000 software writes an SCF output line to the Message File or Report File. The output line can be replaced, modified, or passed as input to another software package. This user exit can also be used to implement enhanced security or to reformat output with installation software.

The output line can be modified in place or in a work area. If a work area is used, its address must be placed in the user work buffer field of the user control block (exit parameter AUSRCTL). Residual information may be included in the output unless the entire buffer (up to LRECL size) is modified or cleared. If the output buffer size is changed to exceed the LRECL size, the excess is not used. Return an action code of 4 or 12 if any modifications were made. Terminating a user at this time can cause a damaged database.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
AERMSGLN	user message length
AERMSGBF	user message buffer
AOBUFSIZ	output buffer size
AOBUF	output buffer
ADBIO	database I/O count
ATIM	elapsed CPU time
ASEGIO	segment database I/O count
ASEGTIM	segment elapsed CPU time

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 20, 32, 40, 44, 48, 52

EXIT15

EXIT15 occurs after an SCF input value has been parsed in the input line. These values occur in update action-clauses, in where-clauses, and in loader streams on the Data File. This user exit can be especially useful for editing input of values that must fall within given ranges, for example, integers between a range of 5 to 9 or allowing only certain values as input in a name. You can also do a table look-up of values, so that encoded data can be stored instead of actual values.

The information provided at this user exit allows the user-exit routine to define a more limited range for values than SYSTEM 2000 software currently does. The component value can be modified only in place. Also, if the character count is changed, the component length must be updated. The value and its length must be acceptable for the database definition. If any modification is made, action code 4 or 12 must be returned.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
AERMSGLN	user message length
AERMSGBF	user message buffer
ACOMPTYP	component type
ACOMPLN	component length
ACOMPKY	component key indicator
ACOMPFLT	component decimal specification
ACOMPVAL	component value
ACOMPNO	component number
APTYPE	processing type

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 24, 36, 40, 44, 48, 52

EXIT16

EXIT16 occurs when a formatted SCF value is ready to be inserted in the output line (into a specific field size if for a LIST command). This user exit can be used to examine data values prior to output in order to enforce security-by-value access. You can also do a table look-up of encoded values so that the resulting report contains readable data.

The component value has been edited into a buffer that was blank-filled prior to storing the value. The output value must be changed in place, and the field size must be set to the correct character count or else the preset field size is used. Return an action code of 4 or 12 if any modifications were made.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
AERMSGLN	user message length
AERMSGBF	user message buffer
ACOMPTYP	component type
ACOMPLN	output field size
ACOMPKY	component key indicator
ACOMPFLT	component decimal specification
ACOMPVAL	component value
ACOMPNO	component number

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 20, 32, 40, 44, 48, 52

EXIT17

EXIT17 occurs prior to the processing of every PLEX command. The command can be a retrieval or an update, or it can be one for which there is no subschema record, for example, the CLEAR command. EXIT17 allows you to edit subschema record values to enforce a range, verification, or security constraint that is unique to the installation.

Be careful not to modify the subschema record in a way that makes it invalid for processing. The changed values must conform to the subschema record picture sizes. See the *SYSTEM 2000 PLEX Manual, Version 12, First Edition* for the subschema record format. Changes should be accompanied by action code 4 or 12.

For a single-user PLEX job, the modifications are in the subschema record for subsequent processing. For a Multi-User environment, the modifications are in a copy area and are returned to the user program only if the modifications survive as part of the retrieved subschema record.

A PLEX return code set at the time of this exit can be overridden by EXIT18 setting a new return code. A non-zero return code prevents any further processing for the PLEX command and passes immediately to EXIT18 (if it is enabled). Terminating a user at this time can result in a damaged database.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
ASCHEMA	subschema record, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 24, 36, 40, 44, 48, 52

EXIT18

EXIT18 occurs prior to the return to a PLEX program. The subschema record accessed is available for examination. The PLEX command could have been a retrieval, an update, or one for which there is no subschema record, for example, the QUEUE command. This user exit permits enforcement of security by value. Also, you can pass the subschema record to a subroutine provided by the user for subsequent report processing. Editing violations can also be enforced at this time.

The subschema record accessed, if any, is available for replacement, modification, or verification. The values in a changed subschema record must be acceptable for correct use by the PLEX program. For example, COBOL can redefine the subschema record into fields different from those portrayed for SYSTEM 2000 software. Changes should be accompanied by an action code of 4 or 12.

A PLEX return code can be set by putting the requested return code in the PLEX return code field of the user control block. Terminating a user at this time can result in a damaged database.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
ASCHEMA	subschema record, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT19

EXIT19 occurs after a component value in a PLEX subschema record has been identified. You can use EXIT19 for value verification, range editing, and security processing.

The value is in a subschema record for an INSERT or MODIFY command, or for a where-clause. The component value must be modified in place, honoring the subschema record picture size and type. Changes should be accompanied by action code 4 or 12.

For single-user PLEX jobs, the changed value is in the subschema record for subsequent processing. In a Multi-User environment, the changed value is in a copy area and is not returned to the user program.

A PLEX return code can be set by putting the requested return code in the PLEX return code field of the user control block. A non-zero return code prevents any further processing of the PLEX command. Terminating a user at this time can result in a damaged database.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
ASCHEMA	subschema record, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer
ACOMPTYP	component type
ACOMPLN	component length
ACOMPKY	component key indicator
ACOMPFLT	component decimal specification
ACOMPVAL	component value
ACOMPNO	component number
APTYPE	processing type
AECB	ECB

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT20

EXIT20 occurs after a retrieved component value has been placed in the PLEX subschema record. The information provided at this user exit allows custom editing, range checking, table look-up on encoded values, and enforced security.

The value in the subschema record can be changed, but it must be acceptable for correct use by the PLEX program, for example, COBOL can redefine a value. Changes must be accompanied by an action code of 4 or 12.

A PLEX return code can be set by putting the requested return code in the PLEX return code field of the user control block. Terminating a user at this time can result in a damaged database.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
ASCHEMA	subschema record, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer
ACOMPTYP	component type
ACOMPLN	component length
ACOMPKY	component key indicator
ACOMPFLT	component decimal specification
ACOMPVAL	component value
ACOMPNO	component number
APTYPE	processing type
AECB	ECB

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT21

EXIT21 occurs whenever the SYSTEM 2000 routine S2KWAIT is called. User resource utilization can be accumulated. Maximum limits can be set for user requests.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
AERMSGLN	user message length
AERMSGBF	user message buffer
ADBIO	database I/O count
ATIM	elapsed CPU time
ASEGIO	segment database I/O count
ASEGTIM	segment elapsed CPU time
AWAITCDE	WAIT code

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT22

EXIT22 occurs after an end-of-file (EOF) on the SCF Command File for a batch SCF job or after an SCF TP end-of-segment (EOS) has been detected. You can do end-of-job functions or supply additional input.

If you supply additional input at this time, set the input buffer size parameter to the address of the new logical record length. Set the user work buffer address field of the control block to the address of the new field containing the Command File syntax, and set action code 4 or 12.

When you supply additional input, EXIT22 continues to be taken until an action code of 0 is returned or the user is terminated with a 40, 44, 48, or 52 action code. After EXIT22 is called, no additional commands can be read from the Command File, even if the Command File is changed to an alternate Command File.

Note: In a Multi-User environment, be sure to specifically identify each user if you need to determine whether a call to this exit is a first call for that user or a subsequent one.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
AERMSGLN	user message length
AERMSGBF	user message buffer
AIBUFSIZ	input buffer size
AIBUF	input buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT23

EXIT23 occurs after EXIT03 and before EXIT07. It can be used to inspect or modify the DCB of each database file, or for security checking. EXIT23 is called once for each DCB of a database to be physically opened.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
AERMSGLEN	user message length
AERMSGBF	user message buffer
ADCB	DCB
AFILENO	database file number

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT24

EXIT24 can be used to inspect or modify the DCB of each database file. It occurs prior to EXIT04. EXIT24 is called once for each DCB of a database to be physically closed.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
AERMSGLEN	user message length
AERMSGBF	user message buffer
ADCB	DCB
AFILENO	database file name

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT25

EXIT25 occurs prior to the logical open of a database only if a physical open does not occur first. This exit is for a Multi-User environment only.

EXIT25 can be used to prevent a user from signing on to a database; it sets an action code of 16 or 28. An action code of 16 or 28 simulates trying to sign on to a database that is already under exclusive use. Action code 28 allows printing of a message; action code 16 does not.

EXIT25 causes Return Code 94 to be issued to a PLEX program even if an OPEN command is issued (not just for an OPENR command). A FRAME command that forces a logical open also calls EXIT25. After the user has all databases opened, any switching among databases does not invoke this exit.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 16, 28, 40, 44, 48, 52

EXIT26 through EXIT35

Reserved for future use.

EXIT36

EXIT36 occurs prior to a physical close of the database files. This is a Multi-User exit only. You can use EXIT36 to prevent Multi-User from doing a physical close of the database files by setting an action code of 16 or 28.

Note: The password is normally unusable because it belongs to the database being opened next. That is, the password is for the database that caused the physical close of this one.

EXIT36 and Special Zap 59 are mutually exclusive. That is, if EXIT36 is enabled, Special Zap 59 has no effect. Eventually, support for Special Zap 59 will be dropped because EXIT36 replaces it functionally.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 16, 28

EXIT37

EXIT37 always occurs after a logical close of a database regardless of whether a physical close (EXIT36) is required. If the database is to be physically closed, EXIT36 is executed (if enabled) after EXIT37. (See EXIT36.) EXIT37 is a Multi-User exit only and is a companion to the open exits, EXIT03 and EXIT25.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information, Multi-User only
ACOMBLK	COMMBLOCK, PLEX only
AS2KDUM	S2KDUM, PLEX only
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12

EXIT38

EXIT38 occurs when a security violation is detected by SYSTEM 2000 software. With EXIT38, you can record security violations that the software detects. Such fields as password, database name, job name, and terminal id are provided for logging purposes.

Listed below are the types of security violations that can be detected and a description of when they occur.

- A password is not on the list of valid passwords for opening the requested database.
- A password in the COMMBLOCK of a PLEX program does not have the authority required to access the component referenced in a PLEX statement.
- A secondary password holder issues an SCF command containing a component the password does not have authority to use.
- A secondary password issues a command reserved for the DBA password holder or the master password holder. Also, the DBA password holder cannot issue a unauthorized command. To see a list of commands that can be used by the DBA password as well as the master password, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*
- A secondary password holder has violated the restrictions on a database with Security by Entry. The restrictions include

  - not establishing the Entry Key
  - specifying an Entry Key condition that contains an operator other than EQ
  - joining the Entry Key condition to another expression with OR.

  For more details about Security by Entry violations, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*
- Other security violations occur when a user attempts to add or change a password that already exists.

S2EXIT sets APTYPE, which is one of the exit parameter fields. APTYPE is a fullword in length and contains a value that corresponds to the security violation that has occurred. Listed below are the values of APTYPE and the security violations that correspond to the values.

APTYPE Value	Security Violation
F0F1 0001	SCF component authority violation
F0F2 0002	PLEX component authority violation
F0F2 0003	master or DBA password violation
F0F4 0004	invalid password at database open
F0F5 0005	Security by Entry violation
F0F6 0006	other

The first halfword of the APTYPE value contains the security violation number in character format for user exits that cannot be written in Assembler. The second halfword of the APTYPE value contains the security violation number in binary format.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information
ACOMBLK	COMMBLOCK control block (PLEX)
ASCHEMA	subschema record control block (PLEX)
AS2KDUM	S2KDUM control block (PLEX)
APTYPE	processing type, used to indicate violation type

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0

EXIT39

EXIT39 occurs prior to the software's reading the Command File, before another I/O. EXIT39 enables you to specify an input buffer of commands; it can be more than 250 characters long.

Initially, AIBUFSIZ and AIBUF equal -1. If you change AIBUF from -1 to an input buffer address, you must specify the buffer length in AIBUFSIZ.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information
AERMSGLN	user message length
AERMSGBF	user message buffer
AIBUFPOS	address of input buffer position
AIBUFSIZ	length of input buffer
AIBUF	address of input buffer
ADBIO	database I/O count
ATIM	elapsed CPU time
ASEGIO	segment database I/O count
ASEGTIM	segment elapsed CPU time

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 16, 28, 40, 44

EXIT40

EXIT40 is reserved for future use.

EXIT41

EXIT41 occurs immediately after Coordinated Recovery has recovered a database.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
APASSWRD	password
ADBNAME	database name
ACOMBLK	COMMBLOCK control block (PLEX)
AS2KDUM	S2KDUM control block (PLEX)
AERMSGLN	user message length
AERMSGBF	user message buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 8, 44

EXIT42

EXIT42 occurs at SYSTEM 2000 termination time; it is the last exit called. You can use EXIT42 for clean-up tasks, such as freeing any storage obtained with GETMAINs, deleting modules loaded, closing any files used by your user-exit code, and so on.

Regardless of whether you use EXIT42, SYSTEM 2000 software frees memory obtained for user exits and deletes S2EXIT if it was loaded at initialization time.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AVALEXIT	available exit mask
AUSRLOAD	user-exit load list

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4

EXIT43

EXIT43 allows you to examine operator console requests to Multi-User software. This exit occurs prior to Multi-User processing of the command.

EXIT43 can determine whether the command was issued by the master console or with an S2OP call. The EMODE field, which is the second word of the user control block (AUSRCTL), will equal 1 if S2OP issued the command or 0 if the master console issued the command.

The software gives EXIT43 the command buffer that Multi-User will process. Any modification of the command in this buffer will be passed to Multi-User software.

If the master console issued the command, EXIT43 has the following constraints:

- EXIT43 is not available until the first user signs on to Multi-User software.
- SYSTEM 2000 software does not print any message contained in AERMSGBF, the user message buffer. You must specify your own WTO if you want to send a message to the console.
- If EXIT43 returns an invalid action code, SYSTEM 2000 software sets the action code to zero, and processing will continue.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AERMSGLN	user message length
AERMSGBF	user message buffer
AIBUFSIZ	input buffer size
AIBUF	input buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 16, 28

EXIT44 through EXIT49

Reserved for future use.

Overview of EXIT00 and Exits 50 through 63

EXIT00 and exits 50 through 63 are the logical exits residing in the dependent region SYS2KJOB interface routine used for all Multi-User batch SCF jobs. The user-exit routines for EXIT00 and for EXIT50 through EXIT63, as well as the S2EXIT interface routine, reside in the same region as SYS2KJOB when they are executed for these exits. Thus, user-exit routine execution for these user exits is tied directly to the execution of SYS2KJOB. These user exits are available only during SYS2KJOB execution for batch SCF jobs.

The ENABLE/DISABLE ROUTINE command for either an SCF and or a PLEX job does not dynamically affect the status of exits 50 through 63 in the dependent region. The status of these user exits can only be modified when the dependent region EXIT00 occurs, because they reside in the SYS2KJOB routine.

You cannot dynamically modify the status of the dependent region exits while SYS2KJOB is running.

The S2EXIT interface routine for the dependent region exits can either be loaded dynamically or link-edited with SYS2KJOB. If S2EXIT is link-edited to SYS2KJOB, the execution parameter EXITS=NO does not disable user exits for the dependent region, and EXIT00 is always taken. The S2EXIT interface routine used in the dependent region is the same as that used for the single-user and Multi-User EXIT00 through EXIT49. However, it executes as a separate copy in the dependent region.

The code for the dependent region initialization EXIT00 exit can be the same as for the EXIT00 routine used for the single-user and Multi-User initialization EXIT00 exit, or it can be different. Bit settings and load list initialization for EXIT01 through EXIT49 have no effect on the dependent region execution. In either case, the EXIT00 code executes as a separate copy in the dependent region.

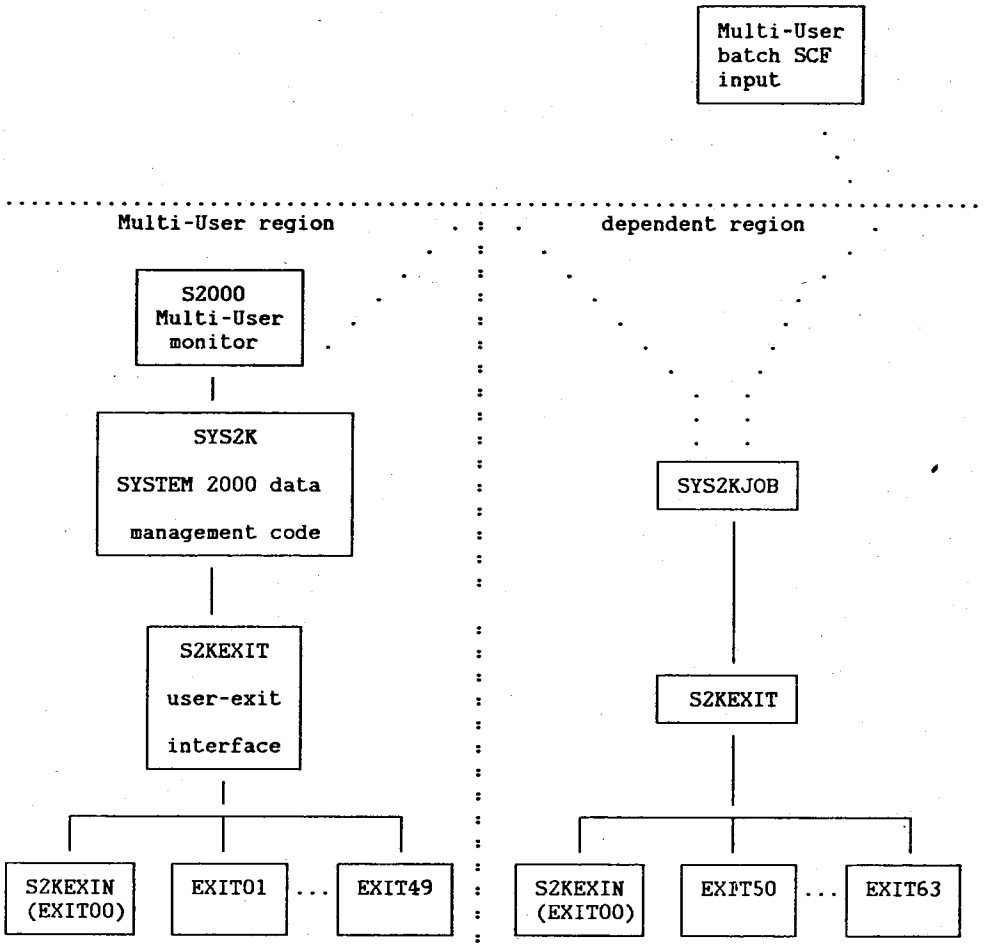
The S2EXIT interface routine in the dependent region is the same as that used for the single-user and Multi-User exits EXIT00 through EXIT49, but it executes as a separate copy in the dependent region. The EXIT00 routine for the dependent region exits can be the same routines as used for for EXIT00 through EXIT49, or they can be different. In either case, the routines reside in the dependent region.

The EXIT00 routine must set up the available exit mask and user-exit load list for EXIT50 through EXIT63. When you use EXIT50 through EXIT63, make allowances for the increase in memory size and execution time required. Also, you cannot use the EXTWAIT macro (for I/O waits in the user-exit routine) in any user-exit routines for EXIT50 through EXIT63. There are no logical user exits in the dependent region code for Multi-User PLEX jobs. (See **Execution-Time PLEX Exits** on page A-15 for PLEX user exits.)

Illus. 6.7 on page 6-56 shows how EXIT50 through EXIT63 occur for dependent region Multi-User batch SCF jobs.

The other topics in this section describe to EXIT50 through EXIT63.

Illus. 6.7 EXIT50 through EXIT63 for Multi-User Batch SCF Jobs (Dependent Region)



Note: The S2KEXIN code for EXIT00 resides with both the SYSTEM 2000 code and the SYS2KJOB code, in their respective regions. Even if the S2KEXIN code is the same for both situations, it must reside as a separate copy in each region, regardless of whether the routine is loaded dynamically or is link-edited.

EXIT50

EXIT50 occurs when SYSTEM 2000 software requires a new input line, just prior to reading the SCF Command File. The user exit either routes the software to the source of an input line, or it indicates that the software is to proceed with reading the Command File.

If an input line is needed, the user exit controls the source of the input line from among several input sources in the SYS2KJOB region. If a replacement buffer is supplied, it must contain the desired SCF input. Also, its address must be set in the user work buffer address field of the control block, even if the normal AIBUF input buffer address is used. If necessary, the input buffer size can be modified. Set the action code to 4 or 12.

If no input line from another source is required, EXIT50 indicates that the software should go ahead and read the Command File. Indicate "no action" by setting an action code of 0 or 8.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AERMSGLN	user message length
AERMSGBF	user message buffer
AIBUFSIZ	input buffer size
AIBUF	input buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT51

EXIT51 occurs after SYSTEM 2000 software has read a new SCF input line/record from the Command File or the Data File. When this user exit occurs, the input line/record can be verified, modified, or substituted.

The input line can be modified in place or in a work area. A work area must be used if the input buffer size is to increase beyond the LRECL limit. If a replacement buffer is supplied, the address must be set in the user work buffer address field of the user control block. If necessary, the input buffer size can be modified, and the action code set to 4 or 12.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AERMSGLN	user message length
AERMSGBF	user message buffer
AIBUFSIZ	input buffer size
AIBUF	input buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 16, 28, 40, 44, 48, 52

EXIT52

EXIT52 occurs prior to SYSTEM 2000 software writing an output line/record, but after the line/record has been formatted for the Message File or the Report File. When this user exit occurs, the output line can be verified, modified, or substituted by the user-exit routine.

If you want to replace the buffer, the address of the replacement buffer must be set in the user work buffer address field of the user control block. Residual information may be included in the output unless the entire buffer up to LRECL size is modified or cleared. Your exit code should return an action code of 4 or 12 if any modifications were made. If you change the output buffer size to exceed the LRECL size, the excess is not used.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AERMSGLN	user message length
AERMSGBF	user message buffer
AOBUFSIZ	output buffer size
AOBUF	output buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 20, 32, 40, 44, 48, 52

EXIT53

EXIT53 occurs after SYSTEM 2000 software detects an end-of-file condition on the SCF Command File. When this user exit occurs, end-of-job processing can be done by the user-exit routine.

If you want to continue the input, the user-exit routine must set a new buffer address in the user work buffer address field of the control block. If necessary, update the input buffer size, and set an action code of 4 or 12. After the new input has been processed, EXIT53 is called again until an action code of 0 is returned or until terminated. If new input is not supplied, no further input can be read from the Command File.

The parameters available for this exit are as follows:

Parameter Label	Field Description
AUSRCTL	user-exit control block
ATMPSTOR	temporary storage
AERMSGLN	user message length
AERMSGBF	user message buffer
AIBUFSIZ	input buffer size
AIBUF	input buffer

See Illus. 6.2 on page 6-8 for field characteristics and special constraints.

The allowable action codes are as follows: 0, 4, 8, 12, 40, 44, 48, 52

EXIT54 through EXIT63

Reserved for future use.

PROGRAMMING STANDARDS

Introduction

Some user-exit functions implemented by various users are basically similar. SAS Institute Inc. has established programming standards for coding user exits so that user-exit routines that do fundamentally the same task do not need to be coded (at least not in their entirety).

Language

User-exit routines should be coded in Assembler language (ALC), using standard IBM calling conventions. Routines must be reentrant for a Multi-User environment. Only in special cases should a routine not be reentrant or be coded in a higher level language (see calling parameters).

Use the EXTWAIT macro (for I/O waits in the user-exit routine) in addition to the EXTPARM macro, which expands into the DSECT for the exit parameter list. All routines must be clearly documented in the source code. The user-exit routines must be as efficient as possible without sacrificing readability and maintainability. Any user-exit routines to be considered for user exchange should conform to the language constraints listed above.

Documentation

All routines submitted to SAS Institute Inc. for evaluation must be accompanied by complete external user documentation. This documentation must describe the function or functions that the user-exit routine performs. JCL and resource requirements must be defined, as well as operating system restrictions. If applicable, the documentation must define all options for input and provide examples describing their use and expected results.

All user-exit messages to be issued must be defined. These messages are written to the user's Message File by the S2EXIT interface routine. User exits permit users to issue a new series of messages with user-supplied text in them. For details, see **User-Exit Messages and Codes** on page 6-70. User exits that are limited to using only the action codes of 0 or 4 cannot issue any messages.

Even though EXIT00 and EXIT02 cannot issue any messages because of the action code limitation, they can issue standard IBM MVS messages, because Multi-User is not yet executing when these exits are taken.

Document all foreseeable conditions that can cause program termination. Also, provide recovery instructions.

Impact on Memory Requirements

Memory requirements must increase to accommodate the user-exit routines in the SYSTEM 2000 region, as well as the working storage areas needed to support reentrant routines. This increase can be calculated as follows:

memory size increase = (T * C) + UR + UI

where

T is the number of threads. For single-user and dependent region exits, T always equals 1.

C is a work area of 848 bytes.

UR is the size of the user-supplied routines.

UI is the size of the S2EXIT interface routine.

Load/Link Considerations

The S2EXIT interface routine and any user-exit routines that are to be dynamically loaded must reside on a STEPLIB library.

To link-edit the S2EXIT interface routine with SYSTEM 2000 software, specify UE=YES in the PRELNK macro. (See Chapter 7, "Configuring SYSTEM 2000 Software: PRELNK Macro.") With this specification, the execution parameter EXITS=NO is ignored, user exits are always enabled, and EXIT00 is always taken. This specification does not link S2EXIT with the dependent region code SYS2KJOB.

The EXIT00 routine S2KEXIN (or any other user-exit routine) can be linked with SYSTEM 2000 software by specifying the module name in the QA parameter of the PRELNK macro.

The S2EXIT interface routine and user-exit modules can be linked with SYS2KJOB by using INCLUDE statements in a link-edit job of SYS2KJOB. If S2EXIT is included, the execution parameter EXIT=NO is ignored. User exits are always enabled, and EXIT00 is always taken for the dependent region exits.

The statement below shows the use of the PRELNK macro configuration that includes the Rollback Log, the REPORT processor, Multi-User, User-Exits, and the Self-Contained Facility. The UE=YES and QA macro parameters specify that both the S2EXIT interface routine and the EXIT00 module (S2KEXIN), respectively, are to be linked with SYSTEM 2000 software.

```
PRELNK OV=FLAT,RB=YES,RW=YES,MU=YES,UE=YES,QA=(S2KEXIN),LANG=NL
```

The following statement shows the use of the PRELNK macro to generate a non-overlayed SYSTEM 2000 configuration that includes the REPORT processor, user exits, and the Self-Contained Facility. Including user exits (UE=YES parameter) specifies that the S2EXIT interface routine is to be linked with SYSTEM 2000 software. The QA macro parameter specifies that the EXIT00 module S2KEXIN and the QAEXIT module are to be linked with SYSTEM 2000 software.

```
PRELNK OV=FLAT,UE=YES,RW=YES,LANG=NL,QA=(S2KEXIN,QAEXIT)
```

The following statements show how to link the user-exit S2EXIT and the EXIT00 module S2KEXIN with the dependent region SYS2KJOB.

```
// EXEC  LKED
//R120    DD DSN=S2K.R120.LOAD,DISP=SHR
//USERLIB DD DSN=S2K.R120.PLILOAD,DISP=SHR
  INCLUDE R120(SYS2KJOB)
  INCLUDE R120(S2EXIT)
  INCLUDE USERLIB(S2KEXIN)
  ENTRY MSYS2K
  NAME SYS2KJOB
```

I/O Wait Constraints

All user-exit routines that do I/O in a Multi-User environment must use READ and WRITE requests in conjunction with the EXTWAIT macro. This practice assures that the SYSTEM 2000 wait logic is used and provides for proper performance. GET and PUT (as well as any action that causes a wait state) must be avoided, because the waits or interrupts generated cannot be controlled by SYSTEM 2000 software.

The following EXTWAIT macro must be used to issue any waits in a user-exit routine:

```
EXTWAIT ECB=x [,WAITEP=y]
```

- x register containing address of DECB to be waited upon. This should define an area of nine fullwords.
- y register containing address of WAIT entry point in SYSTEM 2000 software. This address is kept in AEXTWT of the exit parameter list.

Control is returned to the user-exit routine after the ECB has been posted as complete.

Note: The EXTWAIT macro cannot be used in user-exit routines resulting from SYSTEM 2000 initialization (EXIT00 and EXIT02) or in a routine executing as a result of dependent region SYS2KJOB user exits (EXIT00 and EXIT50 through EXIT63). These user exits can use standard WAITs, GETs, and PUTs.

Here is an example of the EXTWAIT macro and its expansion.

```
SOURCE STATEMENT                                    ASM 0201 17.22
   EXTWAIT ECB=5,WAITEP=7

   LR      1,5          ADDRESS OF AN AREA (9 FULL WORDS) FOR DECB

   LR      15,7         ADDRESS OF EXTWAIT ENTRY POINT

   BALR    14,15

   EXTWAIT ECB=4
   LR      1,4          ADDRESS OF AN AREA (9 FULL WORDS) FOR DECB

   L       15,AEXTWT    ADDRESS OF EXTWAIT ENTRY POINT

   BALR    14,15
```

Calling Parameters for S2EXIT

The S2EXIT interface routine makes all calls to the user-exit routines indicated in the user-exit load list using standard IBM calling procedure conventions.

The contents of registers are shown here.

Register 1 contains the address of the exit parameter list.

Register 13 contains the address of an 18 fullword save area in which the user-exit routine being called can save the S2EXIT interface routine's registers

Register 14 contains the return address of the S2EXIT interface routine.

Register 15 contains the address of the user-exit routine being called.

S2EXIT is written in Assembler language. We recommend writing all user-exit routines in Assembler language. If the user-exit routine is written in FORTRAN or COBOL, the return format conventions of the respective higher level language should be used (the RETURN statement in FORTRAN, the GOBACK statement in COBOL). For PL/I user-exit programs, see the IBM PL/I Programmer Guide for how to establish linkage to an Assembler program.

The action code in the user control block (exit parameter AUSRCTL) is cleared to binary zeros before control is passed to a user-exit routine. If the action code is to be non-zero, it must be set to a valid action code by the user-exit routine prior to returning to the S2EXIT interface routine. S2EXIT interrogates the action code upon return from the user-exit to determine whether further processing is required. Failure to properly set the action code, especially after a data area has been modified (possibly incorrectly), can result in unpredictable results in SYSTEM 2000 execution. If the action code is set to an invalid value, the user job that caused the exit is terminated with a SYSTEM 2000 Error Code.

EXAMPLE OF EXIT00

The sample EXIT00 in Illus. 6.8 on page 6-65 enables EXIT07, EXIT13, and EXIT42. This example uses reentrant EXITBGN and EXITEND macros. It also uses the EXITAREA macro, which is expanded into the DSECT for the exit parameter list, user control table, AVAILEXT, user-exit load list, message buffer, maximum allowable buffer table, and temporary storage. The SETEXIT macro enables the desired exits in the available exit mask and sets the load address for that exit in the user-exit load list.

Illus. 6.8 Example of EXIT00

```
 2*-------------------------------------------------------------------*
 3 *                                                                   *
 4 *  THIS IS THE USER EXIT CALLED AT SYSTEM 2000 INITIALIZATION TIME. *
 5 *  ITS PURPOSE IS TO ENABLE USER EXITS BY SETTING THE AVAILABLE     *
 6 *  EXIT BITS AND STORING THE ADDRESS OF THE USER EXIT IN THE        *
 7 *  LOADLIST.                                                        *
 8 *                                                                   *
 9 *  THIS IS A SIMPLE EXAMPLE THAT USING V-TYPE ADDRESS CONSTANTS     *
10 *  TO GET THE ADDRESS OF THE USER EXIT.  THE SETEXIT MACRO TURNS    *
11 *  ON THE BIT AND STORES THE ADDRESS IN THE LOAD LIST.              *
12 *                                                                   *
13 *-------------------------------------------------------------------*
15 S2KEXIN  EXITBGN
16+R0       EQU   0
17+R1       EQU   1
18+R2       EQU   2
19+R3       EQU   3
20+R4       EQU   4
21+R5       EQU   5
22+R6       EQU   6
23+R7       EQU   7
24+R8       EQU   8
25+R9       EQU   9
26+R10      EQU   10
27+R11      EQU   11
28+R12      EQU   12
29+R13      EQU   13
30+R14      EQU   14
31+R15      EQU   15
32+R00      EQU   0
33+R01      EQU   1
34+R02      EQU   2
35+R03      EQU   3
36+R04      EQU   4
37+R05      EQU   5
38+R06      EQU   6
39+R07      EQU   7
40+R08      EQU   8
41+R09      EQU   9
42+S2KEXIN  CSECT
43+         USING S2KEXIN,R15
44+         B     ST0001              BRANCH AROUND ID
45+         DC    X'07',CL7'S2KEXIN'  IDENTIFIER
46+STR0001    DC  A(0)                ADDRESS OF WORKING STRG.
47+ST0001    DS   0H
```

continued on next page

Illus. 6.8 *continued*

```
 48+          STM   R14,R12,12(R13)     -SAVE REGS
 49+          LR    R12,R15
 50+          DROP  R15                 DROP R15 BEFORE USING R12
 51+          USING S2KEXIN,R12         BASE REGISTER FOR PROGRAM
 53+*       THE FOLLOWING CODE IS NOT REENTRANT, BUT, BECAUSE THE
 54+*    THREAD WILL NOT RELINQUISH CONTROL WITHIN S2K DURING THE
 55+*    EXECUTION OF THE CODE, PSEUDO-REENTRANCY IS MAINTAINED.
 56+          L     R14,STR0001         LOAD STORAGE ADDRESS
 57+          LTR   R14,R14             IS STORAGE THERE?
 58+          BNZ   G0001               YES...DON'T GETMAIN AGAIN
 59+          LA    R0,1024
 60+          BAL   1,*+4                   INDICATE GETMAIN
 61+          SVC   10                      ISSUE GETMAIN SVC
 62+          LR    R14,R1              COPY STORAGE ADDRESS
 63+          ST    R14,STR0001         STORE STORAGE ADDRESS
 64+          L     R1,24(R13)          RESTORE R1
 65+G0001     DS    0H                  HAVE STORAGE ADDRESS
 67+          ST    R13,4(,R14)         -UP PTR
 68+          ST    R14,8(,R13)         -DOWN PTR
 69+          LA    R0,72+0(,R14)       -ADD IN STORAGE LENGTH
 70+          ST    R0,0(R14)           -INC AVAILABLE STACK PTR
 71+          L     R0,20(,R13)         -RESTORE R0
 72+          LR    R13,R14             -BASE FOR WORKING STORAGE
 73+          USING WS,R13
 74+WS        DSECT
 75+          DS    18F
 76+S2KEXIN   CSECT
 77 *
 78 *  INITIALIZE BASE REGISTER FOR NECESSARY CONTROL BLOCKS
 79 *
 80           LR    R3,R1               GET ADDRESS OF EXIT PARAMETERS
 81           USING EXTPARM,R3          R3 IS BASE FOR EXIT PARAMETERS
 82           L     R11,AUSRCTL         GET ADDRESS OF EXIT CONTROL BLOCK
 83           USING USERCB,R11          R11 IS BASE OF USER EXIT CONTROL BLK
 84           L     R10,AVALEXIT        GET ADDRESS AVALEXIT BITS
 85           USING AVAILEXT,R10        R10 IS BASE FOR AVAILABLE EXIT BITS
 86           L     R9,AUSRLOAD         GET EXIT ADDRESS LIST TABLE
 87           USING LOADLIST,R9         R9 IS BASE FOR LOADLIST
 88 *
 89 *  USE SETEXIT MACRO TO ENABLE EXITS 7, 13, AND 42.
 90 *
 91           SETEXIT 7,ON,LOAD=EXIT07
 92+          OI    AVAILEXT+0,B'00000001'
   +                                    SET EXIT BITS
 93+          L     R1,EXIT07
 94+          ST    R1,LOADLIST+7*4     SET ADDR IN LOAD LIST
```

continued on next page

Illus. 6.8 *continued*

```
 95          SETEXIT 13,ON,LOAD=EXIT13
 96+         OI    AVAILEXT+1,B'00000100'
  +                                    SET EXIT BITS
 97+         L     R1,EXIT13
 98+         ST    R1,LOADLIST+13*4     SET ADDR IN LOAD LIST
 99          SETEXIT 42,ON,LOAD=EXIT42
100+         OI    AVAILEXT+5,B'00100000'
  +                                    SET EXIT BITS
101+         L     R1,EXIT42
102+         ST    R1,LOADLIST+42*4     SET ADDR IN LOAD LIST
104 RTNEND   EQU   *                    END OF ROUTINE
105 *
106 *  SET ACTION CODE
107 *
108          LA    R15,4                GOOD RETURN, DATA AREAS MODIFIED
109          ST    R15,ACTION           SET ACTION CODE
110          EXITEND
111+         DS    0H
112+         L     R13,4(,R13)
113+         LM    R14,R12,12(R13)
114+         XR    R15,R15
115+         BR    R14
117 *
118 *   DEFINE V-CON ADDRESSES FOR REQUESTED USER EXITS.
119 *
120          DS    0F                    WORKING STORAGE
121 EXIT07   DC    V(S2KEX07)
122 EXIT13   DC    V(S2KEX13)
123 EXIT42   DC    V(S2KEX42)
125          EXITAREA
126+EXTPARM  DSECT
127+*  S2K - EXTPARM - USER EXIT PARAMETER LIST
128+*****************************************************************
129+*        USER EXIT PARAMETER LIST
130+*****************************************************************
131+AUSRCTL  DS    F          ADDRESS USER CONTROL BLOCK
132+ATMPSTOR DS    F          ADDRESS TEMPORARY STORAGE
133+AEXTWT   DS    F          ADDRESS ENTRY POINT FOR USER WAIT MACRO
134+APASSWRD DS    F          ADDRESS PASSWORD
135+ADBNAME  DS    F          ADDRESS DATA BASE NAME
136+ACCTINFO DS    F          ADDRESS OF ACCOUNTING INFORMATION
137+ACOMBLK  DS    F          ADDRESS PLI COMMBLOCK
138+ASCHEMA  DS    F          ADDRESS PLI SCHEMA
139+AS2KDUM  DS    F          ADDRESS PLI S2KDUM
140+ABUFSIZ  DS    F          ADDRESS DATA BASE PAGE SIZE
```

continued on next page

Illus. 6.8 *continued*

```
141+ADBBUF   DS    F          ADDRESS DATA BASE BUFFER
142+AERMSGLN DS    F          ADDRESS USER MESSAGE LENGTH
143+AERMSGBF DS    F          ADDRESS USER MESSAGE BUFFER
144+ABUFTBL  DS    F          ADDRESS ALLOWABLE BUFFER TABLE
145+ACOMPTYP DS    F          ADDRESS OF COMPONENT TYPE FLAG
146+ACOMPLN  DS    F          ADDRESS OF COMPONENT LENGTH
147+ACOMPKY  DS    F          ADDRESS OF COMPONENT KEY/NON-KEY FLAG
148+ACOMPFLT DS    F          ADDRESS OF COMPONENT FLOATING POINT DESIGNATOR
149+ACOMPVAL DS    F          ADDRESS COMPONENT VALUE
150+AIBUFPOS DS    F          ADDRESS INPUT BUFFER CURRENT POSITION
151+AIBUFSIZ DS    F          ADDRESS INPUT BUFFER SIZE
152+AIBUF    DS    F          ADDRESS INPUT BUFFER
153+AOBUFPOS DS    F          ADDRESS OUTPUT BUFFER CURRENT POSITION
154+AOBUFSIZ DS    F          ADDRESS OUTPUT BUFFER SIZE
155+AOBUF    DS    F          ADDRESS OUTPUT BUFFER
156+AVALEXIT DS    F          ADDRESS AVAILABLE EXITS MASK WORDS
157+AUSRLOAD DS    F          ADDRESS USER LOAD LIST
158+ADCB     DS    F          ADDRESS DCB
159+AEXTGM   DS    F          ADDRESS ENTRY POINT FOR USER GETMAIN MACRO
160+ADBIO    DS    F          ADDRESS (SEG STATS) DATA BASE I/O COUNT
161+ATIM     DS    F          ADDRESS (SEG STATS) ELAPSED CPU TIME
162+ASEGIO   DS    F          ADDRESS (SEG STATS) DB I/O COUNT AT SEG START
163+ASEGTIM  DS    F          ADDRESS (SEG STATS) CPU TIME AT SEG START
164+ACOMPNO  DS    F          ADDRESS COMPONENT NUMBER
165+APTYPE   DS    F          ADDRESS PROCESS TYPE 0=WHERE CLAUSE,1=UPDATE
166+AFILENO  DS    F          ADDRESS DATA BASE FILE NUMBER
167+AECB     DS    F          ADDRESS ECB
168+AWAITCDE DS    F          ADDRESS WAIT CODE FOR TYPE OF WAIT
169+AHASHFN  DS    F          ADDRESS HASH FUNCTION INFO BLOCK
170+AMHBUFSZ DS    F          ADDRESS MULTIPLE LOCAL HOLD BUFFER( EXIT7)
171+RES1     DS    F                    RESERVED
172+RES2     DS    F                    RESERVED
173+RES3     DS    X'80',AL3(0)   RESERVED
175+*        USER CONTROL TABLE
176+*
177+USERCB   DSECT
178+EXITCODE DS    F          EXIT NUMBER
179+EXECMODE DS    F          EXEC MODE - 0==NL,1==PLEX
180+ENVIRON  DS    F          ENVIRONMENT - 0==SA,1==MU
181+ACTION   DS    F          ACTION CODE
182+PLIRC    DS    F          PLI RETURN CODE
183+ENABLFLG DS    F          ENABLE/DISABLE FLAG 1==ENABLE,0==DISABLE
184+ENABLBIT DS    2F         ENABLE/DISABLE BITS
185+MODNAMES DS    10CL8      ENABLE/DISABLE MODULE NAMES
186+BUFFADDR DS    0F         USER'S/BUFFER ADDR
187+TAPEOP   DS    F                    1 => KEEP/RESTORE, >1 OTHERWISE
```

continued on next page

Illus. 6.8 *continued*

```
189+*          AVAILABLE EXIT BITS
190+AVAILEXT DSECT
191+           DS    5F
193+*          LOAD LIST = ARRAY OF EXIT ENTRY POINT ADDRESSES
194+*
195+LOADLIST DSECT
196+           DS    151F
198+*          MESSAGE BUFFER = CONTAINS MSG SET BY USER
199+*
200+USRBUF    DSECT
201+           DS    30F
203+*          MAXIMUM ALLOWABLE BUFFER TABLE
204+*
205+BUFTBL    DSECT
206+           DS    8F
208+*          TEMPORARY STORAGE AREA
209+*               = AREA PROVIDED TO USER FOR REENTRANT WORKING STORAGE
210+TEMPSTOR DSECT
211+           DS    100F
212            END
```

SAMPLE CODING TECHNIQUES FOR USER EXITS

The preceding sections introduced user exits and described the purpose, function, and output available for each exit. This section presents sample coding techniques for implementing user exits.

After the functions of the desired exits have been defined, the next process is writing the exits. The initialization exit, S2KEXIN (EXIT00), must be coded first because no other exits can be executed without it. We strongly recommend that you use the routine S2KEXIN provided by SAS Institute Inc. This exit processes the call table mentioned earlier so that enabling new exits is simple. You can add a new exit by adding a new routine name to the call table for EXIT00 and coding the routine to enable the desired exit. Logical portions of the processing can be enabled or disabled in separate routines. These logical portions can be chosen or omitted by adding or deleting names from the call table.

The next step is to write the remaining exits. We recommend that you use the code supplied by the Institute as a template. The entry code and working storage maintenance should be handled by the EXITBGN and EXITEND macros. If the exits are to be used in a Multi-User environment, they must be coded to maintain reentrancy, because only one copy of the exit is entered for each user in the system.

Finally, exits must be tested thoroughly before they are implemented in a production environment. The testing objectives should ensure

- that the exits do the desired task
- that the exits can be located
- that all error conditions are handled properly
- that all possible action codes are valid
- that reentrancy is maintained if the exits are to be used in a Multi-User environment.

Sample code for S2KEXIN (EXIT00), for the macros necessary to generate the call tables, and for sample exits (S2KEX07, S2KEX13, and S2KEX42) are provided on the Release 12.0 delivery tape to all customers with user exits.

User-Exit Messages and Codes

Messages issued by user exits A unique set of SCF message numbers and PLEX return codes is available for use by the user-exit routines. The text for these messages is determined by the user and is written with the user job output. The message numbering scheme is -9xx for SCF messages, and 9xx for PLEX return codes, where the range is 950 through 999. Message numbers 901 through 949 are reserved for use by the Institute.

To print these user-exit messages, the user-exit routine must place the message in the user message buffer. It must also update the user message length field that is addressed by the address in AERMSGLN and AERMSGBF, respectively, of the user-exit parameter list. (See Illus. 6.2 on page 6-8). The user-exit routine must then set the appropriate action code. (See Illus. 6.3 on page 6-12). Before the S2EXIT interface routine returns control to the software, the message in the message buffer is written to the Message File for that user. The user-exit routine must ensure that the message number precedes the message text.

Users must provide their own in-house documentation for any new user-exit messages. That documentation should be included where it is most appropriate, preferably as a special user-exit appendix in the necessary manuals. This user documentation should also be placed in your Messages and Codes manual, preferably as an appendix.

PLEX return codes issued by user exits PLEX return codes can be modified by the user-exit routines for EXIT03, EXIT04, and EXIT17 through EXIT20 by placing the appropriate return code in the PLEX return code field of the user-exit control block. This modification of a PLEX return code can be made only by substituting a valid PLEX return code.

Standard SYSTEM 2000 messages and codes For details about standard messages and codes, see *SYSTEM 2000 Messages and Codes, Version 12, First Edition.*

Sample Coding Techniques for S2KEXIN User Exit

The first user exit, S2KEXIN (EXIT00), must have a CSECT name of S2KEXIN. It is called at SYSTEM 2000 initialization time. S2KEXIN enables any additional user exits by setting the available exit bits and by storing the addresses of those user exits in the user-exit load list.

In the example provided, S2KEXIN uses V-type address constants to obtain the addresses of the other exits that are enabled. Then it utilizes the SETEXIT macro to turn on the appropriate bit and to store the address in the appropriate word in the user-exit load list.

Upon entry to S2KEXIN, you must save the calling routine's registers and establish addressability to the user exit parameter list, user exit control block, available exit bits, and user-exit load list. The sample Assembler language code shown below saves the registers using the EXITBGN macro and establishes the appropriate addressability.

```
S2KEXIN  EXITBGN
         LR     R3,R1              GET ADDRESS OF EXIT PARAMETERS
         USING  EXTPARM,R3         R3 IS BASE FOR EXIT PARAMETERS
         L      R11,AUSRCTL        GET ADDRESS OF EXIT CONTROL BLOCK
         USING  USERCB,R11         R11 IS BASE OF USER EXIT CONTROL BLK
         L      R10,AVALEXIT       GET ADDRESS AVALEXIT BITS
         USING  AVAILEXT,R10       R10 IS BASE FOR AVAILABLE EXIT BITS
         L      R9,AUSRLOAD        GET EXIT ADDRESS LIST TABLE
         USING  LOADLIST,R9        R9 IS BASE FOR LOADLIST
```

The main purpose of S2KEXIN is to enable other user exits and to provide their addresses. There are several ways of doing this. You can define a bit string constant with the appropriate bits set to 1; that is, 1 means the exit will be enabled. Addresses for the user exits can be obtained with the LOAD macro.

The sample S2KEXIN uses the supplied SETEXIT macro and V-type constants to supply the addresses of the other exits, which means all exits must be link-edited together.

```
            SETEXIT 7,ON,LOAD=EXIT07
            SETEXIT 13,ON,LOAD=EXIT13
            SETEXIT 42,ON,LOAD=EXIT42
               .
               .
               .
EXIT07   DC    V(S2KEX07)
EXIT13   DC    V(S2KEX13)
EXIT42   DC    V(S2KEX42)
```

In the final step, you must set the action code to 4, which indicates that modifications occurred. **Note**: An action code of 0 will not enable any exit.

The following code shows how to set the action code to 4 and return to S2EXIT:

```
   LA    R15,4              GOOD RETURN, DATA AREAS MODIFIED
   ST    R15,ACTION         SET ACTION CODE
   EXITEND
```

Sample Coding for EXIT01 THROUGH EXIT49

The S2EXIT routine is the interface between SYSTEM 2000 software and your user exits. It uses the DEXTPARM macro to obtain the required storage that defines all the variables and areas that are passed to your user exit. For Multi-User environments, one area is assigned for each thread.

The DEXTPARM macro is provided on the Release 12.0 installation libraries. This macro allows you to acquire addressability for all fields that may be passed to your user exit, using only one base register.

Upon entry to your user exit, R1 points to the parameter list. The second word of the parameter list points to the address of the 400-byte temporary storage area. The LOAD and USING instructions below demonstrate this technique. Sample exits S2KEX07, S2KEX13, and S2KEX42 provided on the Release 12.0 delivery tape use this technique.

```
   S2KEXnn  EXITBGN
            L     R3,4(,R1)          SECOND WORD OF PARMLIST IS
   *                                   TEMPORARY STORAGE
            USING DUSRPRM,R3
            .
            .
            .
            DEXTPARM
            END
```

EXITBGN macro EXITBGN provides pseudo-reentrant entrance code. It chains save areas, provides addressability, and does housekeeping. Your code should use EXITBGN.

Note: The default value for the STORAGE parameter is GETMAIN. In this situation, working storage and save area space for each exit are obtained with GETMAIN the first time the exit is called. The default area obtained is 1024 bytes. You can also specify STORAGE=TEMPSTOR, which means the exit uses the working storage passed to it by SYSTEM 2000 software. However, only 400 bytes are provided, and this may not be enough for many applications. Within this space, EXITBGN chains the save areas and working storage areas for the exit and its subroutines. They are chained in order.

The syntax for the EXITBGN macro is

```
label EXITBGN [length] [,S] [,WS=name]

          [,STORAGE= |GETMAIN ] [,STR SIZE=nnnn]
                     |TEMPSTOR
```

label is the name of the CSECT to be created.

length is a symbol or number that specifies the number of bytes (not including the save area) to be allocated for the exit's working storage.

S indicates that this is a subroutine for an exit.

name is the name of the DSECT used for save area and working storage space.

GETMAIN
TEMPSTOR
GETMAIN is the default, which obtains space for the save area and working area. TEMPSTOR indicates SYSTEM 2000 software will pass the working storage area to the exit. This parameter is ignored if you code the S option.

nnnn specifies the size of the area to be obtained with GETMAIN. The default is 1024 bytes. This parameter is ignored if you code the S option or if STORAGE equals TEMPSTOR.

EXITEND macro EXITEND should be used to return to the program that called the user-exit routine. The syntax for the EXITEND macro is

```
EXITEND
```

Chapter 7

Configuring SYSTEM 2000 Software: PRELNK Macro

SYSTEM 2000 CONFIGURATIONS: OVERVIEW

SYSTEM 2000 software consists of several functional modules, each of which performs a specific database management function or a set of related functions. Because a user session can use every module or only a few, the PRELNK macro gives each installation the opportunity to choose its own module configuration. In fact, the SYSTEM 2000 supervisor depends on you to supply complete information about the configuration structure for all versions of the software.

The default configuration supplied on the SYSTEM 2000 delivery tape is a flat version that uses more memory but needs no I/O for overlay usage. This version serves most sites.

Note: Field Fix distribution procedures require you to execute the PRELNK macro after you apply a Field Fix distribution.

Segments and Modules

This discussion of segments and modules pertains only to overlayed versions. The PRELNK macro allows you to generate a fully overlayed system, a flat system, or any variety of overlay configurations according to the needs at your site. For an overlayed system, a segment consists of one or more functional modules brought into memory simultaneously and treated as a single unit of executable code. SYSTEM 2000 executable memory contains at most three segments at one time -- one root segment, one primary segment, and one secondary segment. The software consists of only one root segment and several primary and/or secondary segments. Only the root segment can bring a primary segment into memory. Only a primary segment can bring a secondary segment into memory.

Segments at the same level (for example, several secondary segments) are mutually exclusive. That is, modules in different segments at one level are not callable by modules in other segments at the same level. If a required segment is not in memory, it must be brought in and perhaps swapped (overlayed) for the segment currently in memory.

The FULL and FLAT parameter options in the PRELNK macro allow you to select a preset configuration. The GROUP parameter in the PRELNK macro and the GROUP and ENDLNK macros allow you to generate a site-specific configuration for the root, primary, and secondary segments. The most efficient SYSTEM 2000 configurations minimize segment swapping. By knowing which modules a job uses -- when, how often, and in what combinations, you can decide how to group them into primary or secondary segments.

Each SYSTEM 2000 functional module is a member of a partitioned data set. Each module has a number and member name, as shown below.

Illus. 7.1 Module Numbers, Member Names, and General Functions

Module Number	Member Name	General Function
(0,0)	S2K00	Root
(1,0)	S2K10	DEFINE
(5,0)	S2K50	CONTROL
(3,0)	S2K30	QUEST
(3,1)	S2K31	QUEST syntax
(3,2)	S2K32	TALLY/DESCRIBE
(3,3)*	S2K33	QUEST where-clause
(3,4)	S2K34	Retrievals
(3,5)	S2K35	QUEUE syntax
(3,6)	S2K36	Hierarchical Table update
(3,7)	S2K37	Data Table update and variable sort
(3,8)	S2K38	Index update
(3,9)**	S2K39	QUEUE where-clause
(3,11)	S2K3B	APPLY
(3,12)	S2K3C	MAP, REORGANIZE, INDEX
(3,13)	S2K3D	REPORT syntax
(3,14)	S2K3E	REPORT record build
(3,15)	S2K3F	Report generation
(3,17)	S2K3H	PLEX
(3,18)	S2K3I	PLEX load mode
(3,19)*	S2K3J	Non-key where-clause
	S2KRW	REPORT processor companion to (3,0)
	S2KRBX	Dummy Rollback routine See PRELNK macro parameter considerations in **The PRELNK Macro Instruction** on page 7-9.
	S2KRB0	Rollback companion to (0,0); used in Coordinated Recovery processing.
	S2KRB3	Rollback companion to (3,0); used in Coordinated Recovery processing.
	S2KRB5	Rollback companion to (5,0); used in Coordinated Recovery processing.

* QUEST, PLEX and the REPORT processors use the QUEST where-clause modules (3,3) and (3,19).

** PLEX requires (3,9) for non-level-zero REMOVE TREE commands in queue mode.

SYSTEM 2000 Flow of Control

You need a clear understanding of the software's internal flow of control in order to configure a particular installation properly. Flow of control means each SYSTEM 2000 module must have access to any module it requires, regardless of whether the required module is in the same segment or a different one.

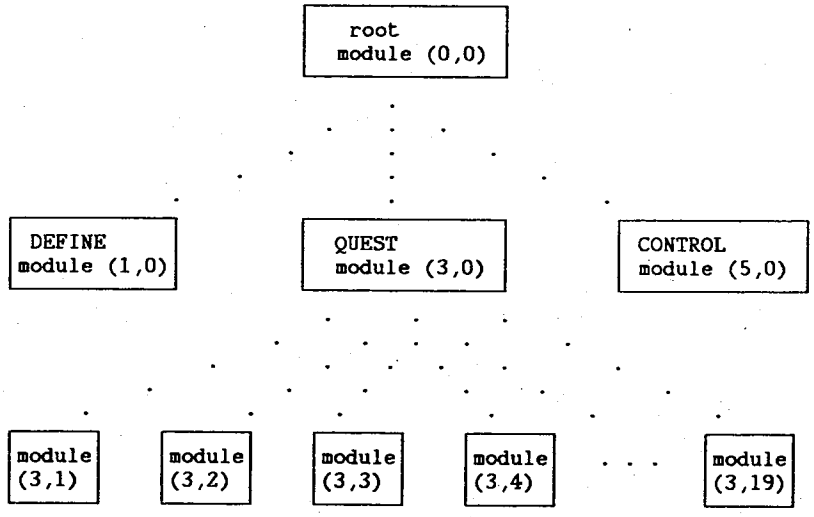
For example, the root module (0,0) must always have access to the CONTROL module (5,0). Therefore, with the root module in the root segment, the CONTROL module must either be in the root segment or in a primary segment. It cannot be in a secondary segment.

The root module (0,0) of SYSTEM 2000 software is the command monitor. It must belong to the root segment because all SYSTEM 2000 commands invoke the root module. The root module routes each command to the appropriate processor module, DEFINE (1,0), CONTROL (5,0), or QUEST (3,0). The processor module processes the command and then returns control to the root module.

The QUEST processor has several subprocessor modules that are controlled and called by the QUEST module (3,0). The DEFINE and CONTROL processor modules do not have any subordinate modules.

This internal flow of control is shown in the following illustration. A flow of control path must always exist from the root module to the DEFINE, QUEST, and CONTROL processor modules. In addition, a flow of control path must always exist from the QUEST processor module (3.0) to all the QUEST subprocessor modules (3,1) through (3,19).

Illus. 7.2 SYSTEM 2000 Internal Flow of Control



The root module can call each processor module, that is, (1,0), (3,0), and (5,0). Therefore, these processor modules must either belong to the root segment or to one or more of the primary segments. They cannot be configured into a secondary segment. You can place the QUEST subprocessor modules in either the same segment as the QUEST module (3,0) or in a different segment that is callable by module (3,0).

If the QUEST module (3,0) is in the root segment, all the QUEST subprocessor modules must belong to either the root or a primary segment. If the QUEST module is in a primary segment, all the QUEST subprocessor modules can belong to either that primary segment or a secondary segment. The former situation, however, eliminates the possibility of assigning any modules to a secondary segment. Placing the QUEST module (3,0) carefully is crucial to the configuration of SYSTEM 2000 software, because its placement determines where the QUEST subprocessor modules can be placed.

When OV=FULL or OV=FLAT, the configuration of the module and segment structure is determined by using the other PRELNK parameters. In this case, SYSTEM 2000 software does not allow an invalid structure (for example, one in which a flow of control path cannot be established) to be configured from the PRELNK parameters. However, when OV=GROUP, the SYSTEM 2000 configuration is determined by the input parameters that you specify for the GROUP macro. If you specify an invalid configuration in the GROUP parameters, an appropriate message appears.

Executable SYS2K Configuration

When Multi-User is configured (MU=YES), both SCF and PLEX are included when PRELNK is executed; the LANG parameter is ignored. The SYS2K configuration generated will serve all Multi-User jobs and single-user SCF jobs. For single-user PLEX jobs, see the next topic.

Single-User PLEX Configuration

To configure the system for single-user PLEX jobs, execute the PRELNK macro with MU=NO and LANG=PLI. Single-user PLEX jobs execute with the generated S2KPLI. Single-user SCF jobs execute SYS2K (see the previous topic).

Overlay Contention

In a Multi-User environment, many users share the various SYSTEM 2000 modules. Overlay contention can occur if you have not planned the overlay configuration carefully. Regardless of the number of threads specified for the site configuration, throughput of multiple jobs can be severely degraded -- to the extent of forcing jobs to wait for segment overlays in "single-thread fashion." Adding more threads does not solve this problem.

The ideal solution to segment overlay contention is a completely non-overlayed (FLAT) configuration. That is, all modules are in the root segment. This arrangement makes every module available to all users, but it uses the maximum amount of memory. The flat configuration serves well at most sites since SYSTEM 2000 software uses virtual memory.

The other extreme is a completely overlayed (FULL) configuration that uses minimum memory but significantly raises the risk of overlay contention. Between these two extremes, modules can be grouped into various segments as long as a flow of control path exists. The choice of configuration depends on available memory and the tasks requested by the users. Trade-offs to be considered include memory requirements, overlay swap time, and the job mix of SYSTEM 2000 functions to be performed in the Multi-User environment.

Consider the following examples. They show some of the many situations possible in which several users are contending for use of various modules. Remember that true multi-threading cannot take place unless the appropriate functional modules of SYSTEM 2000 software are available in the overlay segments.

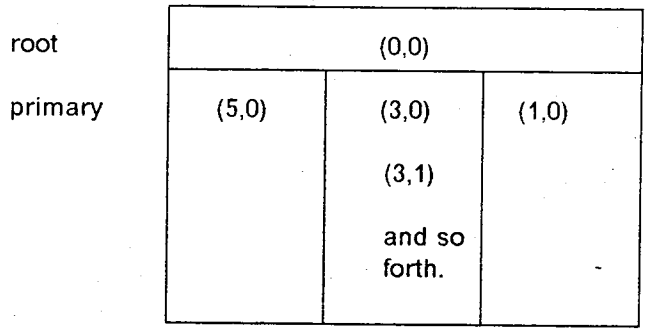
Example 1

The CONTROL processor (5,0) is not in the root segment. Therefore, if one or more users are retrieving or updating with the QUEST processor (3,0), a new user must wait to sign on, because (5,0) has been swapped out of memory. Also, any user wanting to use the DEFINE processor (1,0) must wait until the other users' retrievals and updates are complete.

The GENERATE command is processed by module (3,15). This overlay is locked for the duration of the report generation. Therefore, in a Multi-User environment, other users must wait to sign on until a report is finished. They must wait because the CONTROL processor (5,0) is used for sign-on and because (5,0) and (3,0), with its associated (3,15), are configured in separate primary segments. This could happen with multiple threads or single-thread systems under Multi-User, and it would cause unnecessary delay.

To avoid this situation, set up the overlay configuration with (3,0) and (5,0) in the same segment.

Illus. 7.3 CONTROL Processor (5,0) Not in Root Segment

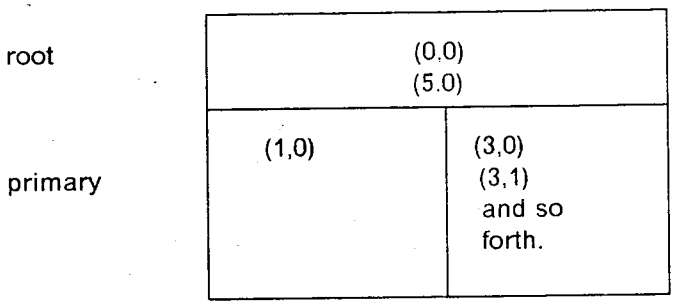


Example 2

The CONTROL processor (5,0) resides in the root segment and is always available to all users. This configuration restores or saves databases without overlay contention. Also, a new user can sign on at any time with no conflict, because the root segment containing (5,0) is always in memory.

However, requests for the DEFINE processor (1,0) must wait until all retrievals and updates using the QUEST processor (3,0) are complete before the primary segment containing DEFINE processor (1,0) can be loaded. Likewise, a user defining a lengthy new database definition uses (1,0) and causes all retrieval and updating jobs to wait in their threads (if the multiple threads are specified) until the definition is completed. Then the QUEST processor (3,0) can be loaded.

Illus. 7.4 CONTROL Processor (5,0) in Root Segment



Example 3

Another configuration that could force single-thread processing is one in which (3,7) and (3,17) swap each other out of memory. SYSTEM 2000 sort routines are in (3,7), and the PLEX syntax module is in (3,17). If several PLEX programs and SCF jobs require sort processing, contention between (3,7) and (3,17) must be avoided in order to achieve true multi-threading. Each PLEX command, except for load mode inserts, must be processed by (3,17). If a prolonged sort in (3,7) locks (3,17) out of memory, PLEX commands cannot be processed.

To avoid this contention, place modules (3,7) and (3,17) in the same segment.

Example 4

This example shows a method of avoiding excessive wait time during updates.

If several users are requesting lengthy updates composed of long LOAD/TERMINATE or queue mode update sessions, contention among modules (3,6), (3,7), and (3,8) (which update the database tables) could cause excessive wait times.

You can group these three modules into one segment with little impact on memory requirements. Depending on resources available, you can also place (3,6), (3,7), and (3,8) in a primary segment with (3,0). This arrangement frees secondary level overlay space for retrieval modules or sorts, depending on the job mix for the session.

SYSTEM 2000 LINKAGE MACRO INSTRUCTIONS

SYSTEM 2000 linkage macro instructions (PRELNK, GROUP, and ENDLNK) allow configuring the software to unique installation requirements. When the macro instructions are properly assembled, they produce linkage-editor input consisting of the S2KSEGTB CSECT and the linkage-editor directives necessary to configure the software according to PRELNK specifications. All releases of SYSTEM 2000 software must be configured using these linkage macro instructions.

PRELNK, GROUP, and ENDLNK Overview

The PRELNK macro and its optional associated macro instructions, GROUP and ENDLNK, produce any valid configuration of SYSTEM 2000 modules that you specify. The PRELNK macro defines the type of system being linked and allows a limited means of configuring the system. You have greater flexibility when you use the GROUP macro instruction. The ENDLNK macro instruction is used in combination with one or more GROUP macros to indicate that the GROUP definitions have been completed.

If you can specify a satisfactory configuration for your site by using only the PRELNK macro, then do not include the GROUP and ENDLNK macro instructions in the assembly. For specific details about the PRELNK, GROUP, and ENDLNK macros, see subsequent topics.

END Assembler Directive

The END assembler directive must always be the last record in any input containing the PRELNK macro and its optional associated macro instructions, GROUP and ENDLNK. The END directive consists only of the word 'END', starting in column 10.

THE PRELNK MACRO INSTRUCTION

You must use the PRELNK macro to specify the configuration of SYSTEM 2000 software at your installation. This section gives the syntax for the PRELNK macro statement parameters and shows the generated PRELNK CSECT.

Examples of typical SYSTEM 2000 configurations that can be generated using the PRELNK macro are also shown in this section. **The GROUP and ENDLNK Macro Instructions** on page 7-19 illustrates some common configurations that can be generated using the PRELNK macro OV=GROUP parameter and the GROUP and ENDLNK macro statements.

PRELNK Macro Statement Parameters

The PRELNK macro statement parameters, their functions, and the allowable options are documented in Illus. 7.5 on page 7-10. Then, Illus. 7.6 on page 7-12 shows the PRELNK CSECT generated by assembling the PRELNK macro.

Consider the following rules when specifying parameters for PRELNK:

- For a single-user environment, there is no default for the LANG parameter. Either NL or PLI must be specified.

  LANG=PLI requests and allows GROUP specification of only the following members: the root, CONTROL, QUEST, WHERE, USTRUC, UDATA, UINDEX, APPLY, OPTLOAD, EXTPLI, NKWHERE, and QWHERE.

  LANG=NL requests and allows GROUP specification of all members except RWRD, RWDS, and RWRG. (The RW=YES operand must be coded to include these.)

  If you specify MU=YES, the LANG parameter is ignored. Multi-User is configured with both the Self-Contained Facility (LANG=NL) and PLEX (LANG=PLI). The generated SYS2K configuration can also be used by single-user SCF jobs. To generate S2KPLI for single-user PLEX jobs, you must execute PRELNK with MU=NO and LANG=PLI.

- GROUP specification of all members is allowed when MU=YES.

- RW=YES must be specified to include the REPORT processor modules.

- If Rollback is not specified, then S2KRBX must be included in (0,0) to satisfy external references. If Rollback is specified, these members must be included: S2KRB0, S2KRB3, and S2KRB5.

Illus. 7.5 PRELNK Macro Parameters

```
Col 10    Col 16                                                    Col 72

PRELNK    OV=  |FULL ,     RW=   |NO ,      RB=   |NO ,     continuation-
               |FLAT               |YES             |YES    character
               |GROUP

          NAME= name,      MU=   |NO ,      C2K=  |NO ,
                                 |YES             |YES

          LANG=  |NL ,     QA=(module-list),  UE=   |NO
                 |PLI                               |YES
```

OV=FULL|FLAT|GROUP — specifies the configuration to be used in this generation of SYSTEM 2000 software.

FULL produces the most compact memory-saving configuration, which minimizes memory use but maximizes the number of overlay calls required during SYSTEM 2000 execution. If OV is not specified, FULL is the default.

FLAT collapses the segment overlay structure of SYSTEM 2000 software into a single load module. This option creates a copy of the software that uses maximum memory but does not call overlays during execution.

GROUP specifies a very tailored configuration through the use of subsequent GROUP and ENDLNK macro statements. This option offers you selective tradeoffs between memory usage and overlay swapping.

RW=YES|NO — YES supports the REPORT processor. This parameter is available only for LANG=NL or MU=YES.

NO does not support the REPORT processor in this configuration. RW=NO saves approximately 6K bytes over RW=YES. If RW is not specified, NO is the default.

RB=YES|NO — YES supports Coordinated Recovery in this configuration.

NO does not support Coordinated Recovery processing. If Rollback is not specified, NO is the default.

continued on next page

Illus. 7.5 *continued*

NAME=*name*	is for comments only. The specification of the load module name to be used in link-editing SYSTEM 2000 software must be accomplished through the SYSLMOD DD statement of the linkage-editor job step. The procedure for link-editing the software provides a default name of SYS2K. This default must be overridden with the name S2KPLI when the PLEX load module is link-edited.
MU=YES|NO	YES supports the Multi-User environment. The resulting load module includes all SCF and PLEX modules. When MU=YES, the LANG parameter is ignored. NO does not support Multi-User in this configuration. If MU is not specified, NO is the default.
C2K=YES|NO	YES supports the CONTROL 2000 processor XDD (additions to the DEFINE processor (1.0)). This parameter is invalid for installations that do not have XDD. In this case, C2K=YES results in linkage-editor diagnostics. NO does not support the XDD processor. If C2K is not specified, NO is the default.
LANG=NL|PLI	NL produces a configuration for the Self-Contained Facility in a single-user environment. This parameter is ignored if MU=YES. There is no default for LANG; either NL or PLI must be specified. PLI supports PLEX in a single-user environment. This parameter is ignored if MU=YES. There is no default for LANG; either NL or PLI must be specified.
QA=(*module-list*)	is a list of module names (separated by commas) that are to be included in the SYSTEM 2000 root segment. If the module list contains the name of only one module, the enclosing parentheses are optional. For example, QA=QAEXIT means include the QAEXIT module in the root segment. Omit this parameter if modules, such as QAEXIT or the EXIT00 module S2KEXIN, are not to be linked into SYSTEM 2000 software. QASTAT cannot be specified in the module list and must be linked with the PLEX application program. There is no default for QA.

continued on next page

Illus. 7.5 *continued*

UE=YES|NO	YES includes the SYSTEM 2000 user-exit interface module S2EXIT in the SYSTEM 2000 root segment. If you specify UE=YES, user exits are always active regardless of the EXITS= execution parameter. If UE=NO, then you can dynamically load S2EXIT at execution time. (See the EXITS parameter in Chapter 6, "User Exits"). If UE is not specified, NO is the default.
continuation-character	is an X in column 72, which means that the PRELNK parameters are continued on the next statement. Continued parameters must begin in Column 16. A continuation character must not appear on the parameter statement just prior to the END statement.

Illus. 7.6 PRELNK CSECT Generated

CSECT Name	Purpose	Conditions
S2KSEGTB	gives data to SYSTEM 2000 software at execution time regarding the system configuration selected.	unconditional

PRELNK Examples

This discussion shows some common PRELNK specifications and the resulting overlay structures. The diagrams represent SYSTEM 2000 modules by their module numbers. (See list in Illus. 7.1 on page 7-3.) Member modules boxed together belong to the same segment. Segments and their module members appear top to bottom in their actual order.

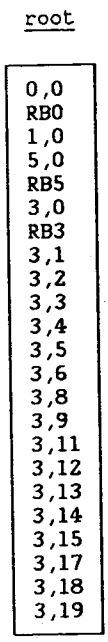
Example 1

The configuration structure below shows maximum memory usage for a non-overlayed (flat) SYS2K version of SYSTEM 2000 software, with Report Writer and Rollback enabled.

Illus. 7.7 Flat Non-Overlayed SYS2K Module with Coordinated Recovery Processor and Report Writer

```
Col 10                                                    Col 72

PRELNK OV=FLAT,MU=YES,RW=YES,                             X
       RB=YES,NAME=SYS2K
END
```



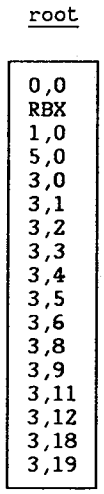
Example 2

This configuration structure, specified as OV=FLAT, uses maximum memory and minimum overlay swapping. The non-overlayed SYS2K load module shown below includes PLEX but not the REPORT or Coordinated Recovery processors.

Illus. 7.8 Flat Non-Overlayed SYS2K Module without Coordinated Recovery

```
Col 10

PRELNK OV=FLAT,MU=YES,NAME=SYS2K
END
```



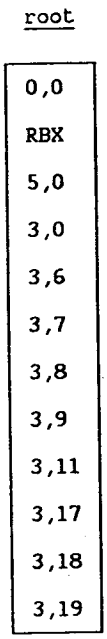
Example 3

This example shows a configuration that uses maximum memory when OV=FLAT is specified for a non-overlayed S2KPLI load module. This structure does not include Coordinated Recovery.

Illus. 7.9 Flat Non-Overlayed S2KPLI Module

```
Col 10

PRELNK OV=FLAT,LANG=PLI
END
```



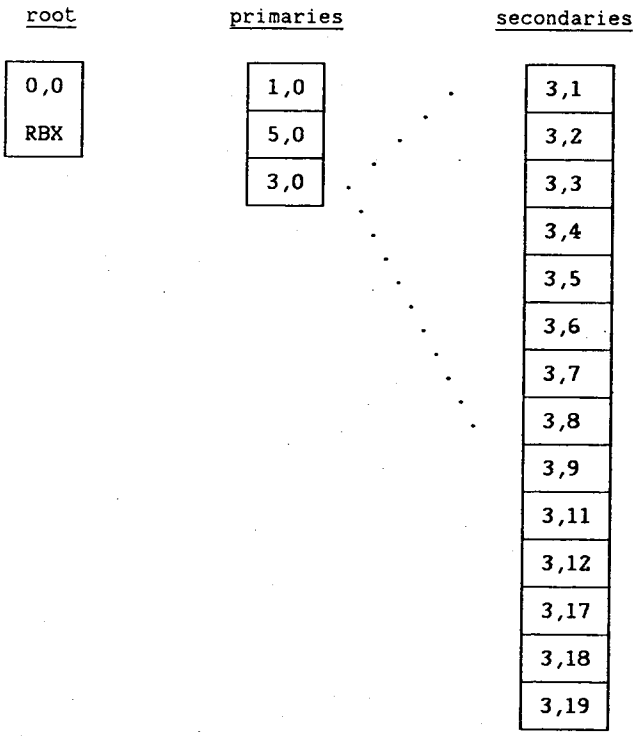
Example 4

This example shows the structure for a fully overlayed SYS2K load module. It has minimum memory requirements and maximum overlay calls. It also includes PLEX, but not the REPORT or Coordinated Recovery processors.

Illus. 7.10 Fully Overlayed SYS2K Module

```
Col 10

PRELNK OV=FULL,MU=YES,NAME=SYS2K
END
```



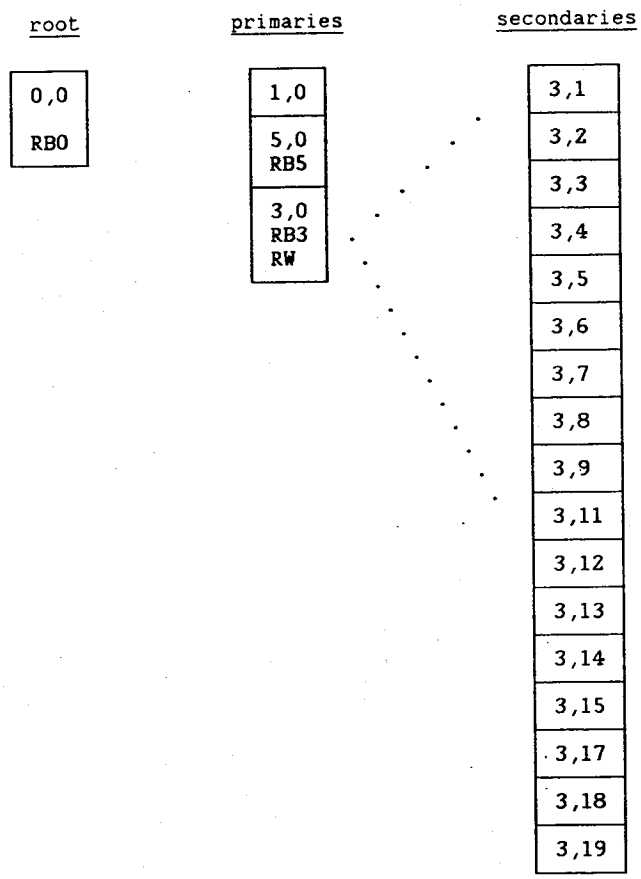
Example 5

This structure is for a fully overlayed SYS2K configuration, including the Coordinated Recovery and REPORT processors.

Illus. 7.11 Fully Overlayed SYS2K Module with Coordinated Recovery and REPORT Processors

```
Col 10                                          Col 72

PRELNK OV=FULL,MU=YES,RB=YES,                   X
       RW=YES,NAME=SYS2K
END
```



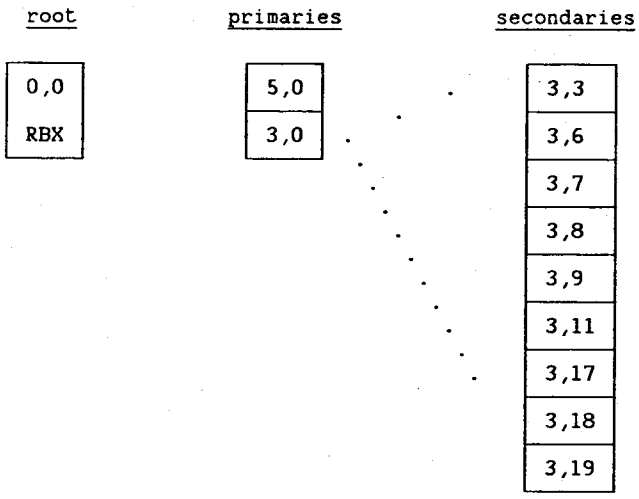
Example 6

This example shows a fully overlayed structure for the S2KPLI load module. It uses minimum memory and includes PLEX but not Coordinated Recovery.

Illus. 7.12 Fully Overlayed S2KPLI Module

```
Col 10

PRELNK OV=FULL,LANG=PLI
END
```



THE GROUP AND ENDLNK MACRO INSTRUCTIONS

The basic FULL and FLAT options of the PRELNK macro allow for only a general selection of configurations. To refine the configuration, you must use the GROUP and ENDLNK macros. In combination with the PRELNK macro, they allow you to tailor a SYSTEM 2000 configuration that has an effective tradeoff of memory versus I/O required for swapping overlays during SYSTEM 2000 execution.

The OV=GROUP parameter is required in the PRELNK macro when the GROUP macro is used.

GROUP Macro Statement Parameters

The GROUP macro statement parameters and their functions are shown in Illus. 7.13 on page 7-21. There are no defaults for the GROUP macro parameters.

Use the GROUP macro statements to position modules into segment overlays. Grouping modules into the same segment overlay helps to control and minimize segment overlay I/O during SYSTEM 2000 execution.

A separate segment is created for each GROUP macro occurrence. That is, for two modules to exist in the same segment, you must code both names in the same GROUP macro statement.

You can include modules other than the root module in the root segment only if the ROOT operand appears as the first operand in the GROUP macro statement on which the modules are specified. If the root module is not specified, it is placed in the root segment by itself.

If the DEFINE, CONTROL, and QUEST modules are not specified, they each appear in separate primary segments. If you want to group them with other modules, they do not have to appear as the first parameter in the statement. These three modules can be grouped together. However, the DEFINE and CONTROL modules cannot be grouped with any modules subordinate to QUEST unless QUEST is also included. See **SYSTEM 2000 Flow of Control** on page 7-4 for a discussion of SYSTEM 2000 flow of control.

By using the GROUP macro instruction, you can construct segment overlay configurations that have either one level (with all code in the root segment, as in OV=FLAT) or two levels (a root segment and a number of parallel primary segments). You can also construct an overlay configuration with three levels, composed of a root segment, a number of primary segments at the same level, and a number of secondary segments at the same level under one of the primaries, that is, (3,0).

If you place QUEST (3,0) in the root segment, then you can create only two levels in the overlay structure: the root segment and primary segments.

Col 10	Col 16	Col 72
GROUP	*(parameters)*	*continuation-character*

parameters is a list of one or more GROUP parameters separated by commas and enclosed in parentheses. If only one parameter is specified, the parentheses are optional. There are no default parameters.

continuation-character is an X in column 72 indicating that the GROUP parameters are continued on the next line. Parameters on a continuation line must begin in column 16. A continuation character must not appear on the parameter statement before the ENDLNK statement.

Illus. 7.13 GROUP Macro Parameters

Parameters	Function	Module
ROOT	General processing routines, operating system interface, and so forth	(0,0)
DEFINE	DEFINE processor	(1,0)
CONTROL	CONTROL processor (OPEN, SAVE, RESTORE, sign-on and so forth)	(5,0)
ACCESS	General processing routines unique to the QUEST processor	(3,0)
IALANG	QUEST language scanning	(3,1)
TALLY	TALLY/DESCRIBE command processing	(3,2)
WHERE	QUEST where-clause processor	(3,3)
PRINT	QUEST print/ordering processing	(3,4)
QLANG	QUEUE language scanning	(3,5)
USTRUC	Hierarchical Table updating	(3,6)
UDATA	Data Table updating and variable sort	(3,7)
UINDEX	Index Tables updating	(3,8)
WHERE	Fixed sort and SCF QUEUE where-clause processing	(3,9)
APPLY	APPLY command processing	(3,11)
MAP	MAP with restructuring and RELOAD processing	(3,12)
RWRD	REPORT language syntax analyzing	(3,13)
RWDS	REPORT processor record building	(3,14)
RWRG	REPORT processor report generating	(3,15)
EXTPLI	PLEX processor	(3,17)
OPTLOAD	PLEX load mode processing	(3,18)
NKWHERE	Where-clause processor (non-key)	(3,19)

ENDLNK Macro Statement

The ENDLNK macro statement appears only once in the input for a PRELNK job using the OV = GROUP operand. ENDLNK marks the end of the GROUP definitions. The GROUP macro statement just prior to the associated ENDLNK macro must not contain a continuation character (X in Column 72). The END assembler directive must be given immediately after the ENDLNK macro statement. Here is the ENDLNK macro specification:

```
Col 10

ENDLNK
```

GROUP/ENDLNK Macro Examples

Example 1

The GROUP macro in this example creates a system configuration identical to that created by Example 3 for the PRELNK macro in Illus. 7.9 on page 7-15. This structure is for a non-overlayed S2KPLI configuration.

Illus. 7.14 GROUP Macro for a Non-Overlayed S2KPLI Module without Coordinated Recovery

```
Col 10                                                    Col 72

PRELNK OV=GROUP,LANG=PLI
GROUP  (ROOT,                                             X
        CONTROL,                                          X
        ACCESS,                                           X
        WHERE,                                            X
        USTRUC,                                           X
        UDATA,                                            X
        UINDEX,                                           X
        QWHERE,                                           X
        APPLY,                                            X
        OPTLOAD,                                          X
        EXTPLI,                                           X
        NKWHERE)
ENDLNK
END
```

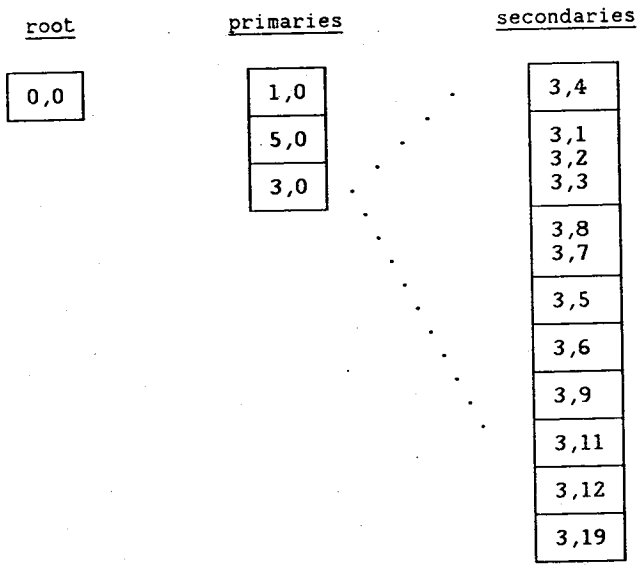
Example 2

This example illustrates the use of GROUP to create a configuration with multiple segments for QUEST retrieval and update routines. Notice that (3,0) is in the primary segment and that specifying PRINT alone in the first GROUP statement allows it to appear alone in the first secondary segment.

Illus. 7.15 GROUP Macro Configuration with Multiple Segments for QUEST Retrieval and Update

```
Col 10

PRELNK OV=GROUP,LANG=NL
GROUP (PRINT)
GROUP (IALANG,TALLY,WHERE)
GROUP (UINDEX,UDATA)
ENDLNK
END
```



Example 3

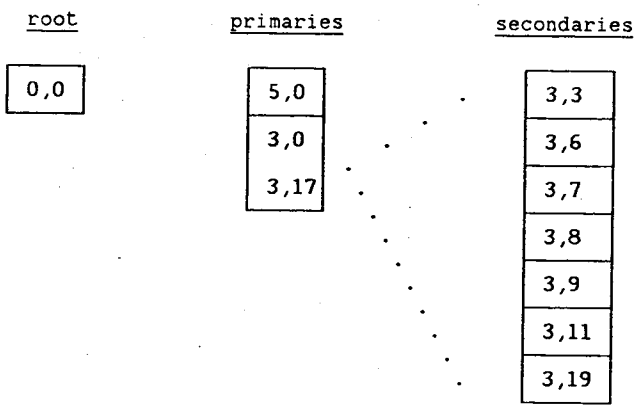
This example shows a SYSTEM 2000 configuration structure for efficient PLEX usage of one database. The interface module (3,17) is called for every PLEX command. Also, for efficiency, you can include (3,17) with (3,0).

This example includes modules to support all PLEX commands. The configuration is inefficient for using PLEX with more than one database, because the primary segment containing module (3,17) is swapped with the one containing module (5,0) each time a different database is referenced.

Illus. 7.16 Efficient PLEX Usage for One Database

```
Col 10

PRELNK OV=GROUP,LANG=PLI
GROUP (ACCESS,EXTPLI)
ENDLNK
```



SPECIAL CONSIDERATIONS

Executable Load Modules SYS2K and S2KPLI

SYS2K services both SCF and PLEX Multi-User requests and is loaded dynamically by the Multi-User monitor S2000. The Institute always link-edits S2000 into a single non-overlayed load module with entry point SUPS2K before delivering SYSTEM 2000 software to a customer site. Single-user SCF jobs execute SYS2K directly. The entry point of SYS2K must be S2K, whether it is executed by itself in its own region (single-user job) or is dynamically loaded by the Multi-User monitor, S2000.

S2KPLI is loaded dynamically into a single-user PLEX region at START S2K time. The entry point of S2KPLI must be S2KPLI. The PLEX START S2K command has an optional USE parameter that can be used to establish linkage to Multi-User software instead of loading the single-user PLEX module S2KPLI into the user region. S2KPLI is not used by Multi-User PLEX jobs. For details about the USE parameter, see the *SYSTEM 2000 PLEX Manual, Version 12, First Edition*.

The PRELNK macro is a valuable aid in assigning the right entry point to the right executable module. However, it cannot prevent certain user errors, because the linkage-editor NAME statement might not be specified in the same input stream as an object module. Even if it could be, the member name in the linkage-editor output library (DDNAME=SYSLMOD) always overrides the NAME statement. Consequently, you can ask PRELNK to construct a single-user PLEX module with the keyword LANG=PLI, which should be named S2KPLI. If the member name in the SYSLMOD DD statement is SYS2K, unpredictable results occur at run-time. In other words, you must assign the proper member name, because software such as PRELNK cannot assign or validate the member name.

For example, if you specify the PRELNK parameter MU=YES, the LANG parameter value is ignored because both SCF and PLEX use SYS2K (with entry point S2K) in the Multi-User environment. However, suppose you try to generate a Multi-User PLEX executable module by specifying MU=YES and LANG=PLI, and you assign S2KPLI to the member name in the SYSLMOD DD statement. PRELNK generates an ENTRY S2K linkage-editor input control statement because it assumes that SYS2K is being created. The result is a copy of S2KPLI with entry point S2K, causing unpredictable results at run time.

PRELNK is a valuable aid in configuring SYSTEM 2000 software. However, be careful to ensure that S2KPLI always has entry point S2KPLI and that SYS2K always has entry point S2K. This can be accomplished easily by always inspecting the linkage-editor output messages.

Field Fix Distributions and PRELNK

After applying SYSTEM 2000 problem fixes and field fix distributions, you must execute the PRELNK macro to recreate the SYS2K and S2KPLI executables.

Configuring for the REPORT Processor

To make the REPORT overlays available to single-user or Multi-User jobs. you must run PRELNK with RW=YES; this includes (3,13), 3,14), and (3,15) in the configuration structure.

To use the REPORT processor later, the RW execution parameter must be set to YES when users will be executing the software. The default is RW=NO, which saves approximately 9K bytes of work area per thread when users will not be using the REPORT processor. Any attempt to use the REPORT processor with RW=NO for single-user or Multi-User sessions results in SYSTEM 2000 Error Code 28, regardless of the link-edit structure. Also, SYSTEM 2000 Error Code 28 is issued if the PRELNK RW=NO and the execution parameter RW=YES. (For details, see the messages and codes discussion on this page.)

JCL FOR PRELNK EXECUTION

The JCL for executing PRELNK is listed in Illus. 7.17 on page 7-27. Note that the library containing PRELNK (S2K.R120.TEST on the delivery tape) must be specified first in the SYSLIB DD statement during the assembly step.

MESSAGES AND CODES FOR PRELNK

For information on messages and codes that can be issued when assembling the PRELNK macro, see *SYSTEM 2000 Messages and Codes, Version 12, First Edition*.

SYSTEM 2000 Error Code 28 SYSTEM 2000 Error Code 28 is issued any time you try to use a function that is not present in the executable load module. The code implies an incomplete link-edit of the software. Perhaps an optional feature was not installed or the link-edit failed to include all the necessary routines for the requested function.

In addition, any attempt to use the REPORT processor with the RW execution parameter set to NO results in a SYSTEM 2000 Error Code 28, regardless of the link-edit structure.

Illus. 7.17 JCL for Executing PRELNK

```
//JCLS2KLN JOB (ACTINFO),
//    USERID,CLASS=A
//*
//*****************************************************************
//* RELINK EXECUTABLE MODULES SYS2K AND S2KPLI
//*****************************************************************
//*
//S2KLINK  PROC OUT=A,WRKUNIT=SYSDA
//ASM      EXEC PGM=IEV90,
//             PARM='OBJECT,NODECK,XREF(SHORT)'
//SYSLIB   DD   DSN=S2K.R120.TEST,DISP=SHR
//SYSUT1   DD   UNIT=&WRKUNIT,SPACE=(1700,(600,100))
//SYSUT2   DD   UNIT=&WRKUNIT,SPACE=(1700,(300,30))
//SYSUT3   DD   UNIT=&WRKUNIT,SPACE=(1700,(300,50))
//SYSPRINT DD   SYSOUT=&OUT
//SYSLIN   DD   DSN=&&DECK,UNIT=&WRKUNIT,
//             SPACE=(80,(200,50)),DISP=(NEW,PASS)
//LKED     EXEC PGM=IEWL,
//    PARM=(MAP,XREF,LET,LIST,NCAL,'SIZE=(256K,64K)',
//          'AMODE=31','RMODE=ANY')
//SYSLMOD  DD   DSN=S2K.R120.LOAD(&LNAME),DISP=OLD
//SYSLIN   DD   DSN=&&DECK,DISP=(OLD,DELETE)
//LOAD     DD   DSN=S2K.R120.LOAD,DISP=SHR
//SYSPRINT DD   SYSOUT=&OUT
//SYSUT1   DD   UNIT=&WRKUNIT,SPACE=(1024,(400,20))
//         PEND
//*
//*****************************************************************
//* RELINK EXECUTABLE MODULES SYS2K AND S2KPLI
//*****************************************************************
//*
// EXEC S2KLINK,LNAME=SYS2K
//ASM.SYSIN DD *
      TITLE 'LINK EDIT SYSTEM 2000 FLAT SYSTEM FOR SCF'
      PRELNK OV=FLAT,RW=YES,LANG=NL,RB=YES,MU=YES,QA=QAEXIT
      END
// EXEC S2KLINK,LNAME=S2KPLI
//ASM.SYSIN DD *
      TITLE 'LINK EDIT SYSTEM 2000 FLAT SYSTEM FOR PLEX'
      PRELNK OV=FLAT,RB=YES,LANG=PLI,MU=NO,RW=NO
      END
//*
```

Appendix A

Supplementary PLEX Information

This appendix contains information you may need when using PLEX.

- **S2KDUM Control Block** on page A-2 describes the S2KDUM control block.
- **COMMBLOCK and SUBSCHEMA Control Blocks** on page A-8 describes the COMMBLOCK and SUBSCHEMA control blocks.
- **Operation Codes for PLEX Commands** on page A-13 lists the operation codes for all PLEX commands.
- **Execution-Time PLEX Exits** on page A-15 discusses execution-time PLEX exits.

S2KDUM CONTROL BLOCK

The names supplied for the S2KDUM control block fields are internal SYSTEM 2000 names. Different names are assigned by each PLEX processor. Illus. A.2 on page A-5 through Illus. A.5 on page A-7 show the expansion of the S2KDUM control block for each PLEX processor.

Illus. A.1 S2KDUM Control Block Format

Offset Hex	Offset Dec	Name	Length	Type/ Value	Field Description
0	0	S2KOPR	4	Binary	SYSTEM 2000 operation code; set by expanded PLEX code at run time
4	4	S2KRTC	4	Binary	SYSTEM 2000 return code; set by a PLEX command execution
8	8	S2KCNT	4	Binary	Value of S2KCNT or S2KCOUNT; set by PLEX program
C	12	S2KLOC	4	Binary	Value of stack subscript or Locate File subscript; set by expanded PLEX code at run time; includes subscript of stack/file to be copied by COPYS/COPYL
10	16	S2KMAX	4	Binary	Value of maximum subscript for Locate File; used by START S2K statement, set by expanded PLEX code at run time; subscript of stack/file to be created by the COPYS/COPYL, set by expanded PLEX at run time
14	20	S2KTYP	4	Binary	PLEX program type; initialized by PLEX processor
				1	FORTRAN
				2	COBOL
				3	PL/I optimizing or check out
				4	PL/I (F)
				5	Assembler
18	24	S2KUSE	4	Binary	SYSTEM 2000 indicator; set by expanded PLEX code and used by S2KPLR at START S2K time to decide which environment to run the software in
				0	single user - S2KPLI
				1	Multi-User - MUPLINT

continued on next page

Illus. A.1 *continued*

Offset Hex	Offset Dec	Name	Length	Type/ Value	Field Description
1C	28	S2KUSE1	4	Alphameric	If S2KUSE is neither fullword binary zero (0) nor one (1), then S2KUSE concatenated with S2KUSE1 contains the name of a special module to be used (left-aligned and blank-filled). S2KPLR loads this module instead of S2KPLI or MUPLINT.
20	32	S2KPGM	8	Alphameric	Name of the program or subroutine (left-aligned and blank-filled) containing S2KDUM with initial values, set by the PLEX processor. In COBOL and FORTRAN, this name is derived from a program statement or overridden by the $NAME directive. In PL/1, this name is specified by the $NAME directive.
28	40	S2KVER	4	Alphameric	Release number of SYSTEM 2000 PLEX processor used and set by PLEX processor. If changes are made to the Program Service Processor, S2KVER is used to identify which PLEX processor release was used for the program.
2C	44	S2KDTE	8	Alphameric	Date of PLEX processor run (YYDDD, left-aligned and blank-filled); set by PLEX processor
34	52	S2KTIM	8	Alphameric	Time of PLEX processor run (HHMMSSS, left-aligned and blank-filled); set by PLEX processor
3C	60	S2KVS2	4	Alphameric	Release number of SYSTEM 2000 DBMS supplied at runtime by SYSTEM 2000 DBMS
40	64	S2KZRO	4	Binary	Reserved for internal use
44	68	S2KALL	4	Binary	Reserved for internal use
48	72	S2KPOS	4	Alphameric	Reserved for internal use
4C	76	S2KPLR	4	Binary	Address of run-time interface; set by S2KPL

continued on next page

Illus. A.1 *continued*

Offset Hex	Offset Dec	Name	Length	Type/ Value	Field Description
50	80	S2KS2K	4	Binary	Address of SYSTEM 2000 DBMS or Multi-User PLEX interface; set by S2KPLR
54	84	S2KSTR	4	Binary	Address of working storage; set by run-time interface. This address can be used by installation run-time exit routines S2KEX1 or S2KEX2.
58	88	S2KSNP	4	Binary	Snap request indicator; modified by expanded PLEX code
				0	off
				1	on, provides snapshot data
5C	92	S2KTME	4	Binary	Timing request indicator; modified by expanded PLEX code
				0	off
				1	on, provides TIMING data
60	96	S2KSRT	4	Binary	No-sort request indicator; set by PLEX program
				0	sort normally
				1	do not sort
64	100	S2KIFSV	4	Binary	Reserved for internal use.
68	104	S2KFIL	4	Binary	Reserved for internal use.
6C	108	S2KFLG	1	Bit	Switches set and used by runtime interface S2KPLR; initialized to 0 by PLEX processor
				0	link history not modified
				1	link history has been modified
6D	109		3		Reserved for internal use.
70	112	S2KLKH	40	Binary	Ten fullwords representing LINK0 through LINK9 return codes; set by S2KLR for linked retrievals; initialized by PLEX processor to ten words of binary minus one (-1)

Illus. A.2 S2KDUM Control Block Expansion in COBOL

```
01  S2KDUM
    02 S2KDU1      PIC S9(9) USAGE COMP VALUE +0.
    02 S2KRTC      PIC S9(9) USAGE COMP VALUE +0.
    02 S2KCOUNT    PIC S9(9) USAGE COMP VALUE +0.
    02 S2KDU2      PIC S9(9) USAGE COMP VALUE +0.
    02 S2KDU3      PIC S9(9) USAGE COMP VALUE +0.
    02 FILLER      PIC S9(9) USAGE COMP VALUE +2.
    02 S2KDU4      PIC S9(9) USAGE COMP VALUE +0.
    02 FILLER      PIC S9(9) USAGE COMP VALUE +0.
    02 FILLER      PIC X(8) VALUE 'TEST'.
    02 FILLER      PIC X(4) VALUE '12.0'.
    02 FILLER      PIC X(8) VALUE '85204   '.
    02 FILLER      PIC X(8) VALUE '2137524 '.
    02 FILLER      PIC X(4) VALUE '    '.
    02 S2KDU5      PIC S9(9) USAGE COMP VALUE +0.
    02 S2KDU6      PIC S9(9) USAGE COMP VALUE -1.
    02 S2KDU7      PIC X(4) VALUE '    '.
    02 FILLER      PIC S9(9) USAGE COMP VALUE +0.
    02 FILLER      PIC S9(9) USAGE COMP VALUE +0.
    02 FILLER      PIC S9(9) USAGE COMP VALUE +0.
    02 S2KDU8      PIC S9(9) USAGE COMP VALUE +0.
    02 S2KDU9      PIC S9(9) USAGE COMP VALUE +0.
    02 S2KSRT      PIC S9(9) USAGE COMP VALUE +0.
    02 S2KDU11     PIC S9(9) USAGE COMP VALUE +0.
    02 S2KDU12     PIC S9(9) USAGE COMP VALUE +0.
    02 FILLER      PIC S9(9) USAGE COMP VALUE +0.
    02 S2KLKH      PIC S9(9) USAGE COMP OCCURS 10 TIMES.
```

Illus. A.3 S2KDUM Control Block Expansion in PL/1

```
DCL 1 S2KDUM
      2 S2KDU1      FIXED BIN(31) INIT (0),
      2 S2KRTC      FIXED BIN(31) INIT (0),
      2 S2KCOUNT    FIXED BIN(31) INIT (0),
      2 S2KDU2      FIXED BIN(31) INIT (0),
      2 S2KDU3      FIXED BIN(31) INIT (0),
      2 S2K1        FIXED BIN(31) INIT (3),
      2 S2KDU4      FIXED BIN(31) INIT (0),
      2 S2K2        FIXED BIN(31) INIT (0),
      2 S2K3        CHAR(8)       INIT ('        '),
      2 S2K4        CHAR(4)       INIT ('12.0'),
      2 S2K5        CHAR(8)       INIT ('85207   '),
      2 S2K6        CHAR(8)       INIT ('2137524 '),
      2 S2K7        CHAR(4)       INIT ('    '),
      2 S2KDU5      FIXED BIN(31) INIT (0),
      2 S2KDU6      FIXED BIN(31) INIT (-1),
      2 S2KDU7      CHAR(4)       INIT ('    '),
      2 S2K8        FIXED BIN(31) INIT (0),
      2 S2K9        FIXED BIN(31) INIT (0),
      2 S2K10       FIXED BIN(31) INIT (0),
      2 S2KDU8      FIXED BIN(31) INIT (0),
      2 S2KDU9      FIXED BIN(31) INIT (0),
      2 S2KSRT      FIXED BIN(31) INIT (0),
      2 S2K11       FIXED BIN(31) INIT (0),
      2 S2K12       FIXED BIN(31) INIT (0),
      2 S2K13       FIXED BIN(31) INIT (0),
      2 S2KLKH(10) FIXED BIN(31) INIT ((10)-1);
```

Illus. A.4 S2KDUM Control Block Expansion in FORTRAN

```
 COMMON/S2KDUM/S2KDUM(38)
 INTEGER S2KDUM,S2KDU1,S2KRTC,S2KCNT,S2KDU2,S2KDU3,S2KDU4,S2KDU5,
-S2KDU6,S2KDU7,S2KDU8,S21KDU9,S2KSRT,S2KLKH(10)
 EQUIVALENCE (S2KDUM(1),S2KDU1),(S2KDUM(2),S2KRTC),(S2KDUM(3),
-S2KCNT),(S2KDUM(4),S2KDU2),(S2KDUM(5),S2KDU3),(S2KDUM(7),S2KDU4),
-(S2KDUM(17),S2KDU5),(S2KDUM(18),S2KDU6),(S2KDUM(19),S2KDU7),
-(S2KDUM(23),S2KDU8),(S2KDUM(24),S2KDU9),(S2KDUM(25),S2KSRT),
-(S2KDUM(29),S2KLKH(1))
```

Illus. A.5 S2KDUM Control Block Expansion in Assembler

```
Col       Col   Col
1         10    16
S2KDUM    DS    0CL152
S2KDU1    DC    F'0'
S2KRTC    DC    F'0'
S2KCNT    DC    F'0'
S2KDU2    DC    F'0'
S2KDU3    DC    F'0'
          DC    F'5'
S2KDU4    DC    F'0'
          DC    F'0'
          DC    C'        '
          DC    C'12.0'
          DC    C'85201   '
          DC    C'2137524 '
          DC    F'0'
S2KDU5    DC    F'0'
S2KDU6    DC    F'-1'
S2KDU7    DC    CL4' '
          DC    3F'0'
S2KDU8    DC    F'0'
S2KDU9    DC    F'0'
S2KSRT    DC    F'0'
          DC    3F'0'
S2KLKH    DC    10F'-1'
```

COMMBLOCK AND SUBSCHEMA CONTROL BLOCKS

The names supplied for the COMMBLOCK and SUBSCHEMA control block fields are internal SYSTEM 2000 names.

The formats of the date and time fields in the COMMBLOCK vary with the programming language used.

The 'S' and 'P' in the Set by column indicate whether values for the field are set by SYSTEM 2000 software or by the PLEX Processor.

Illus. A.6 COMMBLOCK Control Block Format

Offset Hex	Offset Dec	Name	Length	Type/ Value	Set by	Field Description
		The following fields represent the COMMBLOCK prefix.				
0	0	FCNAM	16	Alphameric	P	Database name, can be modified by PLEX program before open.
10	16	FCACTV	4	Binary 0 ≠0	P	Database active Inactive Value of SYSTEM 2000 internal database identification
14	20	FCLNTH	4	Binary		Length in bytes of COMMBLOCK including the prefix initialized by the PLEX processor to include or exclude CPUP as the last field.
18	24	FCFOR	4	Binary		FOR BLOCK chain Pointer to current FOR BLOCK initialized to binary zero by the PLEX processor and modified by S2KPLR.
		The following fields represent the variables described in the COMMBLOCK section of the *SYSTEM 2000 PLEX Manual, Version 12, First Edition*.				
1C	28	FCSCHM	32	Alphameric	S	Subschema record name The name of the last action subschema record referenced except when certain non-zero return codes occur. May contain the name of the item that caused the command to be rejected.
3C	60	FCRTC	4	Binary	S	Return code

continued on next page

Illus. A.6 *continued*

Offset Hex	Offset Dec	Name	Length	Type/ Value	Set by	Field Description
40	64	FCUSER	4	Binary		Available to the user; initialized by PLEX processor to binary zero
44	68	FCLAST	4	Binary 0 1	S	Last data record indicator. One or more data records remain. This is the last data record.
48	72	FCPASS	4	Alphameric		Password set by PLEX program; must be left-aligned and blank-filled
4C	76	FCNRG	4	Binary	S	Number of data records selected by a GET1 or LOCATE command
50	80	FCPOS	4	Binary	S	Relative position of the last data record accessed from a Locate File
54	84	FCLEV	4	Binary	S	Hierarchical level of the last data record accessed
58	88	FCTIME	8	Alphameric	S	Time of last update HH:MM:SS
60	96	FCDATE	8	Alphameric	S	Date of last update MM/DD/YY
68	104	FCCYC	4	Binary	S	Last database cycle number
6C	108	FCSEP	4	Alphameric	S	System separator right-aligned, binary zero filled
70	112	FCENT	4	Alphameric	S	Entry terminator word left-aligned, blank-filled
74	116	FCSTAT	4	Binary 0 1	S	Database status at open time Undamaged Damaged (updates not allowed)
78	120	FCCPUP	4	Binary	S	Optional variable indicating the parent of the last data record retrieved

Illus. A.7 SUBSCHEMA Control Block Format

Offset Hex	Offset Dec	Name	Length	Type/ Value	Set by	Field Description
0	0	SCBNAM	30	Alphameric	P	Name of subschema record If SCBNUM=0, the name must be the name or C-number of a schema record
1E	30	SCBRSV	2	Alphameric		Reserved for internal use
20	32	SCBBUF	4	Binary	P	Length in bytes of the subschema record I/O area following the last ECB
24	36	SCBINT	4	Binary	S	Internal number of the schema record represented by this subschema record; initialized to binary zero
28	40	SCBLEV	4	Binary	S	Internal level number of schema record initialized to binary zero
2C	44	SCBNUM	4	Binary	P	Number of ECBs that belong to this subschema record (number of items represented by this subschema record)
30	48	SCBDBI	4	Binary	S	Internal values of the last database used with this subschema record initialized to binary zero
34	52	SCBACT	1	Bit		Activity indicator;
				.... 1...		Subschema record control block identification; initialized by PLEX processor;
				....n		Use entire subschema record or check individual items; set by PLEX program or S2KPLR; n=0 Each ECB is to be checked to see if it should participate. n=1 All items in this subschema record are to participate.
				.... ..1.		Link command is active for this subschema record; zero if link is not active set by S2KPLR
35	53	SCBFIL	3	Binary		Reserved for internal use; initialized to binary zero.

continued on next page

Illus. A.7 *continued*

The following fields are repeated for each item in the subschema record.

Offset Hex	Offset Dec	Name	Length	Type/ Value	Set by	Field Description
0	0	ECBNAM	30	Alphameric	P	Name of item SYSTEM 2000 name or C-number of item
1E	30	ECBRSV	2	Alphameric		Reserved for internal use.
20	32	ECBBUF	4	Binary	P	Number of bytes displaced from beginning of current ECB to its position in the I/O area that immediately follows the last ECB in this subschema record
24	36	ECBINT	4	Binary	S	Internal SYSTEM 2000 number of the item; initialized to binary zero
28	40	ECBLEV	4	Binary	S	Internal SYSTEM 2000 number of the schema record level; initialized to binary zero
2C	44	ECBNUM	4	Binary	S	Internal SYSTEM 2000 number of the owner schema record; initialized to binary zero
30	48	ECBDBI	4	Binary	S	Internal value of the last database this ECB was used with; initialized to binary zero
34	52	ECBACT	1	Bit .... 0... ....n		Activity indicator; item identification; initialized by PLEX processor; item is or is not to participate checked only if corresponding SCBACT bit is zero; set by PLEX program. n=1 item is to participate n=0 item is not to be used

continued on next page

Illus. A.7 *continued*

Offset Hex	Offset Dec	Name	Length	Type/ Value	Set by	Field Description
35	53	ECBTYP	1	Binary	P	Item picture type
				0 (00)		alphanumeric
				1 (01)		alphabetic
				2 (02)		unsigned numeric - display code
				3 (03)		signed numeric - display code
				4 (04)		numeric - fullword binary
				5 (05)		numeric - halfword binary
				6 (06)		unsigned packed decimal
				7 (07)		signed packed decimal
				8 (08)		FORTRAN - picture is based on definition
				9 (09)		numeric - float single
				10 (0A)		numeric - float double
				11 (0B)		date - MMDDYYYY
				12 (0C)		date - MMDDYY
				13 (0D)		date - DDMMYYYY
				14 (0E)		date - DDMMYY
				15 (0F)		date - YYYYMMDD
				16 (10)		date - YYMMDD
				17 (11)		varying length alphanumeric string. First two bytes contain the current length of the string and are included in ECBTOT.
36	54	ECBRGT	1	Binary	P	The number of digits to the right of the decimal
37	55	ECBTOT	1	Binary	P	Total number of bytes in the field

OPERATION CODES FOR PLEX COMMANDS

SSR indicates a subschema record.

Illus. A.8 Opcodes for PLEX Commands

PLEX Command	OPCODE Decimal	Hex
GET1 *SSR* FIRST	00	00
GET1 *SSR* LAST	01	01
GET1 *SSR* S2KCOUNT	03	03
GET1 *SSR* NEXT	04	04
GET1 *SSR* WHERE *where-clause*	05	05
GET1 *SSR* DYNAMICALLY	07	07
GET1 *SSR* PRESENT	08	08
LOCATE *SSR* WHERE *where-clause*	10	0A
GETP *SSR* WHERE *single-key-condition*	11	0B
LOCATE *SSR* DYNAMICALLY	12	0C
GET *SSR* FIRST	20	14
GET *SSR* LAST	21	15
GET *SSR* PREVIOUS	22	16
GET *SSR* S2KCOUNT	23	17
GET *SSR* NEXT	24	18
GET *SSR* PRESENT	25	19
GETD *SSR* FIRST	30	1E
GETD *SSR* LAST	31	1F
GETD *SSR* S2KCOUNT	33	21
GETD *SSR* NEXT	34	22
GETD *SSR* PRESENT	36	24
GETA *SSR*	40	28
Reserved for future use	46	2E
Reserved for future use	47	2F
Reserved for future use	48	30
INSERT *SSR*	50	32
INSERT *SSR* AFTER	50	32
INSERT *SSR* BEFORE	51	33
MOVE TREE *SSR*	55	37
MOVE TREE *SSR* AFTER	55	37
MOVE TREE *SSR* BEFORE	56	38
OPEN *database*	60	3C
LINK	63	3F
OPENR *database*	64	40
LOAD *database*	65	41
FOR	67	43
REMOVE *SSR*	70	46
MODIFY *SSR*	80	50
REMOVE TREE *SSR*	90	5A
START S2K (*n*)	91	5B
STOP S2K	92	5C

continued on next page

Illus. A.8 *continued*

PLEX Command	OPCODE Decimal	OPCODE Hex
ORDER BY *ordering-clause*	93	5D
CLEAR	94	5E
CLEAR UPDATE LOG	-406	FFFF FE6A
CLEAR AUTOMATICALLY	95	5F
CLEAR UPDATE LOG AUTO	-405	FFFF FE6B
CANCEL QUEUE	96	60
CLOSE *database*	97	61
QUEUE	98	62
TERMINATE	99	63
APPLY	201	C9
KEEP	202	CA
Reserved for future use	203	CB
Reserved for future use	204	CC
RELEASE	205	CD
SUSPEND	206	CE
FULL PASSES/NO FULL PASSES	207	CF
ENABLE/DISABLE ROLLBACK	208	D0
FRAME	209	D1
END FRAME	210	D2
DROP HOLD	211	D3
RESET ROLLBACK	212	D4
SAVE	213	D5
RESTORE	214	D6
COPYS	215	D7
COPYL	216	D8
COMMIT	217	D9
ROLLBACK	218	DA
ENABLE/DISABLE VALUES PADDING	220	DC
ENABLE/DISABLE EXITS	240	F0

EXECUTION-TIME PLEX EXITS

Execution-time exits can be linked to S2KPLR. "Dummy" exits are provided in the standard, distributed copy of S2KPLR to avoid unresolved external references. These can be replace-linked with routines written at your site.

S2KEX0 is a fullword containing the number of additional bytes of memory (binary) required by the other two executable exits. S2KPLR is a reentrant module that does a GETMAIN to obtain new working storage for each new user. The exits must conform to reentrant requirements if S2KPLR is to retain its reentrancy. S2KPLR dynamically obtains memory for the exits as requested in S2KEX0. The supplied S2KEX0 has a value of zero. The address of the working storage obtained for the exits is stored in a fullword at offset 84 (X'54') of S2KDUM, where it can be accessed by S2KEX1 and S2KEX2.

S2KEX1 is given control through conventional MVS linkage before each command is passed to SYSTEM 2000 software. Register 1 contains the address of a list of addresses. The first address in the list is that of the S2KDUM vector, and the second is that of the COMMBLOCK (if any). Registers must be saved and restored according to standard MVS conventions.

S2KEX2 is given control after each command is processed by SYSTEM 2000 software. Conventions are the same as those for S2KEX1.

Appendix B

Converting Databases to Version 12 Format

INTRODUCTION

Databases created before Version 12 must be rebuilt with the new Version 12 format. Version 12 software cannot access database tables created prior to Version 12. Once you have converted a database to Version 12 format, the database files are not downward compatible with previous releases of SYSTEM 2000 software.

To convert your existing databases, use the CNVRT12 conversion program provided on the Release 12.0 delivery tape. CNVRT12 reads an existing Savefile and produces a Version 12 database that has larger internal pointers to accommodate more records and data.

If you want to rebuild your databases with Version 12 software, you can create the database definition with Release 12.0, using the same process as in earlier releases. If your definition is not already on a file, you can use the DESCRIBE/DEFINE/ command to extract the definition in DEFINE format. (For documentation of the DESCRIBE/DEFINE/ command, see *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition.*)

To load the data, you can use the SCF LOAD command or PLEX optimized load mode. The Release 12.0 delivery tape contains two programs (S2KGUNLD and S2KGLOAD) that generate COBOL PLEX programs for unloading and loading your database. Rebuilding the database eliminates reusable space and yields a freshly loaded, compact database.

THE CNVRT12 CONVERSION PROGRAM

The CNVRT12 conversion program reads a Savefile that was created before Version 12 and produces disk database files in Version 12 format. The sizes of internal address pointers are longer, which allows more data records and larger file sizes. Once a database is in Version 12 format, it cannot be accessed by previous releases of SYSTEM 2000 software. Similarly, Version 12 software cannot read, for example, Release 11.6 databases unless they are rebuilt with the new Version 12 format.

The CNVRT12 program

- runs in stand-alone batch mode, which allows you to convert individual databases offline as needed
- reads a Savefile as input
- allows you to change the block size of a saved database (except for File 3) with the REBLOCK option.

Note: You can also use the REBLOCK option to change the block size of a Version 12 database at any time.

To execute the CNVRT12 program, allocate the new Version 12 database files in the JCL job stream for the batch job. If the database files already exist, the conversion program issues an error message and stops processing.

Executing CNVRT12 takes slightly more time than processing the CONTROL language RESTORE command. The program reads a Savefile as input and writes the new database files. Therefore, after you run the program, you can still maintain two versions of the database until you install Release 12.0 software for production use.

How the CNVRT12 Program Works

The CNVRT12 program runs as a stand-alone batch job. It reads a Savefile and expects a parameter file with the DDname of CNVPARM. The parameter file specifies the names of the databases to be converted and whether you want to reblock the databases. Database files must be allocated in the JCL for the batch job so that the program can write the reformatted tables to them.

If the data sets designated as the database files in the JCL are not formatted, the CNVRT12 program formats them for you. Also, if the data sets do not have block sizes assigned at OPEN time (regardless of the REBLOCK option), the CNVRT12 program assigns block sizes as defined in the Savefile.

Note: The CNVRT12 program cannot convert Savefiles that were written prior to Release 10.1.

For conversion or for reblocking, all key items in the database are changed to non-key. Due to the intricate structures of the index tables, File 2 and File 4 (the Distinct Values Table and the Multiple Occurrence Table) are not converted directly by the program. During the conversion process, key items are treated as non-key items.

As each key item is changed to a non-key item, the program places a reference to the item in a temporary internal file. The program uses that file later to create a Command File, called KEYS. The KEYS file contains CREATE INDEX commands for all key items that were changed to non-key during conversion. After the conversion process completes, you can run an SCF job to create indexes for the key items in the Version 12 database. With this procedure, you have the opportunity of editing the KEYS Command File prior to executing it. You can either remove items from the key list or add items that were not key originally but now need to be key for your applications.

If the Update Log and the Rollback Log were enabled in the original Savefile, they are turned off during conversion. The program remembers whether either or both were enabled. After conversion, the program writes appropriate commands to the KEYS Command File to reactivate update logging and/or rollback. The SAVE/ command activates update logging if necessary and saves the Version 12 database on a dynamically allocated Savefile. If rollback was enabled on the original Savefile, the ENABLE ROLLBACK command is included in the KEYS Command File.

The Input Parameter File

The DDname of the input parameter file must be CNVPARM. You must supply this file to the CNVRT12 program. It contains the name of one or more databases to be converted in the batch job. For each database, you can specify the CONVERT option or the REBLOCK option or both, provided the database was created before Version 12. For Version 12 databases, you can specify only the REBLOCK option.

The syntax for the parameter entries is as follows:

database(option,option)

where

database is the name of an existing database that was saved on a Savefile.

option is the keyword CONVERT or REBLOCK. If you specify both, they can be in either order; separate them with a comma.

CONVERT indicates you want to convert a database created by Release 10.1, Release 11.0, Release 11.5, or Release 11.6. CONVERT is the default if you do not specify any options.

REBLOCK indicates you also want the database to be reblocked.

If the block sizes of the database files on the Savefile are different than the block sizes specified in the JCL, you must specify the REBLOCK option for the database, or an error message is issued.

Note: You can also reblock a Version 12 database with the CNVRT12 program by specifying the database name followed by the REBLOCK option, for example, SALES(REBLOCK). If you specify CONVERT for a Version 12 database, an error message appears.

Notice that the CNVPARM file can contain more than one database entry in a single run, which allows you to convert or reblock multiple databases in one run of the conversion program.

Sample Parameter Entries

The following parameter entries rebuild the EMPLOYEE, CARS, SALES, and FINANCE databases. The SALES database will also be reblocked according to the block sizes specified in your JCL. The last parameter entry can be specified to reblock a database that is already in Version 12 format.

```
EMPLOYEE
CARS(CONVERT)
SALES(CONVERT,REBLOCK)
FINANCE(REBLOCK)
```

Sample DDnames

Because the CNVRT12 program runs as a stand-alone batch job, you must define all files in the JCL submitted to run the job. The conversion program assumes that each database name supplied in a parameter entry is the basic DDname in the JCL. DDnames must use the standard SYSTEM 2000 naming conventions. That is, the six DDnames consist of the first seven characters of the database name with blanks replaced by Xs, followed by an integer from 1 to 6; they represent database Files 1 through 6.

Suppose you specify the following parameter entry:

```
EMPLOYEE(CONVERT)
```

When you specify the new Version 12 database files in your JCL, the DDnames must be as follows:

```
//EMPLOYE1
      :
      :
//EMPLOYE6
```

The KEYS Data Set

The KEYS DD statement in the JCL defines a sequential data set that will contain the Command File generated by the CNVRT12 program. It will contain SCF commands that create the key items in the Version 12 database. It also will contain commands that activate the Update Log and enable rollback, if these were activated on the original Savefile.

Here is a sample KEYS data set. Four items will become key items. Also, the Update Log and the Rollback Log are reactivated.

```
CONTROL:
DBN IS EMPLOYEE:
CONTROL:
CREATE INDEX C1,C2,C3,C101:
PRINT SIZE:
ENABLE ROLLBACK:SAVE/:
QUEST:
COMMAND FILE IS INPUT:
```

The PRINT SIZE command displays the number of pages in use for each database file after the conversion and after the indexes are recreated. Some of the files may be larger than the original files due to the longer pointers in Version 12.

You can edit this data set prior to executing it as a Command File for a batch SCF session. The commands rebuild the index tables in the Version 12 database using the keys from the original database; you might want to add more key items or remove some according to existing application needs. Notice that the last command on the file returns to INPUT, which allows you to include additional SCF commands in the job.

Note: This Command File does not begin with a USER command, which would contain the master password. You must start the SYSTEM 2000 session from an input Command File that gives the USER command, then switches to the KEYS Command File. For example,

```
USER,ABLE:
COMMAND FILE IS KEYS:
```

Or you could edit the appropriate master password and USER command into the KEYS Command File before you submit it.

Executing the CNVRT12 Program

To convert or reblock a database with the CNVRT12 program, follow these steps:

1. Allocate the Version 12 database files, File 1 through File 6, in your JCL. If you do not specify block sizes, the program uses the Savefile block sizes. If you want new block sizes for the Version 12 database files, specify the new block sizes(s) in the DD statement(s) for the appropriate files and include the REBLOCK option in the parameter entry for the database.

2. Allocate a data set to be used for the output Command File. The DDname must be KEYS.

3. Establish the parameter entries for the program. Place the parameter entries in a data set having the DDname CNVPARM. Each entry contains the name of a database to be converted or reblocked, followed by the CONVERT or REBLOCK options where appropriate. You can convert one or more databases in one job execution.

4. Execute the CNVRT12 program.

5. Execute an SCF job to execute the KEYS Command File. If necessary, edit the file before you run the job.

Sample JCL

This sample JCL shows how to convert and reblock the EMPLOYEE database. It includes the REBLOCK option and assumes that the new block sizes were specified when the database files were cataloged.

In this example, the KEYS data set is specified as a pre-defined and catalogued data set. Alternatively, you could define the data set inline.

Or you could specify the KEYS data set as a member of an existing partitioned data set.

```
//KEYS     DD DSN=S2K.TEST.SOURCE(KEYS1),DISP=SHR
```

The following statements show the sample JCL for executing CNVRT12:

```
//CONVERT JOB . . .
//CONVERT EXEC PGM=CNVRT12
//STEPLIB DD DISP=SHR,DSN=S2K.R120.LOAD
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//EMPLOYE1 DD DISP=OLD,DSN=USER.R120.EMPLOYE1
//EMPLOYE2 DD DISP=OLD,DSN=USER.R120.EMPLOYE2
//EMPLOYE3 DD DISP=OLD,DSN=USER.R120.EMPLOYE3
//EMPLOYE4 DD DISP=OLD,DSN=USER.R120.EMPLOYE4
//EMPLOYE5 DD DISP=OLD,DSN=USER.R120.EMPLOYE5
//EMPLOYE6 DD DISP=OLD,DSN=USER.R120.EMPLOYE6
//EMPLOYES DD DISP=OLD,DSN=USER.R116.SAVEFILE
//KEYS     DD DSN=S2K.R120.KEYS1,SPACE=(TRK,(2,1)),
//             UNIT=RIO,DCB=(LRECL=80,BLKSIZE=80,RECFM=F),
//             DISP=(NEW,CATLG,CATLG)
//CNVPARM  DD *
EMPLOYEE(CONVERT,REBLOCK)
/*
//EXEC PGM=SYS2K
        .
        .
        .
//KEYS     DD  DSN=S2K.R120.KEYS1,DISP=SHR
//S2KCOMD DD *
USER,DEMO:COMMAND FILE IS KEYS:EXIT:
//
```

In this example, the parameters are specified inline. Alternatively, you could specify an 80-byte sequential data set, for example, with the data set name of USER.PARMFILE.

```
//CNVPARM DD DISP=SHR,DSN=USER.PARMFILE
```

Or you could allocate the parameter file as a member of a partitioned data set, for example,

```
//CNVPARM DD DISP=SHR,DSN=USER.PDS(member)
```

The second logical step in the convert process executes a batch SCF job stream that will use the KEYS data set as input.

This same procedure could be executed as a separate job.

THE UNLOAD AND LOAD PROGRAMS

The Release 12.0 delivery tape contains two PL/I program generators, which create COBOL PLEX programs to unload and load SYSTEM 2000 databases. That is, one program unloads a database created before Version 12, the other loads the new Version 12 database. The COBOL PLEX programs can be precompiled with the COBOL PLEX preprocessor. Then you can execute them with the usual compile, link, and go procedures. The delivery tape also contains sample JCL, which you can modify for your site applications.

If you do not already have PLEX programs to unload and load your databases, you can use the program generators to convert existing databases to Version 12 format. You can also use them at any time to compact Version 12 databases. They create databases that are freshly loaded with no reusable space.

The unload program generator is S2KGUNLD. The load program generator is S2KGLOAD. The following topics give details about generating and executing these programs.

The S2KGUNLD Program Generator

The S2KGUNLD program generator is a PL/I program that generates COBOL PLEX source code for unloading a SYSTEM 2000 database. The database is unloaded to a sequential file. Each record contains the data from one data record, plus a 4-byte header.

The S2KGUNLD program generator is delivered as source code and as an executable load module. To compile and link the S2KGUNLD program from the source code, use the JCLGPUN job provided on the Release 12.0 delivery tape.

Input to the S2KGUNLD program You need two input files for the S2KGUNLD program.

- The DDname for the first input file is DESC. This file is required. It must contain the output from a SYSTEM 2000 DESCRIBE command for the database to be unloaded. The S2KGUNLD program uses the component definitions to build the COBOL WORKING STORAGE subschemas.

  Note: The lengths for the subschema items in the generated subschemas are equal to the pictures in the database definition, and no overflow is allowed. That is, although SYSTEM 2000 software allows overflow for long CHARACTER and TEXT items, the generated COBOL PLEX program does not; it truncates long values to the number of characters in the item pictures. The second input file provides a method of overriding the pictures for items that have overflow.

- The DDname for the second input file is SYSIN. This file is optional. To override the CHARACTER and TEXT type pictures defined in the database, the SYSIN file must point to a file containing 80-character records. Each entry in the SYSIN file is numeric. This file is free format with the following requirements:

  1. The first record must contain BLKSIZE and LRECL (record length) parameters, separated by one or more blanks. The first number must be the block size. If you specify 0, the block size defaults to 6212. If you specify a number greater than 32760, the block size is reduced to 32760. BLKSIZE and LRECL pertain to the S2KGUNLD output file OUTFL.

The minimum record length (LRECL) is four bytes longer than the smallest database record. The maximum record length is four bytes less than the BLKSIZE value.

2. Each subsequent record must consist of a pair of numbers, separated by one or more blanks. The first number is a CHARACTER or TEXT component number (without the C). The second number is the length of the longest data value for the specified item. To avoid truncation, choose a length that will be equal to or longer than the longest value. (To determine that second number for key items, you can review the output of a TALLY command.)

   From the sample SYSIN file shown below, the program generator will create a block size of 23464 and an LRECL of 150 for the OUTFL file. Components C2, C8, C201, and C301 will have lengths of 20, 15, 25, and 30 bytes, respectively, in the subschema definitions generated for the COBOL PLEX program.

```
23464 150
2     20
8     15
201   25
301   30
```

   Note: The first value in a SYSIN file must be the block size, and the second value must be the LRECL. Subsequent entries must be in pairs. If you do not adhere to this format, results are unpredictable. For example, suppose the LRECL (150) were omitted. The program would try to generate a block size of 23464 and an LRECL of 2, then component C20 would have a length of 8, C15 a length of 201, and so on.

If you dummy out the SYSIN file, the following defaults are used:

- OUTFL is a variable length blocked file with a block size of 6216.
- Item lengths in the subschemas default to the pictures given in the DESC file, that is, the pictures in the database definition.
- LRECL defaults to the length of the record containing the largest total number of characters for its items.

Output from the S2KGUNLD program The output from the S2KGUNLD program is a COBOL PLEX program that will unload a SYSTEM 2000 database. A DD statement with the name PROGRAM must appear in the JCL for the execution step of S2KGUNLD. This DD statement points to the data set that will save the COBOL PLEX source code. Typically this data set has an LRECL of 80. It can be a PDS member or a sequential file.

Executing the generated unload program The source code for the generated COBOL PLEX program to unload a database can be precompiled, compiled, and executed with the standard SYSTEM 2000 PLEX preprocessor, compile, link, and go procedures. Job JCLGCUN on the SYSTEM 2000 TEST library for Release 12.0 provides sample JCL for these tasks. Job JCLGCUGO provides sample JCL for execution.

The generated unload program requires the database files that are to be unloaded. It also requires a DD statement with a DDname of OUTFL. This data set will contain the data unloaded during the execution step. DCB parameters are not needed, because the BLKSIZE and LRECL values are assigned by the program.

Also, in order to unload the database, the unload program requires either the master password or a password that has retrieval authority for all components. You must specify the password as a parameter value in the GO step that executes the COBOL PLEX program, for example, PARM='DEMO' for the EMPLOYEE database. If you do not supply an appropriate password to the program, SYSTEM 2000 Error Code 24 is issued at execution time.

The S2KGLOAD Program Generator

The S2KGLOAD program generator is a PL/I program that generates COBOL PLEX source code for loading a SYSTEM 2000 database. The freshly loaded database tables will be created in Version 12 format.

The S2KGLOAD program generator is delivered as source code and as an executable load module. To compile and link the S2KGLOAD program from the source code, use the JCLGPLD job provided on the Release 12.0 delivery tape.

The load program reads the sequential file of data created by the unload program and loads the database using PLEX optimized load mode.

Input to the S2KGLOAD program The input files for the S2KGLOAD program are the same two files that you used for the S2KGUNLD program generator, the file containing the DESCRIBE output and the optional SYSIN file of item lengths. (For details, see **Input to the S2KGUNLD program** on page B-7.) Generation of the subschemas in the COBOL WORKING STORAGE section is also the same as for the S2KGUNLD program generator.

Output from the S2KGLOAD program A DD statement with the name PROGRAM must appear in the JCL for the execution step of S2KGLOAD. This DD statement points to the data set that will contain the COBOL PLEX source code generated by S2KGLOAD.

Executing the load PLEX program The source code for the generated COBOL PLEX program to load a database can be precompiled, compiled, and executed with the standard SYSTEM 2000 PLEX preprocessor, compile, link, and go procedures. Job JCLGCLD on the SYSTEM 2000 TEST library for Release 12.0 provides sample JCL for these tasks. Job JCLGCLGO provides sample JCL for execution.

Before you execute the load program, you must define the specified database with Release 12.0 of the software. You can use the DESCRIBE/DEFINE/ command to "unload" the database definition, then use that output as input to create the Version 12 database definition. You can define the appropriate passwords, strings, and functions for the Version 12 database in the same manner.

The load COBOL PLEX program requires a DD statement with the DDname of INFL. This data set contains the data to be loaded during the execution step. (The INFL data set is the OUTFL data set that was written when you executed the unload COBOL PLEX program.)

Also, the load program requires the master password to actually load the database. You must specify the master password as a parameter value in the GO step that executes the COBOL PLEX program, for example, PARM='DEMO'.

Special Considerations and Limitations

Consider the following guidelines and limitations when using the program generators:

- The block size for the OUTFL data set cannot exceed 32760 characters. The default size is 6216.

- The record length for the OUTFL data set must be at least four bytes less than the block size. The default record length is that of the largest schema record in the database to be unloaded.

- The COBOL PLEX programs that are generated reorder the schema items according to item type. Reordering is necessary because of DOUBLE and REAL items that might exist. DOUBLE type items must be on doubleword boundaries, and REAL type items must be on fullword boundaries in the COBOL PLEX programs.

- If you use a SYSIN file, the program generator programs require an input value for LRECL. The LRECL value determines the maximum length possible for a schema record in the database to be unloaded and loaded. If the largest schema record will not fit at the end of a block, the record will not be written until the next block. If the largest schema record length is relatively large, for example, 5000 bytes, then 4999 bytes could be wasted at the end of a block. Therefore, try to choose a record length that minimizes unused space. Also, try to have a record length that is small relative to the block size.

Installing the Program Generators

The two program generator programs are supplied in the Release 12.0 TEST library. The jobs listed below are supplied in the same library. They are generated by macros at base installation time when the JCLS2KIV job is run.

JCLGPLD	JCL to compile and link the S2KGLOAD program.
JCLGPUN	JCL to compile and link the S2KGUNLD program.
JCLGPLGO	JCL to execute the S2KGLOAD program.
JCLGPUGO	JCL to execute the S2KGUNLD program.
JCLGCLD	JCL to precompile, compile and link the generated COBOL PLEX load program.
JCLGCUN	JCL to precompile, compile and link the generated COBOL PLEX unload program.
JCLGCLGO	JCL to execute the generated COBOL PLEX load program.
JCLGCUGO	JCL to execute the generated COBOL PLEX unload program.

Sample Procedure

This section presents a sample procedure that unloads a Release 11.6 version of the EMPLOYEE database and loads it into a Version 12.0 database.

Step 1: Create the DESC File Submit an edited copy of the JCLEMP job from your Release 11.6 TEST library to send the results of a DESCRIBE command to a Report File. That Report File will be used as the DESC file for the PL/I program generators. The following commands send the DESCRIBE output to an alternate Report File called OUT1:

```
USER,DEMO:  DBN IS EMPLOYEE:
REPORT FILE IS OUT1:
DESCRIBE:
EXIT:
```

Also, issue some PRINT COUNT *record* commands so that you can use those results later to validate that the database has been built correctly.

Step 2: Determine the overflow for CHAR and TEXT items The unload and load programs will default to the item pictures in the database definition, which means that overflow characters will be truncated unless you override the pictures. If you already have PLEX programs that account for overflow, you can use them.

The TALLY command is useful for determining actual lengths for key item values. In the sample TALLY output shown next, the value AMERICAN INDIAN contains 15 characters. However, the picture for the ETHNIC ORIGIN item is only 9 characters long. Therefore, unless you override this length with an entry in the SYSIN file, only AMERICAN will be unloaded and loaded. Similarly, for C102, the value ADMINISTRATION & FINANCE is

23 characters long, and the picture for C102 is only 14 characters. Therefore, you need to override the picture for C102 also.

Here is sample TALLY output for items C8 and C102:

```
TALLY C8,C102:

*****************************************
 ITEM-         ETHNIC ORIGIN
*****************************************
OCCURRENCES   VALUE
-----------------------------------------
         2    AMERICAN INDIAN
         2    ASIAN
         7    BLACK
        31    CAUCASIAN
         8    HISPANIC
-----------------------------------------
         5 DISTINCT VALUES
-----------------------------------------
        50 TOTAL OCCURRENCES
-----------------------------------------
*****************************************
 ITEM-         DEPARTMENT
*****************************************
OCCURRENCES   VALUE
-----------------------------------------
         8    ADMINISTRATION & FINANCE
        11    CORPORATION
        18    INFORMATION SYSTEMS
        21    MARKETING
-----------------------------------------
         4 DISTINCT VALUES
-----------------------------------------
        58 TOTAL OCCURRENCES
-----------------------------------------
```

Step 3: Create the SYSIN file The entries in the SYSIN file override the default block size, record size, and item pictures used by the program generators S2KGUNLD and S2KGLOAD. Here is a sample SYSIN file for the EMPLOYEE database.

```
23464 160
2     15
8     20
102   25
104   20
201   25
301   29
411   40
422   20
```

The first pair of values override BLKSIZE and LRECL. The block size specified is 23464, which is one half-track of space on a 3380 disk. The record size of 160 approximates the number of characters in the largest record after accounting for overflow of item values.

Each subsequent pair of numbers represents a component number and an item length. The eight component numbers in this example represent items whose picture size in the database definition is too small to accommodate all the values for that item.

Although item values can overflow the picture size in the database, they cannot do so in S2KGUNLD and S2KGLOAD. That is, any values that exceed the item's picture will be truncated, unless the picture is overridden by an item length specified in the SYSIN file.

For a key item, you can avoid truncation by issuing a TALLY command, reviewing the output, and specifying a length in the SYSIN file that accommodates all the values. For a non-key item, you will need to estimate an appropriate length (if you believe some values are overflowing the picture in the database definition). It is better to estimate too high than too low.

Step 4: Run jobs JCLGPUGO and JCLGPLGO Run the JCLGPUGO and JCLGPLGO jobs from your Release 12.0 TEST library to generate the COBOL PLEX unload and load programs. The input to both programs consists of the DESCRIBE output file (DDname DESC) created in Step 1 and, if needed, the SYSIN file created in Step 3.

The output from the JCLGPUGO job is the COBOL PLEX source program to unload the database. The source code is sent to a file with the DDname PROGRAM. Usually, this file is a PDS with an LRECL of 80.

The output from the JCLGPLGO job is the COBOL PLEX source program to load the Version 12 database. The source code is sent to a file with the DDname PROGRAM. Be sure the data set that this DDname points to is a different member or data set than the one pointed to by DDname PROGRAM for the unload program (JCLGPUGO).

Step 6: Compile the COBOL PLEX programs In this step, you precompile, compile, and link the COBOL PLEX unload and load programs. Use job JCLGCUN for the unload program and job JCLGCLD for the load program. You can also compile both programs in one job by using two EXEC statements with COBNAME=*program* parameters.

Step 7: Unload the database with job JCLGCUGO In this step, you unload the database to be rebuilt, using job JCLGCUGO. The output file (DDname OUTFL) will contain the unloaded data. Each record is unloaded in logical order with a 4-byte header. If the output goes to a disk file, as in this example, the space allocated must be at least large enough to accommodate database Files 3 and 6, which contain the database data and overflow characters.

Step 8: Create a Command File to define the Version 12 database Issue SCF commands to create a Command File that can be used to define the Version 12 database. Modify the JCLEMP job from your Release 11.6 TEST library. Include a DD statement with the DDname of OUT2 for the alternate Report File.

In the following example, the SCF commands unload the EMPLOYEE database definition, strings, and functions in DEFINE format. They also write the MAP and EXIT commands to OUT2:

```
USER,DEMO: DBN IS EMPLOYEE:
REPORT FILE IS OUT2:
DESCRIBE/DEFINE/:
DESCRIBE/DEFINE/ STRINGS:
DESCRIBE/DEFINE/ FUNCTIONS:
ECHO "MAP:":
ECHO "EXIT:":
EXIT:
```

Step 9: Define the Version 12 database This step uses the Command File generated in Step 8 to create and define the Version 12 database. Using Release 12.0 of the software, issue the following commands to define the new database:

```
USER,DEMO: NDB IS EMPLOYEE:
COMMAND FILE IS OUT2:
```

Step 10: Load the database with job JCLGCLGO Load the new Version 12 database by running job JCLGCLGO. The job uses PLEX optimized load mode. The input is the data that were unloaded in Step 7. The DDname for this input file is INFL.

Step 11: Validate the new database After loading the Version 12 database, validate it to make sure that it was built correctly. Use the same PRINT COUNT *record* commands that you used in Step 1. Compare the output to ensure that all records were unloaded and loaded. You might also want to run some TALLY commands on key items so that you can check the values that were loaded.

Index

E

Z

3

Your Turn

If you have comments or suggestions about SYSTEM 2000 software or *SYSTEM 2000® Product Support Manual, 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.

SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513

1-55544-183-1