SYSTEM 2000® Software:
Product Support Manual

Version 1
First Edition

SAS Publishing

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SYSTEM 2000® Software: Product Support Manual*, *Version 1, First Edition,* Cary, NC: SAS Institute Inc., 2000.

SYSTEM 2000® Software: Product Support Manual, Version 1, First Edition

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

1st printing, November 2000

Table of Contents

Chapter 1: What Is SYSTEM 2000 Software?

Overview

This documentation describes Multi-User and single-user modes in SYSTEM 2000 software on IBM mainframes. Your operating environment must be OS/390 System/370XA or higher. System/390 under VM/ESA operating systems are supported, but they are not required.

This manual is for users who prepare SYSTEM 2000 jobs for execution, maintain SYSTEM 2000 at their sites, and monitor the status of data and job activities. You should be familiar with the basic concepts of your operating environment.

SYSTEM 2000 is a general-purpose database management system that enables you to define new databases and modify the definition of existing databases, and to store, retrieve, and update values in these databases. SYSTEM 2000 allows the organizational structure of the database to represent information about the data. The designer of a SYSTEM 2000 database determines which relationships are relevant and how data is represented. SYSTEM 2000 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
: allows you to define and change the definition of a database.

CONTROL
: is used for administrative functions, such as saving and restoring databases, assigning passwords, and enabling or disabling rollback.

QUEST
: refers to commands that are used to access a database for retrievals and updates. It is also known as ACCESS and natural language.

QUEUE
: combines updates into batches of commands for more efficient processing. It is available in both SCF and PLEX; however, it is best suited for PLEX where you can combine many update commands (commands are retained in work files) and have them all applied to the database during one pass of the database. Using QUEUE is more efficient than applying each update as the command is invoked. It is appropriate for use for large volume updates.

REPORT
: allows you to produce reports that have running subtotals, special formats, and logical pages within a physical page.

The PLEX facility allows you to access and update SYSTEM 2000 databases within a COBOL, a FORTRAN, a PL/I, or an Assembler program.

SYSTEM 2000 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 that prompts you for report specifications, which display database information formatted in columns.

SYSTEM 2000 Maintenance

Previous releases of SYSTEM 2000 were designed to allow for an overlaid system in memory. This was an important feature to some customers when the addressing range was limited to 16 megabytes (16,777,215 bytes). The need for an overlaid system has diminished, and it is no longer a choice. Consequently, maintenance procedures are simplified for Version 1. You do not need to re-link SYS2K and S2KPLI after applying a zap. The fix is applied directly to the executable module. The macro PRELNK, which was used to configure the system and to re-link after applying fixes, is not distributed.

Although transparent to the user, load module S2KPLI no longer exists. The functionality of S2KPLI is incorporated into SYS2K. Your existing PLEX programs will work with Version 1. This is possible because S2KPLI was loaded by the PLEX interface instead of being called, but now SYS2K is loaded, and processing control is passed to the PLEX entry point. Example job JCLS2KLN to re-link SYS2K is in Chapter 6, “Example Job Streams.”

Note: It might be necessary for you to re-link any SYSTEM 2000 module if

- you want to include user exit routines for security reasons
- object or load modules are distributed in place of a zap.

Chapter 2: Modes of Operation: Single-User and Multi-User

Choosing the Mode of Operation

This chapter gives an overview of the two modes of operation for SYSTEM 2000: single-user and Multi-User. After reading this chapter, you should be able to select the best mode for a specific task and know which method of communications, SVC or Cross-Memory Service (XMS), is best for your environment. The JCL and syntax for each mode are almost identical. However, Multi-User requires two extra files. Multi-User also has additional code to allow for inter-region communications, job scheduling, resource management by the user, and database integrity.

You can access your databases in either single-user mode or Multi-User mode. The basic difference is that, in single-user mode, you are the only user accessing a specific database; in Multi-User mode, up to 230 users can simultaneously access the same or different databases. In either mode, you can include JCL DD statements for database and work files, or you can omit the JCL and let SYSTEM 2000 dynamically allocate files.

Single-User Mode

The single-user mode enables one job to access one or more databases. Other users are not allowed access to those databases until the job terminates. Management of this limited access is accomplished by using JCL. Specifically, the option DISP=OLD for any file precludes other jobs from allocating the file. Additionally, if you specify the option DISP=SHR for database File 1, the database is opened for read-only access.

Single-user access is faster than Multi-User access because data is not passed from one region to another, and there is no overhead for job scheduling. Single-user mode is often used for high-volume updates in time periods when access to the database by multiple users is not needed.

Multi-User Mode

The Multi-User mode automatically coordinates database usage to avoid conflicts. For example, two users may issue requests to update the same record. Multi-User ensures that one update is complete before the second update begins. The coordination, temporary suspension, and dispatch of work are transparent to the user.

All resources, such as database files, work files, and buffers, are allocated in the Multi-User region where many users share them. You access databases by passing commands to Multi-User where the work is performed, and the results (replies to or acknowledges your request) are returned to you. Communications between your region (referred to as a dependent region or a dependent job) and Multi-User are automatic and transparent to you.

The Multi-User program S2000 runs in one address space or region. User jobs (dependent regions) are executed in other regions or address spaces. Dependent regions communicate with Multi-User by means of an SVC, or S2KPC for XMS. WAIT and POST are used to control the sequence of processing. When you invoke a command or a command stream by pressing the ENTER key, or if you are in batch mode when an end-of-command is encountered, the dependent region interface code issues an SVC or a PC call and then a WAIT. The dependent region remains in wait status until Multi-User responds with an answer and a POST to the waiting ECB.

Communications between your region and Multi-User occur in one of two ways: SVC calls or XMS. The recommended method is XMS, but both techniques are supported. There is no noticeable difference to the end user between the two systems. However, from a systems perspective, the differences are significant. Two important differences are the SVC slots and the SYSTEM 2000 load libraries that are required.

XMS needs one SVC slot per version. That is, Version 1 and Release 12.1 cannot share the same SVC slot. However, each Version 1, XMS Multi-User can share this one SVC slot with other Version 1 Multi-Users. Module S2KCMC is zapped with the SVC slot number, and then uses the S2KCOM file to identify each Multi-User. Also, S2KCMC uses enqueue logic to ensure that each Multi-User has a unique S2KCOM data set. This means that you can run several Multi-Users from the same load library, and only the S2KCOM is different. Even a Version 1 test Multi-User can share the same slot as the Version 1 production Multi-User with no interference. The one exception is when you are testing changes to the S2KPC routine itself.

On the other hand, the SVC version requires a unique SVC slot for each Multi-User. Because this SVC number must be zapped in dependent-region interface code, as well as in the Multi-User code, you need a separate load library for each Multi-User region. Maintaining multiple load libraries requires more work when periodic maintenance is applied.

SVC Version

The differences between the SVC and the XMS versions of Multi-User are the type of communications link and where memory areas are assigned. Table 2.1 compares the two versions.

Table 2.1 Differences between XMS and SVC

XMS	SVC
data areas are in Multi-User	
requires any type of SVC slot	requires a type 2 SVC for each active Multi-User
no SVC to link	must be linked into the nucleus
CSA usage is limited to the load module S2KPC (about 12K)	some data areas (primarily, copyareas) are acquired from the CSA
several Multi-Users can be active simultaneously, all using the same SVC slot	

The MRISVC macro that is supplied by SAS generates a type 2 SVC. Active Multi-Users must have their own type 2 SVC. The SVC runs enabled, except when system queues are searched or when disabling is required to call system routines. The SVC validates each call to prevent unauthorized access. Calls are accepted only if originated by a SYSTEM 2000 routine. The SVC is not re-entrant.

CLEARS2K is a program that

- initializes a table (SVCTABLE) within the SVC
- posts users who are still active when Multi-User abnormally terminates
- frees GETMAINed areas in the CSA.

Although CLEARS2K is necessary only when Multi-User abnormally terminates, you can run CLEARS2K before initializing Multi-User to ensure that you do not get WTO119 and user abend 122. It is safe to run CLEARS2K at any time, except when Multi-User is active.

Cross-Memory Services

You can use the same SVC number to run a maximum of 4096 XMS Multi-Users. The cross-memory module S2KPC is re-entrant, and it communicates with the correct user and Multi-User based on the Cross-Memory Services linkage table (instruction LX). A communications file (DD name S2KCOM) holds the linkage information. An S2KCOM file must be allocated in both the Multi-User and dependent regions. Part of the Multi-User initialization process is to acquire cross-memory linkage authorization and to write that data to the S2KCOM file.

You allocate the same communications file in your region (dependent region). The SYSTEM 2000 interface reads the file and issues a PC instruction to direct communications, by means of S2KPC, to the Multi-User that shares the same file. Use of the S2KCOM file to direct communications allows many Multi-Users to work with a single copy of the cross-memory communications routine S2KPC.

An SVC slot is needed to hold the address of S2KPC. The SVC number is zapped into S2KCMC, the authorized program that you execute to bring up XMS Multi-User. It is zapped at displacement X'OA' with an SVC number. During initialization,

1. S2KCMC indexes the system SVC table to the slot of the SVC number that is zapped into S2KCMC. The slot contains the address of S2KPC or 0's if S2KPC is not loaded.

2. If the address is not 0's, S2KCMC confirms that S2KPC resides at that location. If not, or if the address is 0's, S2KCMC loads S2KPC into the CSA and uses the macro SVCUPDTE to put the address into the SVC table.

3. S2KCMC then obtains cross-memory linkage authorization from the system and writes the data to the S2KCOM file, along with other identifying information. The S2KCOM file is 30 bytes in length.

4. S2KCMC then loads Multi-User (program S2000) from the SYSTEM 2000 LOAD library and transfers control to SYSTEM 2000 for the duration of the session.

S2KCMC attaches CLEARS2K during termination processing. Functions performed by XMS CLEARS2K are similar to those performed for the SVC version. However, the SVC table is located in the Multi-User region, and you never execute CLEARS2K. It is an automatic process invoked by S2KCMC.

Executable Programs

A list of programs that you can execute, and a brief description of each program follows. The program you execute determines single-user or Multi-User mode. In Multi-User mode, the program you execute determines whether you are batch SCF or TP SCF.

Single-User Programs

SYS2K — Single-user SCF. This program can be executed in batch, or it can be called from TSO as an interactive job. Interactive means that you enter a command, wait for a response, and then enter another command. In TSO, the program is active until you enter the EXIT command.

PLEX — The START S2K statement determines single-user or Multi-User. START S2K USE(0) is single-user. You can also use a variable name that has a value of 0 to serve the same purpose. That is, START S2K USE(SWITCH) where SWITCH has the value of 0. Any positive value means Multi-User.

Your PLEX program can run in either 24- or 31-bit addressing mode. The SYSTEM 2000 interface ensures 31-bit mode when calling SYSTEM 2000 software. The interface returns to your program in the same mode in which the interface was entered. You can switch modes as often as you like within your program. Your user exits will always be entered in 31-bit mode.

Multi-User Programs

S2KCMC — Program for the XMS version of Multi-User. S2KCMC must reside in an authorized library.

S2000 — Program for the SVC version of Multi-User.

Multi-User Interface Programs

SYS2KJOB

Batch SCF interface to Multi-User. One thread is allocated to the job until the job terminates. This program can also be called as a TSO interactive program. Interactive means that you enter a command, wait for a response, and then enter another command. In TSO, the program is active until you enter the EXIT command.

SYS2KTPI

TP SCF interface to Multi-User. The TP SCF interface is designed to require fewer resources than SYS2KJOB so you can support more users at the same time. The processing of SCF commands is the same, but minimal resources are required between commands. Activities such as user think-time and typing new commands have minimal impact on Multi-User throughput.

There is one key difference between batch and TP SCF. Batch SCF continues to read input until it has a complete command. TP SCF needs the complete command in one TP segment. This means that every command you type must end with the colon (:) terminator or you will get the message: UNEXPECTED END-OF-COMMAND FILE. A TP segment is limited to 1200 characters.

Additionally, processes that require multiple commands must all be entered in the same TP segment, for example,

- DEFINE: <define commands> MAP:
- QUEUE: <SCF commands> TERMINATE:
- REPORT: <report writer program> GENERATE ALL:
- FRAME: <SCF update commands> END FRAME:

An alternative is to use an alternate file, for example, COMMAND FILE IS *ddname*: where *ddname* identifies a sequential file in the Multi-User region. SYS2KTPI also allows for alternate files to be allocated to the dependent TSO region by adding the word LOCAL in front of the alternate file command, for example, LOCAL COMMAND FILE IS *ddname*:

SYS2KTPI allows the TSO SUBSET command which means you can issue TSO ALLOC commands so you can use the LOCAL option without exiting SYSTEM 2000.

CICS S2KU TRANSACTION

S2KU is a CICS transaction that invokes a TP SCF interface to Multi-User. S2KU has the same restrictions for a TP segment as is documented for SYS2KTPI. However, it does not have the LOCAL option or the TSO SUBSET command. It does have a command editor, which is invoked by using the S2KE transaction that is processed similar to a local command file. See *SYSTEM 2000 Software: Interface to CICS Version 1, First Edition* for more details.

PLEX

Each PLEX command generates a CALL statement. The CALL statement is always to S2KPL for non-CICS programs. For CICS programs, the call is to PLXPBLD. For COBOL II programs, the call is to PLXFRMT. S2KPL or PLXPBLD are linked with your program to form an executable module of any name you choose. PLXFRMT is loaded by CICS; it should be resident for performance.

Except for CICS programs (CICS is always Multi-User), the same load module can be run in single-user or Multi-User mode. The mode is determined by the USE parameter in the START S2K statement. USE can refer to a variable that is defined within the program, such as START S2K USE(SWITCH). If SWITCH has a value of 0, it is single-user mode. Any other positive value invokes Multi-User mode. You can set the value of SWITCH by using a VALUE statement, reading an input line, or passing a parameter in the EXECUTE statement.

Your PLEX program can run in either 24- or 31-bit addressing mode. The SYSTEM 2000 interface ensures 31-bit mode when calling SYSTEM 2000 software. The interface returns to your program in the same mode in which the interface was entered. You can switch modes as often as you want to within your program. Your user exits will always be entered in 31-bit mode.

Chapter 3: Database

Overview

This chapter explains

- how DDnames for database files are derived for JCL statements.
- logical and physical opens.
- how to open databases for read-only or for update.
- why a specific status is placed on a database in Multi-User mode.
- each database file capacity so you can make appropriate space allocations .
- how to prevent out-of-space conditions by monitoring file usage using SCF PRINT SIZE or the EXAMINE utility.

A database consists of six disk files. DDnames for the files are composed of the first 7 characters of the database name and a numeric suffix. Use the character X to replace a space in the DDname if your database name is shorter than 7 characters. The numeric suffix ranges from 1 through 6. In addition, there are two files used for recovery. DDnames for these files are composed in the same way as for the first six files. See Table 3.1 for file descriptions.

Table 3.1 Contents of Database Files

Files	Contents
1	information about the database, such as the definition, update cycle, and size of the database.
2 and 4	items defined as KEY.
3 (overflow file)	character or text data value that exceeds the defined length. (For example, you define an item as 10 characters in length, but some values exceed 10 characters so the excess overflows to File 3).
5 (hierarchical file)	one 18-byte entry for each record in your database. An entry contains pointers to its parent record, next sibling, and first descendant. It also has a pointer to the corresponding data record in File 6.
6 (data file)	all of your data (except for overflow, which is in File 3).
7 (Update Log or transaction file)	information about every update to the database. (You can use File 7 to recover updates if your database becomes damaged).
8 (Rollback Log)	image of blocks before they are updated. This file allows the database to be rolled back to a previous cycle.

An open database means that the macro OPEN has been processed for each of the database files, and that the files are available for access. A *physical open* is the process of opening the files. A *logical open* is the process of making a physically open database available to you. For example, when a user is working with the database EMPLOYEE, the database is physically open. You then access the same database. This invokes a logical open. The database EMPLOYEE is now physically open to one user and logically open to a second user.

There also is a physical and logical close of databases. A *physical close* is a process where the macro CLOSE is invoked, and the files cannot be accessed until they are physically opened. A *logical close* is the process of removing the database from your list of open databases.

Open Read-Only

Database files are always opened for update in Multi-User mode because many users may access the database; some for retrieval and others for update. There are two ways you can limit access authority. One is with a password, and the other is with user exit 45. In single-user mode, database files may be opened for update or for read-only. The criterion for read-only is DISP=SHR for File 1 or a password that has no update authority. One or both of these conditions cause the files to be opened for input only. In all other cases, the files are opened for update. However, you can still use exit 45 to limit access to read-only.

DISP=SHR on File 1 allows you to use single-user mode for retrievals while the database is open for any type of activity in Multi-User mode or for retrievals in another single-user job. Although you are allowed to process retrievals in single-user mode while the same database is being updated in Multi-User mode, there is a possibility of an error condition in single-user. The error is harmless, but your single-user job will be prematurely terminated. Potential for error exists because the buffer manager in Multi-User writes buffers only when necessary. Therefore, data on disk that is available to single-user may be incomplete.

If you attempt to update a database that is open for read-only, one of the following error conditions is invoked:

- SCF error code 816 or PLEX return code 47 is issued if the password does not have proper authority.
- SYSTEM 2000 error code 815 occurs if you attempt to update a database in single-user mode by using DISP=SHR for File 1.
- Any update attempt that was not stopped with error code 816 or 815 causes an I/O error (attempt to write to a file that is opened for input only) and SYSTEM 2000 error code 817.

Database Status

There are many types of hold and lock conditions that may be put on a database. Additionally, a single record may be held (local hold) in PLEX. All holds apply only to Multi-User. In single-user mode, the implication is that you are the only user accessing the database, therefore, hold and lock are unnecessary conditions.

Most restrictions are invoked and removed automatically to ensure data integrity. Three exceptions are EXCLUSIVE, LOCAL HOLD, and FRAME. The duration of these three conditions is strictly up to you.

EXCLUSIVE — means the database is locked and can be used only by you. You cause the exclusive condition by specifying exclusive use when you issue the DBN (SCF) or OPEN (PLEX) command. Exclusive use is granted only when the database is not open to another user. After it is granted, no one else can open the database until you exit, open another database in SCF (current database is closed to you), or issue a DBN command for the database without the EXCLUSIVE option. When using PLEX, you must close the database to remove the exclusive condition. Additionally, EXCLUSIVE in PLEX prevents you from having any other database open.

LOCAL HOLD	means that no other PLEX user can place a hold on that specific record while the hold is in effect. Other users can retrieve, providing they do not specify a hold option. LHOLD does not prevent the record from being updated by another user who does not use a hold option when ROLLBACK is disabled. (See LHOLD execution parameter.) A local hold is placed when you specify /HOLD on a retrieval. The hold is maintained until a COMMIT or DROP HOLD command is invoked. Moreover, a local hold is automatically placed on a record that you update if coordinated recovery is enabled. There is no LOCAL HOLD in SCF.
UPDATE	is placed on a database in response to an update command. Other activities directed against the database while it has UPDATE status are temporarily suspended, and then automatically dispatched when UPDATE status is removed. The change in database status is rapid and transparent to the user.
NON-KEY MODIFY	means a PLEX user is issuing a MODIFY or a REMOVE command for a subschema record or inclusion-list that contains all non-key items. Retrievals and local holds can process simultaneously when in non-key modify status.
RETRIEVAL	means that data is being extracted from the database. Commands that require update status are suspended while RETRIEVAL is in effect. RETRIEVAL applies to both SCF and PLEX.
ROLLBACK	is placed on the database for the duration of coordinated recovery. ROLLBACK cannot co-exist with any of the other conditions described in the database status section except for LOCAL HOLD.
GLOBAL HOLD	is a status that allows multiple users to have the database open, and it allows retrievals. GLOBAL HOLD cannot co-exist with any other type of hold or with update status. Any update command for the database is suspended if it is not from the user who issued the global hold. Situations that cause a GLOBAL HOLD are • GET*x* or HOLD commands in PLEX queue mode. • where-clause phase of an SCF update command. The global hold is changed to update when WHERE processing is complete. Both WHERE and retrieval parts of the SCF retrieval command acquire only retrieval status. • FRAME. • TERMINATE of queue or terminate sessions with updates (SCF). • SAVE puts update status on a database for the duration of the SAVE process. • KEEP puts a global hold on a database for duration of the KEEP process. • Re-set of Rollback Log.

Savefile

The Savefile is a QSAM file, which can be tape or disk. The SAVE command copies the six database files to the Savefile. The RESTORE command copies the Savefile to the six database files. SAVE and RESTORE provide an exact copy. Nothing is altered. The page size and CISIZE cannot be changed when using the SAVE and RESTORE commands.

No pad space or SYSTEM 2000 buffers (POOL 0-7) are used during SAVE and RESTORE operations. VSAM buffers are used for I/O to the database files. QSAM buffers are used for I/O to the Savefile. The default number of buffers for each access method, VSAM and QSAM, is 5. You can gain substantial performance improvements by specifying more buffers for both. To specify the number of buffers, for VSAM, use the SYSTEM 2000 parameter DBBUFN=; for QSAM, allocate the Savefile in JCL and specify DCB=(bufno=*n*). The best number of buffers to use depends on the size of your database. For fine-tuning, you may want to start with 30 VSAM buffers and 30 QSAM buffers, then check your job message file for CPU time and I/O counts. Adjust the buffer counts until you achieve the best performance. Note that VSAM and QSAM buffers are independent of each other. For example, you might find that the best performance is achieved with 50 VSAM buffers and 40 QSAM buffers. Additionally, you will find that more is not always best. Excessive buffer assignments can have a negative effect on CPU time and I/O counts.

Keepfile

The Keepfile is a BSAM file, which can be tape or disk. The file is needed by SYSTEM 2000 only while a KEEP or an APPLY command is being processed. At all other times, the file is closed and, unless specified in JCL, unallocated.

A buffer from POOL 7 is assigned while the file is open. The Keepfile must have a blocksize equal to S2KPAD00 CISIZE-8. The blocksize is set automatically when the KEEP that is being processed is written to the beginning of the file. If a KEEP is to be appended to other KEEP commands already executed in the file, the blocksize is not changed. A KEEP is appended when you issue multiple KEEP commands without issuing an intervening SAVE or RESTORE command.

Database File Capacities and CISIZE

You can have a maximum of 10,000 items in a database definition and a maximum of 119,304,647 data records. The maximum capacity of a VSAM file is 4,294,967,295 bytes, which is the limit of database File 6.

File 1 After File 1 is created by using the NDB command, the required size never increases. You can allocate more space, but it is not used. The size of File 1 is derived from the maximum items (MXCOMP) allowed, which is established by the NDB command. The default is 430. For example, NDB is EMPLOYEE/500 would set MXCOMP as 500.

You can calculate the number of control intervals for File 1 as follows:

(MXCOMP × 18) + 2016 = a (Round the value of **a** to a multiple of CISIZE − 20).

Greater of (CISIZE × 2) or MXCOMP × 100 = b (Round the value of **b** to a multiple of CISIZE − 20).

Greater of (CISIZE × 2) or MXCOMP × 4 = c (Round the value of **c** to a multiple of CISIZE − 20).

(a + b + c = bytes for File 1)

bytes for File 1 / (CISIZE − 20) = required control intervals.

File 2 File 2 is for key items. It has the item value and a pointer into File 5 for each occurrence of a key value. If the exact value for an item occurs more than one time, the value is recorded in File 2 only one time, and the pointer is to File 4 where there is a list of all File 5 pointers for that value. Values for an item are kept in sorted order. Pages are chained with pointers so that every value for the item is recorded in File 2, along with a pointer to either File 4 or to File 5. The pages also are indexed by high value on the page. There can be up to 6 upper-level index pages for a specific item. This indexing technique allows for quick processing of key items, including range specifications in WHERE processing. The upper-level index isolates lower-level pages that are needed, thus keeping I/O to minimum levels.

The padding of values reserves space at the end of a lower-level File 2 page when data is added to the last page for an item. Only the last page for an item is affected. During a load, each page that is being updated is the last page, which results in padding on every page. Padding does require more File 2 space, but gains in performance by reducing page splits make it worthwhile. See the *SYSTEM 2000 QUEST Language and System Wide Commands* manual for details about padding and how to enable it.

File space requirements depend on the number and size of valued key items. The maximum size of File 2 is 1,073,741,823 bytes. That is roughly 263,430 pages of CISIZE 4096.

File 3 File 3 is the Extended Field Table (EFT). File 3 is also called the overflow file. Items defined as character or text might have values that exceed the defined length. Data in excess of the defined length is written to File 3. Definitions for strings and functions, and item names that are longer than 4 characters are also recorded in File 3. If most values for a key item exceed the defined length, you should consider re-defining the item to make it larger. Retrieval or update of an item value that overflows requires an extra I/O. Part of the value is recorded in File 6, and a pointer in that file indicates where the remainder of the value is located in File 3. The maximum size of File 3 is 268,435,455 bytes.

File 4 File 4 contains a list of pointers for key items that have values that are not unique. For example, the value MALE for gender may occur many times. File 2 has a pointer to File 4 where there is a list of pointers to records that have the gender equal to MALE. Pointers for each key item are maintained in sorted order. Consequently, padding may reduce the number of page splits and significantly improve performance for updates. There is a one-word entry for each repetition of a value. The value itself is not recorded in File 4. The maximum size of File 4 is 1,073,741,824 bytes.

File 5 File 5 contains one 18-byte entry for each record in the database. Every access to a record must first retrieve an entry from File 5. The File 5 entry points to the data record in File 6 and to the parent, the first child, and the next sibling data records within File 5. The maximum size of File 5 is 2,147,483,646 bytes, or 119,304,647 records.

File 6 File 6 contains data in data record format. All data values are recorded in File 6. Every access to File 6 must first retrieve a pointer from File 5. The maximum size of File 6 is 4,294,967,295 bytes. Beginning with OS/390, Version 2.8 and DFSMS 1.5, you can allocate File 6 as an extended format VSAM file that has a maximum size of 8,589,934,588 bytes.

File 7 File 7 is the Update Log.

File 8 File 8 is the Rollback Log. The CISIZE of File 8 must equal the largest CISIZE of Files 1 through 6. The CISIZE of Files 1 through 7 can be any of the sizes recognized by SYSTEM 2000. See the list of valid values for CISIZE in Chapter 5, “Buffers and CISIZE.”

Chapter 4: Work Files

Overview

Work files are for the temporary storage of information about a user, or the processing of a command on behalf of a user. Work files are user or command related rather than being associated with a database.

This chapter describes each work file, identifies the mode in which it is required, and gives you the DDname for each file and whether the file can be dynamically allocated.

S2KPAD

S2KPAD is a VSAM ESDS file. Although one or two PAD files should suffice for most jobs, you may use a maximum of 16 PAD files in any single-user or Multi-User job. The previous requirement that multiple PAD files all be on the same device type has been removed. Each PAD file can be multi-volume. SYSTEM 2000 uses secondary allocations if they are available after the primary allocation has been exhausted. You can use JCL to allocate the files, or they may be dynamically allocated. DDnames for the files are S2KPAD with a two-digit numeric suffix that can range from 00 through 15, for example, S2KPAD00. In any jobstream, all PAD files must be the same CISIZE as S2KPAD00 or they will not be used. When an invalid CISIZE is detected, the scan for additional PAD files stops. The PAD file cannot be reformatted using a different CISIZE; therefore, CISIZE and buffer size in POOL7 must match. If they do not match, the buffer size in POOL7 is changed to equal the CISIZE of S2KPAD00.

You must use an S2KPAD file in Multi-User mode, and in single-user mode, when using the Report Writer or Collect feature. In single-user mode, when not using Report Writer or Collect, you can use seven BSAM files that have the DDnames S2KSYS01 through S2KSYS07 instead of S2KPAD. These files are called Scratch files.

The PAD*nn* execution parameter is retained in Version 1 of SYSTEM 2000; however, the only allowable values are YES or NO. Number-of-allocation-units / blocks-per-allocation-unit is no longer valid. SYSTEM 2000 always calculates these values and ignores those entered by users. PAD*nn* must be defined consecutively starting with PAD00. PAD*nn*= YES means that SYSTEM 2000 tries to allocate an existing S2KPAD*nn* file or dynamically define a TEMP file if one does not already exist. PAD*nn*=NO means that the specified PAD parameter and any other subsequent PAD parameters that have higher PAD numbers (*nn*) are ignored.

It is recommended that PAD files be permanent. However, if a file does not exist for a corresponding PAD*nn* parameter, SYSTEM 2000 dynamically defines a TEMP PAD file by calling IDCAMS. The SYSTEM 2000 parameter PADVOL must be coded to specify where these TEMP PAD files will reside. The parameters PADPRI and PADSEC may be used to override the default primary and secondary space allocations of 1 cylinder each. The parameter PADUNIT is no longer valid. If SYSTEM 2000 is unable to define a TEMP data set by using the usual naming convention DSN=*prefix*.S2KPAD*nn*, DISP=OLD; a *suffix* of .TEMP*nn* (*nn*=1 through 9) is added. For example, DSN=*prefix*.S2KPAD*nn*.TEMP*nn*, DISP=OLD. At termination, SYSTEM 2000 calls IDCAMS to delete any TEMP file that was dynamically defined. Due to system crashes or other unforeseen circumstances, SYSTEM 2000 may not be able to delete these files at termination. However, an attempt will be made to re-use them in subsequent jobs.

You can use any of the six values for CISIZE that are recognized by SYSTEM 2000. The sizes range from 4096 to 26624. It is best to use a large CISIZE for the PAD file in order to reduce I/O. One buffer is assigned to each Scratch file in use, and many activities require scratch space less than one buffer in length. In such cases, there is no I/O. For activities that use more than one buffer of data for a Scratch file, there are two I/O's for each block of data. One I/O writes to the file and another I/O reads the file when the data is needed. Two writes using CISIZE 26624 require 6 to 14 writes using 4096.

Scratch Files S2KSYS01 through S2KSYS07

In single-user mode, when not using Report Writer or Collect, you can use BSAM Scratch files instead of S2KPAD. One advantage to using these files is that Scratch files can be assigned to tape or to disk, but at least one must be assigned to disk. For activities that use a lot of scratch space, such as LOAD and RELOAD, you can assign one or more Scratch files to tape. The DDnames for these files are S2KSYS01 through S2KSYS07. RECFM and BLKSIZE are set automatically or changed when the file is used. Scratch files are not dynamically allocated.

Locate Files

Locate Files are used with the PLEX LOCATE command. You can have a maximum of 16 Locate Files for each program. When using PAD files, the space for Locate Files is obtained from the PAD. Therefore, you do not specify JCL for the files.

Note: PAD files are required for Multi-User mode.

If you use Scratch files and PLEX Locate Files (other than LOCATE00), you must allocate the BSAM files. Locate Files are not dynamically allocated. They have DDnames that range from LOCATE00 through LOCATE15. If you use only File 0, you do not need to specify the file in JCL, whether or not you use PAD files. File 0 is equated internally to Scratch file 1. However, if you use a file other than 0, you must allocate all files used, including 0.

Buffers for Locate Files are assigned from POOL7. A buffer is assigned and locked for each active Locate File. The file remains active until job termination, or until you issue a LOCATE command that does not qualify any records. Data that is written to Locate Files consists of a one-word address for each record. Therefore, the amount of space needed is minimal. For example, addresses for 5,000 located records can easily fit into one block.

Sort Files

Sort files are BSAM files. It takes six files to make one set of Sort files. DDnames for the first set are SF01 through SF06. There must be one set of Sort files for each thread (See THREAD=execution parameter). Because Sort files are allocated by thread, their DDnames are derived from the thread number.

Let *t* represent the thread number and *n* the first digit or the first two digits in a three-digit number of a Sort file DDname suffix. Then, the formula for Sort file DDnames is

SF*n*1 through SF*n*6

where $n = t-1$. For example, thread 11 would have Sort files SF101 through SF106 and thread 3 would have Sort files SF21 through SF26.

In single-user mode, you can allocate Sort files to tape or to disk, but at least one file must be a disk file. For Multi-User mode, all Sort files must be allocated to disk. The block size of Sort files is determined at job initialization time. The block size is one-half the Scratch file page size. Sort files cannot span volumes (that is, no multi-volume files). Sort files can be dynamically allocated.

S2KUSERS File

The S2KUSERS file is a VSAM ESDS file that is used only in Multi-User mode. This file is used to record information about TP users when the user is inactive. *Inactive* means that the TP user is signed on to Multi-User, has received a response from the last command to Multi-User, and is not waiting for anything from Multi-User. The S2KUSERS file allows the control block USERCOM or URA to be released for use by other users. The number of control intervals (blocks of data) in the file is at least equal to the value of the TPSCRUN execution parameter. The CISIZE of the file is 12288. The RECORDSIZE is 12281. The S2KUSERS file can be dynamically allocated.

S2KCOM File

The S2KCOM file is a VSAM RRDS file that is used for communications with Multi-User under XMS. This file is not used with SVC type communications. The S2KCOM file is 30 bytes in length. This file is initialized by S2KCMC and contains the SVC number, cross-memory authorization information, and program call information (including the program call number). Before writing to the S2KCOM file, S2KCMC issues an ENQ in the DSN and VOLSER. The ENQ prevents another Multi-User from using the same S2KCOM file. The S2KCOM file must always be allocated with DISP=SHR. It is the S2KCOM file that identifies the Multi-User that your job will communicate with. The S2KCOM file is not dynamically allocated.

S2KPARMS File

The S2KPARMS file applies to both single-user and Multi-User modes. It can be a sequential file, member of a PDS, or an inline file in your job stream. Additionally, parameters can also be specified in the EXEC statement. Its record length is 80 bytes. Each input line must contain only one parameter. The S2KPARMS file can be dynamically allocated. The DSN used for dynamic allocation is S2KPARMS unless you specify 'PREFIX=<*prefix*>' in the EXEC statement. Parameters passed in the EXEC statement take precedence over those in a file. Therefore, you can specify PREFIX in both the EXEC statement and in the parameter file. The value specified in the EXEC statement is used.

User Files

SCF requires 4 user files, through which all communication is performed: 2 files for input and 2 files for output. The files are required for both single-user and Multi-User modes. One of these files, S2KMSG, is also used by PLEX. You do not have to specify user files for TP. Both input and output default to the terminal.

For tape files, you do not need to specify attributes if standard labels are used. However, if you use non-labelled files, you must specify LRECL, BLKSIZE, and RECFM for the DCB. LABEL=(,NL) must also be specified. The LRECL of these files can be any value in the range 44 through 256. If you do not specify the LRECL, appropriate defaults are used. The default is the terminal line size for terminal files. If the LRECL, either in the DCB or in the file definition, is outside the acceptable range, defaults are used. If the incorrect LRECL is less than 44, the defaults LRECL=80 and BLKSIZE=6160 are used. If LRECL is greater than 256, the defaults LRECL=256 and BLKSIZE=4096 are used.

INPUT Files

The command and data files are read by SYSTEM 2000. Each of these files defaults to INPUT, the file identified by the DDname S2KCOMD. For example, the command COMMAND FILE IS INPUT directs SYSTEM 2000 to read the file that has the DD statement S2KCOMD. The command file contains all SCF input commands. It often is specified as //S2KCOMD DD *.

The data file contains input data for commands such as LOAD. It is accessed when *DATA* is encountered in the command file, and when the command DATA FILE IS DDname is invoked. A DD statement for this file is not required unless you invoke an alternate data file.

OUTPUT Files

SYSTEM 2000 writes to the Message and Report Files. Each of these files defaults to OUTPUT, which is the file identified by the DDname S2KMSG. SYSTEM 2000 places a blank in the first character of each output record as the carriage-control character, except for output from the UNLOAD command.

The Message File receives all error diagnostics, informative messages, and echoes of your input commands. The Message file is required for all communications with SYSTEM 2000, including PLEX. The CICS PLEX interface does not write to the Message File. Multi-User sends information for the Message File to the CICS interface, where it is ignored.

The Report File receives output from SCF commands. PRINT, LIST, UNLOAD, and Report Writer output are written to this file. Report and Message are the same file, identified by the DDname S2KMSG. You can use alternate files for either or both files. For example, if you specify the file as //S2KMSG DD SYSOUT=A, information for both the Message and Report Files is written to SYSOUT. If you also specified //X DSN=MYFILE and issued the command REPORT FILE IS X, messages are written to S2KMSG and report information is written to X.

Alternate Files

Alternate files are those that you identify in an SCF command. You can switch between alternate and the standard files as often as you want to. When you switch files, the current file is closed and the file you specify is opened. You identify standard input files (Command and Data) by using the keyword INPUT. For example, COMMAND FILE IS INPUT or DATA FILE IS INPUT. You identify standard output files (Message and Report) by using the keyword OUTPUT. For example, MESSAGE FILE IS OUTPUT, REPORT FILE IS OUTPUT. Either of these commands assigns the specified file to the DDname S2KMSG. You identify an alternate file by using the file DDname in a command.

Put the DD statements that identify alternate files for TP in the Multi-User job stream. For non-TP jobs, put the DD statements in the dependent-side job stream. You can use the LOCAL option to identify alternate files for TP. Put the local file definitions in the dependent-side job stream. For example, LOCAL REPORT FILE IS X. To switch back to the standard file for report, you specify REPORT IS OUTPUT.

S2KOUTP

S2KOUTP is a QSAM file that has a fixed-block format and a block size of 4096. S2KOUTP is an optional output file for TP and is always available for TP. Its purpose is to prevent purge of output in excess of the 4k that is destined for the terminal. Multi-User builds output for TP in a 4k block and returns the output when a 4k block is full, or when all output for a segment is ready. A segment is a single transmission of commands from the interface to Multi-User, up to a maximum of 1200 characters. All commands in an alternate command file are also considered one segment. If you do not allocate S2KOUTP, it is dynamically allocated with a space allocation of (CYL,(1,1)). All files, except alternate, can be dynamically allocated.

Chapter 5: Buffers and CISIZE

Buffers

A *buffer* is a fixed length of primary storage into which data is read from secondary storage, and from which data is written to secondary storage. All computer environments have primary storage and secondary storage. In primary storage (also called main storage or memory), data is computed and manipulated. In secondary storage (tape or disk), data can be permanently stored and accessed but not manipulated. The correct number of buffers for a specific task and the CISIZE of the file are important factors in achieving optimal performance. Both are related to I/O, which is one of the most time-consuming processes. Because most work that is performed by a computer involves a buffer, efficient buffer usage by a DBMS is essential.

SYSTEM 2000 has its own buffer manager system, which is very efficient. The number and size of buffers are specified by using execution parameters. The parameters are POOL0 through POOL7. The POOL0 through POOL6 parameters are for database usage. The S2KUSERS file also uses a buffer from one of these seven pools. The buffers are located above 16mgb, therefore, their usage is limited to VSAM files. The POOL7 parameter is used for work files. The PAD file S2KSYS01 through S2KSYS07, the Locate files, Keepfiles, and sort files use POOL7. The buffers are below 16mgb; therefore, they may be used for I/O by VSAM, BSAM, and QSAM. The minimum buffer count for POOL7 is four.

Pool usage specification in the parameter is: D for Database, DE for Database Exact Fit, and S for scratch files. The B specification for both Database and Scratch files is not valid when using Version 1. If you specify B or S usage for POOL 0 through 6, it is changed to D; if you specify B or D usage for POOL7, it is changed to S.

If you do not specify pool parameters, defaults are assigned. The default buffer size for Database (POOL1) and Scratch (POOL7) is 26624; the number of buffers is USERS * 3. A minimum of 10 buffers is assigned to each pool. The number of buffers that are assigned for S2KUSERS (POOL0) is TPTHREADS + 1; the default size is 12288. If you specify one or more POOL parameters, no defaults are used, even if your specification is inadequate.

CISIZE and Buffers

CISIZE is always used to show buffer and page sizes. Usually, you do not need to consider how many bytes of data can be held in one control interval or the actual size of a GETMAIN area for buffers. However, that information is presented here for clarification.

CISIZE is the size given to VSAM. VSAM uses 7 bytes of that area to record VSAM information. Additionally, SYSTEM 2000 needs 12 bytes when the page is logged to File 8. This results in 19 bytes of the CISIZE not available for data, and that number is rounded up to 20. Consequently, for any VSAM database CISIZE, the actual data length is CISIZE − 20. The data length for PAD files is CISIZE − 8. The CISIZE of File 8 must equal the largest CISIZE of Files 1 through 6. The CISIZE of Files 1 through 7 can be any of the sizes that are recognized by SYSTEM 2000. The actual buffer size for databases is CISIZE +12; for the PAD file, it is CISIZE + 8.

Table 5.1 CISIZE and Page Size Values

CISIZE	Database File PAGE SIZE	Scratch File PAGE SIZE	Sort File PAGE SIZE	% used trk space on 3380 VSAM / SORT	% used trk space on 3390 VSAM / SORT
4096	4076	4088	2044	86 / 77	85 / 76
7168	7148	7160	3580	90 / 83	86 / 82
12288	12268	12280	6140	90 / 90	85 / 87
18432	18412	18424	9212	90 / 78	95 / 81
22528	22508	22520	11260	94 / 95	87 / 79
26624	26604	26616	13308	84 / 84	92 / 92

VSAM Files

Version 1 is the first SYSTEM 2000 release to use VSAM file types. All database PAD and S2KUSERS files are now VSAM ESDS. Improved Control Interval (ICI) processing (shorter I/O path) is used in single-user mode for database files. Depending on the disk device, in some instances CISIZE does not qualify for ICI because the CISIZE and disk block size must be the same. Moreover, ICI is not used in Multi-User mode because the ICI process does not allow overlapped I/O.

Table 5.2 shows the disk type, the CISIZE, and whether the CISIZE is ICI compatible.

Table 5.2 ICI Table

CISIZE	3380	3390
4096	Y	Y
7168	Y	Y
12288	N	Y
18432	N	Y
22528	Y	N
26624	N	Y

How Many Buffers

All work files are allocated by the user, and almost any database activity requires that one or more work files be assigned to the user who is making the request. The number of work files that are assigned to a user at one time (excluding Locate Files) varies but usually is in the range from 1 to 4. When a command is completed, the files are released. Consequently, the files are assigned for only a short period of time. One buffer from POOL7 is locked to each assigned work file.

A buffer is assigned and locked for each active Locate File. The file remains active until job termination, or until you issue a LOCATE command that does not qualify any records.

A guideline for the number of work buffers is USERS × 4 = number of buffers. USERS is the value of the USERS parameter. All users will not simultaneously have the maximum number of work buffers assigned, nor will every user have a command in progress, simultaneously. Consequently, if memory usage is a concern, you might want to allocate less than 4 buffers per user. For example, if your USERS parameter exceeds 40, you might want to allocate 2 buffers per user. The minimum buffer count for POOL7 is 4. Failure to provide at least 4 buffers causes user abend 101.

Some activities may require more than 4 buffers. For example, initial loads and incremental loads may use 5 buffers. The REPORT processor may use up to 10 scratch buffers. The number of buffers required in a DEFINE session is based on MAXCOMP (maximum items for the database); the LDBSIZE execution parameter; and the buffer size. Define processing is done in two phases. Phase 1 builds a temporary definition. Phase 2, at MAP time, finalizes the definition and stores it in database Files 1 and 3.

Phase 1 uses buffers from the largest pool for database usage. The following formula is used to calculate the number of buffers needed. In this formula, bufsize is CISIZE −20. The additional 6 buffers (+ 4 + 2) do not necessarily come from the same pool; 4 of the buffers are for Scratch files (POOL7) and 2 are for database Files 1 and 3.

(((MAXCOMP × 28) / bufsize) + 4 + 2)

Phase 2 requires the following number of bytes:

(LDBSIZE × 4)

If 1 or 2 buffers (from any pool with any usage) are adequate, that pool is used. If more than 2 buffers are needed, they are acquired from the largest pool for database usage. Buffers that are acquired in phase 1 are freed before phase 2, and they may be re-used in phase 2.

Any user dispatched for work by Multi-User must be assigned a thread. Additionally, if it is a TP user, a TPTHREADS slot must be assigned, which is a TP JOBQ entry; a TP buffer is also assigned. Therefore, the number of buffers for the S2KUSERS file (CISIZE 12288) should be no less than the lesser of TPTHREADS or the THREADS parameter. You can allocate more buffers to preclude buffer swaps as TP users switch from inactive to active status. However, you should not allocate too few buffers.

Database buffers are assigned by database. Multiple users accessing a specific database page all use the same database buffer. However, multiple users accessing a database, increases the likelihood of access to different database pages, with each page requiring a separate buffer. When a buffer is assigned to a database page, it remains assigned until it is required for other use. Database buffers are released and re-assigned only when a buffer is needed and there are no unused buffers. In that instance, the buffer that has not been used for the longest period of time is re-assigned. File 7 is an exception to this rule. Buffer usage for File 7 is similar to that for work files. Specifically, a buffer is assigned and locked when the file is opened. Processing of File 7 is always sequential, when a buffer becomes full it is written to File 7, released, and another buffer assigned.

For each database that has significant usage, there should be, at least, 5 buffers assigned for each File 5 and 6 and, at least, 4 buffers for each File 2, 3, and 4. Databases that only have occasional activity will share these buffers. You can use fewer buffers, but your I/O count will increase.

File 8 buffer usage is minimal, and you do not need to plan for it. When the database is opened or rollback is enabled, 1 buffer is assigned. The buffer is available for re-assignment when the Rollback Log is confirmed as valid. When rollback occurs, 1 buffer is assigned. The buffer is released when rollback completes. When possible, writes to File 8 use the buffer of the page that is being logged. If the page being logged has a CISIZE that is less than the CISIZE of File 8, then a buffer is assigned to File 8.

For example, you have a large number of databases open in Multi-User, some that have limited activity and 6 that have moderate-to-heavy activity, Files 2 through 6 have the most I/O and, therefore, the most buffer usage. While you cannot assign buffers by file, you can have a different CISIZE and assign buffers for exact fit. You might allocate the CISIZE and buffers as follows:

File 1 and File 2 = 4096. POOL0=4096/20/DE

File 4 = 7168. POOL1=7168/20/DE

S2KUSERS = 12288. POOL2=12288/15/DE

File 5 = 18432. POOL3=18432/30/DE

File 3 = 22528. POOL4=22528/20/DE

File 6 = 26624. POOL5=26624/30/DE

Alternatively, you may choose a database CISIZE that is the same for all database files. Example is to use the largest CISIZE available (22528 for 3380's and 26624 for 3390's). Using the largest size may reduce the I/O operations that are required. In the previous example, the multiple pools could have a single pool of 26624 with 76 buffers and use the same amount of memory. All database pools allocate memory above the 16meg line; therefore, you may allocate even more buffers with Version 1.

Chapter 6: Example Job Streams

Overview

This chapter presents example job streams for various tasks that you might want to perform. All JCL examples are customized and generated at your site in your Version 1 CNTL library.

Multi-User Job Stream (XMS)

This job stream uses dynamic allocation for all database files and most work files. This example is for a batch job. To change this job to a started task, remove the following:

- jobcard
- PEND statement
- EXEC MUMAIN statement.

```
//JCLMU JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//*************************************************************
//* INITIATE MULTI-USER MAIN TASK
//*************************************************************
//*
//MUMAIN   PROC SYSOUT=A
//S2K      EXEC PGM=S2KCMC,REGION=0M
//STEPLIB  DD   DSN=S2K.V1.AUTH,DISP=SHR
//S2KLOAD  DD   DSN=S2K.V1.LOAD,DISP=SHR
//S2KCOM   DD   DSN=S2K.V1.COM,DISP=SHR
//S2KDBCNT DD   DSN=S2K.V1.CNTL(S2KDBCNT),DISP=SHR
//S2KPARMS DD   DSN=S2K.V1.CNTL(MUPARM),DISP=SHR
//S2KDIAG  DD   SYSOUT=&SYSOUT
//SYSOUT   DD   SYSOUT=&SYSOUT
//SYSUDUMP DD   SYSOUT=&SYSOUT
//*
//*************************************************************
//* FILES FOR MULTI-USER ACCOUNTING LOG
//*************************************************************
//*
//S2KMANX  DD   DSN=S2K.V1.ACCOUNT.MANX,DISP=SHR
//S2KMANY  DD   DSN=S2K.V1.ACCOUNT.MANY,DISP=SHR
//*
//         PEND
//*
//*************************************************************
//* INITIATE MULTI-USER MAIN TASK
//*************************************************************
//*
//         EXEC MUMAIN
//*
```

Multi-User SCF Batch

In this Multi-User SCF batch job, the S2KMSG file is dynamically allocated. The job can be used for both SVC and XMS versions.

```
//JCLMUDEP JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//MUTEST   PROC SYSOUT=A
//TESTSTEP EXEC PGM=SYS2KJOB
//STEPLIB  DD DSN=S2K.V1.LOAD,DISP=SHR
//S2KCOM   DD DSN=S2K.V1.COM,DISP=SHR
//SYSUDUMP DD  SYSOUT=&SYSOUT
//S2KCOMD  DD  DDNAME=SYSIN
//         PEND
//*
//         EXEC MUTEST
//SYSIN    DD   *
USER,DEMO: DBN IS EMPLOYEE:
EXIT:
//*
```

Batch Single-User Job Stream (with Dynamic Allocation)

This is an example of a single-user batch job that allows dynamic allocation to allocate required files. The PREFIX parameter in the EXECUTE statement is needed to dynamically allocate the S2KPARMS file.

```
//JCLSCF JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//*************************************************************
//* SELF-CONTAINED FACILITY
//*************************************************************
//*
//S2KPROC  PROC SYSOUT=A
//GO       EXEC PGM=SYS2K,PARM='PREFIX=S2K.V1'
//STEPLIB  DD   DSN=S2K.V1.LOAD,DISP=SHR
//SYSUDUMP DD   SYSOUT=&SYSOUT
//         PEND
//*
//*************************************************************
//* TEST SELF-CONTAINED FACILITY
//*************************************************************
//*
//SCF1     EXEC S2KPROC
//S2KCOMD  DD   *
USER,LIB:
DBN IS LIBRARY:
EXIT:
//*
```

Batch SCF Single-User Job Stream (without Dynamic Allocation)

This is an example of not using a PAD file or any dynamically allocated files. It also shows the S2KPARMS file in the job stream, rather than in a sequential disk file or as a member of a PDS.

```
//JCLSCF JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//*************************************************************
//* SELF-CONTAINED FACILITY
//*************************************************************
//*
//S2KPROC  PROC SYSOUT=A
//GO       EXEC PGM=SYS2K
//STEPLIB  DD   DSN=S2K.V1.LOAD,DISP=SHR
//SYSUDUMP DD   SYSOUT=&SYSOUT
//S2KSNAP  DD   SYSOUT=&SYSOUT
//S2KMSG   DD   SYSOUT=&SYSOUT
//S2KSYS01 DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//S2KSYS02 DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//S2KSYS03 DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//S2KSYS04 DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//S2KSYS05 DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//S2KSYS06 DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//S2KSYS07 DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//SF01     DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//SF02     DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//SF03     DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//SF04     DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//SF05     DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//SF06     DD   UNIT=SYSDA,SPACE=(CYL(1,1))
//LIBRARY1 DD   DSN=S2K.V1.LIBRARY1,DISP=OLD
//LIBRARY2 DD   DSN=S2K.V1.LIBRARY2,DISP=OLD
//LIBRARY3 DD   DSN=S2K.V1.LIBRARY3,DISP=OLD
//LIBRARY4 DD   DSN=S2K.V1.LIBRARY4,DISP=OLD
//LIBRARY5 DD   DSN=S2K.V1.LIBRARY5,DISP=OLD
//LIBRARY6 DD   DSN=S2K.V1.LIBRARY6,DISP=OLD
//LIBRARY7 DD   DSN=S2K.V1.LIBRARY7,DISP=OLD
//LIBRARY8 DD   DSN=S2K.V1.LIBRARY8,DISP=OLD
//LIBRARYS DD   DSN=S2K.V1.LIBRARYS,DISP=OLD
//LIBRARYK DD   DSN=S2K.V1.LIBRARYK,DISP=OLD,
//         UNIT=(CART,,DEFER),LABEL=(,SL,RETPD=5)
           VOL=SER=M12345
//         PEND
//*
//SCF1     EXEC S2KPROC
//S2KPARMS DD   *
POOL0=18432/20/D
POOL7=26624/10/S
PAD00=NO
//S2KCOMD  DD   *
USER,LIB:
DBN IS LIBRARY:
EXIT:
//*
```

Pre-Compile, Compile, Link, and Execute

This job stream has four steps: PRCOM for pre-compile; COB, for COBOL compile; LKED, for linking the program; and GO, for program execution. The mode (single-user or Multi-User) depends on the value of the USE option in the START S2K statement in the program. If single-user mode, the database files are dynamically allocated because no files are specified.

```
//JCLCOB JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//************************************************************
//*  COBOL PLEX COMPILE, LOAD AND GO
//************************************************************
//*
//S2KCLC   PROC SYSOUT=A,COBNAME=SASCBCE,
//             WRKUNIT=SYSDA,PARMS=NLPARM
//PRECOM   EXEC PGM=PRCOMC
//STEPLIB  DD   DSN=S2K.V1.LOAD,DISP=SHR
//SYSPRINT DD   SYSOUT=&SYSOUT
//SYSIN    DD   DSN=S2K.V1.SOURCE(&COBNAME),
//             DISP=SHR
//SYSGO    DD   DSN=&&CSOURCE,DISP=(MOD,PASS),UNIT=SYSDA,
//             DCB=(BLKSIZE=800,LRECL=80,RECFM=FB),
//             SPACE=(CYL,(1,1))
//SYSUT1   DD   UNIT=&WRKUNIT,SPACE=(TRK,(20,20))
//SYSUT2   DD   UNIT=&WRKUNIT,SPACE=(TRK,(20,20))
//COB      EXEC PGM=IGYCRCTL,
//         PARM=(NONUM,APOST,NODYNAM),
//         COND=(4,LT,PRECOM)
//SYSPRINT DD   SYSOUT=&SYSOUT
//SYSUT1   DD   SPACE=(460,(700,100)),UNIT=&WRKUNIT
//SYSUT2   DD   SPACE=(460,(700,100)),UNIT=&WRKUNIT
//SYSUT3   DD   SPACE=(460,(700,100)),UNIT=&WRKUNIT
//SYSUT4   DD   SPACE=(460,(700,100)),UNIT=&WRKUNIT
//SYSUT5   DD   SPACE=(460,(700,100)),UNIT=&WRKUNIT
//SYSUT6   DD   SPACE=(460,(700,100)),UNIT=&WRKUNIT
//SYSUT7   DD   SPACE=(460,(700,100)),UNIT=&WRKUNIT
//SYSLIB   DD   DSN=S2K.V1.SOURCE,DISP=SHR
//SYSIN    DD   DSN=&&CSOURCE,DISP=(OLD,DELETE)
//SYSLIN   DD   DSN=&&COBOBJ,DISP=(NEW,PASS),
//             SPACE=(80,(900,100)),UNIT=&WRKUNIT
//LKED     EXEC PGM=IEWL,
//             PARM='MAP,XREF,LIST,LET,SIZE=(492K,24K)',
//             COND=((4,LT,PRECOM),(8,LT,COB))
//SYSLIN   DD   DSN=&&COBOBJ,DISP=(OLD,DELETE)
//SYSLIB   DD   DSN=IBM.VSCOBOL2.V1R3M1.COB2LIB,DISP=SHR
//         DD   DSN=S2K.V1.LOAD,DISP=SHR
//SYSLMOD  DD   DSN=S2K.V1.LOAD(&COBNAME),
//             DISP=SHR
//SYSPRINT DD   SYSOUT=&SYSOUT
//SYSUT1   DD   SPACE=(1024,(400,20)),UNIT=&WRKUNIT
//GO       EXEC PGM=*.LKED.SYSLMOD,
//             COND=((4,LT,PRECOM),(8,LT,COB),(8,LT,LKED))
//STEPLIB  DD   DSN=S2K.V1.LOAD,DISP=SHR
//S2KPARMS DD   DSN=S2K.V1.CNTL(&PARMS),DISP=SHR
```

```
//CBREAD   DD   DSN=S2K.V1.SOURCE(PLIINPUT),
//              DISP=SHR
//CBPRINT  DD   DCB=(BLKSIZE=1320,LRECL=132,RECFM=FB),
//              SYSOUT=&SYSOUT
//SYSOUT   DD   SYSOUT=&SYSOUT
//SYSUDUMP DD   SYSOUT=&SYSOUT
//S2KCOMD  DD   DUMMY
//         PEND
//*
//     EXEC S2KCLC
```

S2OP Batch Jobstream

This example job is for batch execution of the alternate console program. Five input commands following the DD statement S2KCOMD are shown.

```
//JCLS2OP JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//*************************************************************
//* EXECUTE SYSTEM 2000 EXTERNAL CONSOLE
//*************************************************************
//*
//S2OP     EXEC PGM=S2OP
//STEPLIB  DD DSN=S2K.V1.LOAD,DISP=SHR
//S2KCOM   DD DSN=S2K.V1.COM,DISP=SHR
//S2KMSG   DD  DCB=(LRECL=132,BLKSIZE=132,RECFM=FBA),
//         SYSOUT=A
//S2KCOMD  DD  *
D A,S
#DBSTAT=EMPLOYEE
#QUEUES
#POOLS
#THREADS
//*
```

IDCAMS Job to Define Database Files

CONTROLINTERVALSIZ
CISZ
specifies the size of the control interval. This must be a valid CISIZE in SYSTEM 2000.

RECORDSIZE
specifies the average and maximum lengths, in bytes, of the records in the data component. SYSTEM 2000 uses fixed-length records and requires that the average and maximum lengths be the same. In addition, the RECORDSIZE must be exactly 7 bytes less than CISIZE.

NONINDEXED
specifies that the file is an entry-sequenced data set (ESDS).

REUSE
specifies that the file can be re-opened as a re-usable data set. Only database Files 1 through 6 require this attribute.

SHAREOPTIONS
SHR
specifies the level of sharing that is allowed by VSAM. SYSTEM 2000 requires a cross-region value of 2.

```
//JCLALLOC JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//************************************************************
//* ALLOCATE DATABASE FILES USING IDCAMS
//*
//* NOTES:
//*    CISIZE MUST BE A VALID SYSTEM 2000 CISIZE
//*    RECORDSIZE MUST BE 7 BYTES LESS THAN CISIZE.
//*
//************************************************************
//*
//VSAM     PROC SYSOUT=A
//         EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=&SYSOUT
//         PEND
//         EXEC VSAM
//SYSIN    DD *
 DEFINE CLUSTER                    +
   (NAME('S2K.V1.DBNAME1')      +
   NONINDEXED                       +
   VOL(DISK01)                      +
   CYLINDERS(1 1)                   +
   CISZ(4096)                      +
   RECORDSIZE(4089 4089)             +
   SHR(2,3)                         +
   REUSE                            +
   SPEED )                          +
   DATA(NAME('S2K.V1.DBNAME1.DATA'))
```

```
DEFINE CLUSTER                        +
  (NAME('S2K.V1.DBNAME2')       +
  NONINDEXED                          +
  VOL(DISK01)                         +
  CYLINDERS(1 1)                      +
  CISZ(4096)                         +
  RECORDSIZE(4089 4089)               +
  SHR(2,3)                            +
  REUSE                               +
  SPEED )                             +
  DATA(NAME('S2K.V1.DBNAME2.DATA'))

DEFINE CLUSTER                        +
  (NAME('S2K.V1.DBNAME3')      +
  NONINDEXED                          +
  VOL(DISK01)                         +
  CYLINDERS(1 1)                      +
  CISZ(4096)                         +
  RECORDSIZE(4089 4089)               +
  SHR(2,3)                            +
  REUSE                               +
  SPEED )                             +
  DATA(NAME('S2K.V1.DBNAME3.DATA'))

DEFINE CLUSTER                        +
  (NAME('S2K.V1.DBNAME4')      +
  NONINDEXED                          +
  VOL(DISK01)                         +
  CYLINDERS(1 1)                      +
  CISZ(4096)                         +
  RECORDSIZE(4089 4089)               +
  SHR(2,3)                            +
  REUSE                               +
  SPEED )                             +
  DATA(NAME('S2K.V1.DBNAME4.DATA'))

DEFINE CLUSTER                        +
  (NAME('S2K.V1.DBNAME5')      +
  NONINDEXED                          +
  VOL(DISK01)                         +
  CYLINDERS(1 1)                      +
  CISZ(4096)                         +
  RECORDSIZE(4089 4089)               +
  SHR(2,3)                            +
  REUSE                               +
  SPEED )                             +
  DATA(NAME('S2K.V1.DBNAME5.DATA'))

DEFINE CLUSTER                        +
  (NAME('S2K.V1.DBNAME6')      +
  NONINDEXED                          +
  VOL(DISK01)                         +
  CYLINDERS(1 1)                      +
  CISZ(4096)                         +
  RECORDSIZE(4089 4089)               +
  SHR(2,3)                            +
  REUSE                               +
  SPEED )                             +
  DATA(NAME('S2K.V1.DBNAME6.DATA'))
```

```
DEFINE CLUSTER                        +
  (NAME('S2K.V1.DBNAME7')     +
  NONINDEXED                          +
  VOL(DISK01)                         +
  CYLINDERS(1 1)                      +
  CISZ(4096)                         +
  RECORDSIZE(4089 4089)               +
  SHR(2,3)                            +
  REUSE                               +
  SPEED )                             +
  DATA(NAME('S2K.V1.DBNAME7.DATA'))

DEFINE CLUSTER                        +
  (NAME('S2K.V1.DBNAME8')     +
  NONINDEXED                          +
  VOL(DISK01)                         +
  CYLINDERS(1 1)                      +
  CISZ(4096)                    +
  RECORDSIZE(4089 4089)   +
  SHR(2,3)                            +
  REUSE                               +
  SPEED )                             +
  DATA(NAME('S2K.V1.DBNAME8.DATA'))

/*
```

IDCAMS Job to Define Other Required VSAM Files

This example defines S2KPAD00, S2KUSERS, and S2KLIB. The S2KUSERS file must be CISIZE 12288.

```
//JCLVSAM JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//************************************************************
//* ALLOCATE S2KPAD00 VSAM FILES USING IDCAMS
//*
//* NOTES:
//*    CISIZE MUST BE A VALID SYSTEM 2000 CISIZE
//*    RECORDSIZE MUST BE 7 BYTES LESS THAN CISIZE.
//*
//* VALID CISIZES AND RECORDSIZES:
//*          4096           4089
//*          7168           7161
//*         12288          12281
//*         18432          18425
//*         22528          22521
//*         26624          26617
//*
//*  ALLOCATE THE MULTI-USER S2KUSERS FILE.
//*
//*  THIS FILE MUST BE CISIZE 12288.
//*
//*
//* ALLOCATE CICS COMMAND EDITOR AND
//* PF KEY DEFINITION VSAM FILE
//*
//************************************************************
```

```
//*
//VSAM     PROC SYSOUT=A
//         EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=&SYSOUT
//         PEND
//         EXEC VSAM
//SYSIN    DD *

 DELETE S2K.V1.S2KPAD00 CLUSTER
 DEFINE CLUSTER                      +
   (NAME('S2K.V1.S2KPAD00')     +
   NONINDEXED                        +
   VOL(DISK01)                       +
   CYLINDERS(5 10)                    +
   CISZ(26624)                    +
   RECORDSIZE(26617 26617)   +
   SHR(2,3)                          +
   SPEED )                           +
   DATA(NAME('S2K.V1.S2KPAD00.DATA'))

 DELETE S2K.V1.S2KUSERS CLUSTER
 DEFINE CLUSTER                      +
   (NAME('S2K.V1.S2KUSERS')     +
   NONINDEXED                        +
   VOL(DISK01)                       +
   CYLINDERS(5 1)                    +
   CISZ(12288)                    +
   RECORDSIZE(12281 12281)   +
SHR(2,3)                          +
   REUSE                             +
   SPEED )                           +
   DATA(NAME('S2K.V1.S2KUSERS.DATA'))

 DELETE S2K.V1.S2KLIB CLUSTER
 DEFINE CLUSTER -
         (NAME(S2K.V1.S2KLIB) -
         VOLUMES(DISK01) -
         RECORDSIZE(2000 19000) -
         REUSE -
         CYLINDERS(5 0) -
         KEYS(12 0) -
         FREESPACE(20 20) -
         SHR(3 3) )
// EXEC VSAM
//DUMMREC DD *
DUMMY
//SYSIN DD *
          REPRO-
          INFILE(DUMMREC)-
          OUTDATASET(S2K.V1.S2KLIB)
/*
```

Link SYS2K with or without S2EXIT

```
//JCLS2KLN JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//*************************************************************
//* RELINK EXECUTABLE MODULE SYS2K WITH S2EXIT
//*************************************************************
//*
//S2KLINK  PROC SYSOUT=A,WRKUNIT=SYSDA
//LKED     EXEC PGM=IEWL,
//    PARM=(MAP,XREF,LET,LIST,NCAL,'SIZE=(256K,64K)',
//    'AMODE=31','RMODE=ANY')
//SYSLMOD  DD  DSN=S2K.V1.LOAD(&LNAME),DISP=SHR
//SYSLIN   DD  DDNAME=SYSIN
//LOAD     DD  DSN=S2K.V1.LOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=&SYSOUT
//SYSUT1   DD  UNIT=&WRKUNIT,SPACE=(1024,(400,20))
//         PEND
//*
// EXEC S2KLINK,LNAME=SYS2K
//SYSIN DD *
 REPLACE S2EXIT
 INCLUDE SYSLIB(SYS2K)
 INCLUDE SYSLIB(S2EXIT)
 ENTRY S2K
 NAME SYS2K
//*
//*************************************************************
//* RELINK EXECUTABLE MODULE SYS2K WITHOUT S2EXIT
//*************************************************************
//*
//* TO DELETE S2EXIT FROM SYS2K, REMOVE THE INCLUDE STATEMENT
//* FOR S2EXIT.  THE NCAL PARAMETER IS NECESSARY TO PREVENT
//* S2EXIT FROM BEING AUTOMATICALLY INCLUDED.
```

Chapter 7: Dynamic File Allocation

Overview

Dynamic file allocation allows your jobs to execute with minimal JCL statements, and it enables you to take a database offline in Multi-User and to lock it offline while you work with that database in single-user mode. File allocation techniques are the same in single-user and in Multi-User modes. Both database and work files can be dynamically allocated. You can use JCL, TSO ALLOC commands, and user exit 2 (your own dynamic allocation of files), or any combination of these methods with SYSTEM 2000 dynamic allocation.

The dataset name (DSN) of the file being allocated must be known in order to do dynamic allocation. S2KDBCNT (database files only) and the value of the execution parameter PREFIX are used by SYSTEM 2000 to derive the DSN. S2KDBCNT is a sequential file created by you that contains database names and dataset names. The value of PREFIX is a partial DSN that is prefixed to the data definition name (DDname) or database name to form a DSN. S2KDBCNT and PREFIX are described in the following paragraphs.

For database files, SYSTEM 2000 derives the DSN by using the S2KDBCNT file. If the database is not listed in S2KDBCNT, the *prefix* value is placed before the first seven characters of the database name. A one-character suffix is added for file identification. A blank in the name is replaced with X; however, X is not used to expand the DSN to seven characters. For example, the DSN for File 1 of database A B C is constructed as *prefix.*AXBXC1. For the Savefile, it is *prefix.*AXBXCS.

For work files**,** the value for *prefix* is placed before the required DDname to derive a DSN. For example, the constructed DSN for the file S2KPAD00 is *prefix.*S2KPAD00. If there is no value for *prefix* , the DDname and the DSN are the same. If a file does not exist that matches the constructed DSN, a new file is created. Dynamically created work files are deleted at job termination.

S2KDBCNT File

S2KDBCNT is a sequential file in which you enter the database name and a partial DSN that is used for construction of the DSN when the files are allocated. The DSN is constructed by retrieving your specified partial DSN from the file and appending an appropriate suffix: 1 for database File 1, 2 for database File 2, S for Savefile, and so on. For example, if the database name is EMPLOYEE, the partial DSN that you enter in the S2KDBCNT file might be S2K.V1.EMPLOYE. Then, in files that are allocated for the EMPLOYEE database, the DSN used for File 1 is S2K.V1.EMPLOYE1.

For dynamic allocation of database files, the last character of the DSN must be a character (1 through 8, S, or K) that is the same as the last character of the DDname that it is associated with. The DSN must be complete, except for the last character. No other sources are used for the databases listed in S2KDBCNT.

The LRECL value of S2KDBCNT is 80. The number of entries in the file can range from 1 to 100. Table 7.1 shows each field for an entry in the file.

Table 7.1

Field	Beginning Column	Length	Description
DBN	1	16	Database name
OFFLINE	17	1	Flag that shows whether the database is online or offline. Usually, you will leave this field blank, but you can pre-set it. 1 indicates offline because of the FREE command. 2 indicates offline because of the VARY command. Blank indicates the database is available.
DSN	18	44	DSN is for database files. All files have the same name except for the last character.
DISP	62	3	Disposition is used for dynamic allocation, that is, OLD or SHR. The default is OLD.

Note: The DSN entry is required. If you omit it, the DSN cannot be constructed.

You specify whether S2KDBCNT is to be used with the parameter ALLOC. When ALLOC=YES, the file may be used, but it is not required. If the file is specified in JCL with the DDname S2KDBCNT, or if the file exists with DSN *prefix*.S2KDBCNT, it will be used; otherwise, it will not be used. When ALLOC=TBL, you must provide S2KDBCNT.

Dynamic Allocation Parameters

There are several execution-time parameters that relate to dynamic file allocation. Those parameters are described in "Execution Parameters" in Chapter 14. Three of the parameters are also described here in order to give you a better understanding of what is required to implement dynamic allocation. Additionally, all parameters related to dynamic allocation are listed here.

PREFIX
: the value of the execution parameter PREFIX applies to all files not listed in S2KDBCNT and not specified in JCL. The value is prefixed to the first 7 characters of the database name to construct the DSN or to the DDname for work files. Examples are *prefix*.S2KPAD00, *prefix*.S2KUSERS, *prefix*.SF*nn* for sort files, and *prefix*.S2KDBCNT for the control file. In order to dynamically allocate the S2KPARMS file, you need to specify *prefix* in the EXEC statement.

ALLOC=YES|NO|TBL
: specifies whether dynamic allocation is allowed. YES indicates that dynamic allocation is allowed. The S2KDBCNT file is optional. NO indicates that dynamic allocation is not allowed. The VARY and FREE commands are not acceptable. TBL indicates that the S2KDBCNT file is required. Only databases listed in S2KDBCNT can be dynamically allocated. Work files can also be dynamically allocated. The default is YES.

FREE =YES|NO
applies to any database that is physically closed and has files dynamically allocated by SYSTEM 2000. YES specifies that files are to be de-allocated when they are physically closed. NO prevents files from being de-allocated except when a CLOSE results from the VARY or the FREE command being executed. The default is NO.

Details for the following parameters are in Chapter 14, "Execution Parameters."

EXPSAVE	PADSEC	SFSPACE
KEEPUNT	PADVOL	SFUNIT
OPT048	RETPRD	SYSOUT
OPT049	SAVEUNT	S2KMSGL
OPT055	SFPRI	TPVOL
PADPRI	SFSEC	

SCF Commands for Dynamic Allocation

The two SCF commands associated with dynamic allocation are ALLOC and FREE. ALLOC allows you to allocate new or existing database files during an SCF session. The syntax for SCF dynamic allocation commands is shown in *CONTROL Language*, *Version 12, First Edition*.

For new files, you can allocate all 8 database files with one command by using the option ALL, or you can allocate each file individually by using the FILES= option.

For existing files, you can allocate all 8 database files with one command. Options are not valid for existing files. Consequently, when you specify DISP=OLD or DISP=SHR, Files 1 through 6 are allocated. Files 7 and 8 are allocated if they exist.

An error message is issued if you invoke the ALLOC command for a database that is listed in S2KDBCNT, and you specify a DSN that is different from that which is listed in the file. If ALLOC=TBL, the ALLOC command is limited to the ALLOC database. No other parameters are allowed. The purpose of this limited command is to allow you to re-allocate a database that has been de-allocated by using the FREE command. If the database is listed in S2KDBCNT, the DSN specified in that file is used. Otherwise, dynamic allocation is not allowed.

The FREE command is similar in functionality to the VARY command. The differences are that FREE is an SCF command that applies only to your current database, which it closes. If the database is identified in S2KDBCNT, the offline lock is set to 1 to show that the database is unavailable. The console commands VARY ONLINE and SCF ALLOC can put the database back in service. If the database is not identified in S2KDBCNT, the files are closed and de-allocated, but there is no offline lock to protect the database. The files can be re-allocated by using the ALLOC, DBN, or NDB command. Files that were allocated by SYSTEM 2000 are de-allocated, regardless of the FREE parameter.

In Multi-User mode, you must open the database for exclusive use. Because you are the only user on the database, closing and de-allocation take place immediately. Exclusive use is implied in single-user mode, therefore if you specify it, it has no effect.

Take Database Offline

You can take a database offline in either single-user or in Multi-User mode. Only the FREE command is available in single-user mode because there is no provision for console communications. The FREE command is described in the preceding section. To take a database offline in Multi-User, use the console commands VARY <dbn> OFFLINE and SCF FREE <dbn>.

VARY OFFLINE is the console or alternate console command that allows you to force the physical close and de-allocation of a database when the usage count drops to 0, regardless of the FREE parameter. Only files dynamically allocated by SYSTEM 2000 are de-allocated. Additionally, if the database is listed in S2KDBCNT, the offline lock is set to 2. The lock prevents the files from being re-allocated until the VARY ONLINE command is invoked. When lock equals 2, the SCF ALLOC command cannot return the database to service. The lock serves as a pending indicator when users are in the database and it cannot be closed. The lock is set when the VARY OFFLINE command is invoked; it directs that the database be closed and de-allocated when the usage count is 0.

In comparison, if the database is not in S2KDBCNT, you can vary the database offline, but it is not under the protection of database lockout. You can open the database anytime without an intervening VARY ONLINE command. Also, the usage count must be 0 when the command is invoked, otherwise the result is an error message and the database is not closed. There is no dependence on the FREE parameter for de-allocation of files initiated by the VARY command.

Put Database Online

After you have taken a database offline that is listed in S2KDBCNT, you can re-activate the database with the VARY ONLINE console command or the SCF ALLOC command. The ALLOC command is described earlier.

The VARY ONLINE command puts a database in service in the session where it was taken offline by using the FREE or the VARY OFFLINE command.

Physical Database Closes

Databases in Multi-User are closed only in response to the FREE or the VARY OFFLINE command, and a directed close from user EXIT36. You can specify execution parameter OPT055=YES to cause databases to be closed if the files were dynamically allocated by SYSTEM 2000 and the database is not listed in S2KDBCNT.

In single-user mode, S2KDBCNT has no effect on physical closes. De-allocation occurs only if you specify FREE=YES, and the files were dynamically allocated by SYSTEM 2000. Physical closes occur in single-user mode when the following commands are issued:

The DBN and NDB (SCF commands) invoke a close for your current database before initiating an open for the specified database.

The RELEASE command deletes the database. It also invokes the close process for the database files.

The EXIT (SCF command) executes sign off from SYSTEM 2000 and causes your database to be closed.

The CLOSE (PLEX command) executes the closing process. In a single-user PLEX job, you can have a maximum of 63 databases open; only the commands CLOSE <*database*> and STOP S2K invoke the closing process.

Chapter 8: User Exits

Overview

A user exit is a user-written routine that is called by SYSTEM 2000 at selected points in a process. It is SYSTEM 2000 that invokes the action identified by the exit. At selected points within SYSTEM 2000, checks are made to determine if a specific exit is enabled. If enabled, that exit is called. Addresses of SYSTEM 2000 data areas that your exit can interrogate or modify are passed in a parameter list. When your exit returns to SYSTEM 2000, a return code (action code) identifies the action taken by the exit. Your exit code can

- accept or reject a command
- change the command
- terminate the job making the request
- write SMF records
- perform many other job management tasks that you choose.

Examples of when exits are executed include: before a physical open of a database (EXIT03), before SCF TP input commands are parsed (EXIT13), and before unauthorized attempts to access data (EXIT38).

S2EXIT is a supplied routine that is the interface between SYSTEM 2000 and your exits. S2EXIT builds a parameter list for the specified exit and calls the exit. You allow exit processing by using the execution parameter EXITS=YES or by linking S2EXIT with SYS2K. The execution parameter is ignored if S2EXIT is linked with SYS2K. If exits are enabled, EXIT00 is invoked during initialization (module and CSECT name must be S2KEXIN). S2KEXIN is loaded by S2EXIT unless it is linked with S2EXIT or SYS2K.

S2KEXIN determines which exits are enabled and stores that information in a data area named AVAILEXT. The address of the exit is derived by using a VCON or a LOAD macro. Macro SETEXIT is provided for use with S2KEXIN. It provides a convenient way to enable an exit. Example exits are provided on the delivery media with the name S2KEXnn where *nn* is the exit number. Here is an example that enables EXIT45. LOAD=EXIT45 means that the variable EXIT45 contains the address of exit 45.

```
SETEXIT 45,ON,LOAD=EXIT45
```

Link S2EXIT

S2EXIT is delivered to you in LOAD module format. You can use it as is, or you can link it with SYS2K. The result of linking it with SYS2K is that any use of SYSTEM 2000 in Multi-User or single-user mode invokes your exits. The exits cannot be disabled with execution parameter EXITS=NO.

You also can link all exits with S2KEXIN and link S2KEXIN with S2EXIT. If you use the S2KEXIN example in this document, all exits are linked with S2KEXIN by using the VCON references in the linkage editor step. Example JCL to assemble and link user exits is job JCLEXIT. JCL to link S2EXIT with SYS2K is job JCLS2KLN. Both jobs are generated at your site during installation.

ENABLE/DISABLE ROUTINE Commands

You can enable or disable any exits other than EXIT00, EXIT01, and EXIT02 by using the ENABLE or DISABLE ROUTINE commands. You must provide EXIT01 before using these commands. EXIT01 alters AVAILEXT in response to the command. Syntax of the command is documented in *SYSTEM 2000 CONTROL Language, Version 12* and in the *SYSTEM 2000 PLEX Manual, Version 12.*

Parameter List

The parameter list is mapped by using the macro DEXTPARM, which generates DSECT DUSRPRM. Data area addresses that are not applicable to a specific exit are set to 0.

DUSRPRM begins when the label TEMPSTOR is used. The data areas TEMPSTOR through TAPEOP are contiguous in memory. They may be used for every exit. Addresses to other areas are contained in the address list that follows the label TAPEOP. The name of each field begins with the letter A (to indicate address) and is completed with a name that is descriptive of the area that it points to. An example is ADBNAME, which points to the current database name.

When your exit is called, Register 1 (R1) points to AUSRCTL, which is within DUSRPRM. AUSRCTL is the first label in the address list. Immediately following AUSRCTL is ATMPSTOR, which is the address of a temporary storage area named TEMPSTOR. You load R1 (or another register) by using displacement 4 off of R1 to point to TEMPSTOR, which is the first label in DUSRPRM, and you specify USING DUSRPRM,R1. You now have addressability to the entire parameter list that is mapped by DUSRPRM and can refer to an individual area by label name.

Action Codes

You must store a value in the field ACTION (within DUSRPRM) before control is returned to S2EXIT. ACTION is an indicator to S2EXIT of processing requirements specified by the exit. Not all action codes are available to all user exits. See the description for each exit (later in this chapter) to determine which codes are applicable.

Table 8.1 Valid Action Codes

Action Code	Indicates
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 message. Continue processing.
32	Reject output line. Print message. Continue processing.
36	Reject command. Print message. Continue processing.
40	Terminate user. Can damage database.
44	Terminate user. Print user message. Can damage database.
48	Terminate user only if database would be left undamaged.
52	Terminate and print message if database left undamaged.

Temporary Storage

There are two storage areas available to your exits that may be used in any way that you choose. Each area is 400 bytes in length. The areas are contiguous in memory, so 800 bytes are available. The first area is TEMPSTOR (first area in DUSRPRM). The second area is TEMPSTR2. You control data in these areas while an exit is executing. However, because the areas are allocated by thread, they are also available to other exits.

TEMPSTR2 was implemented for EXIT45, but use is not limited to that exit. Each time EXIT45 is called, 400 bytes of data in the URA are moved to TEMPSTR2. If you alter the area in EXIT45 and set the action code to 4 or 12, TEMPSTR2 is copied to the URA. That same data is copied back to TEMPSTR2 the next time EXIT45 is called by that user. The URA is a data area unique to a user and remains assigned to a user until that job terminates.

Messages Issued by User Exits

To send a message to a user S2KMSG file, move the message text to USRBUF and store the length of the message in USRLEN. The maximum length is 120 bytes. You set an action code that specifies that a message is to be printed. Messages issued by an exit usually are assigned a number in the range of 950 through 999, and the number is a part of the message. This helps identify the message. You may want to include the exit name that issued the message. The exact message you put into the message buffer is written to the S2KMSG file for both SCF and PLEX.

PLEX Return Code

Exits 3, 4, 17, and 20 modify a PLEX return code by storing the code that you want in PLIRTN and selecting an action code that specifies the data areas that were modified. It is recommended that you use return codes in the range of 950 through 999 for your exit to ensure that they are distinguishable from those originated by SYSTEM 2000.

AVAILEXT

AVAILEXT is five words in length (160 bits). It is used to show which exits are enabled. An enabled exit has its corresponding bit position in AVAILEXT set to 1. For example, if exits 7, 8, and 11 are enabled, bits set in the first two bytes of AVAILEXT are 00000001 and 10010000. A pointer to this area is in AVALEXIT (DUSRPRM). You load AVALEXIT into a register and then set appropriates bits. For exits 7, 8, and 11, you would see OI AVAILEXT,X'01'; OI AVAILEXT+1,X'80'; OI AVAILEXT+1,X'10' to turn on bits 7, 8, and 11. This area is available only to EXIT00 and EXIT01.

User Exit Load List

The address of each enabled exit is stored in the User Exit Load List area (DSECT LOADLIST). This area, 152 words in length, is indexed by the exit number. For example, 45 for EXIT45 is multiplied by 4 to derive displacement 180 (X'B4'), which is where the address for EXIT45 is stored. AUSRLOAD has the address of the load list. The exit number is in the field OPCODE of DUSRPRM. The User Exit Load List is available only to EXIT00 and EXIT01.

Exit Names

With one exception, you choose the names of your exit CSECT and load module. The exception: EXIT00 must be named S2KEXIN.

31-Bit Mode

Your exit is entered in 31-bit addressing mode, and it returns in 31-bit mode. Additionally, some data areas that are passed in the parameter list are above 16 megabytes and can be accessed only in 31-bit mode.

Re-Entrant Code

Your exits should be written in Assembler language and be re-entrant.

I/O and Waits

Any I/O performed in your exits should be an asynchronous request that is followed by the EXTWAIT and CHECK macros. This process allows your thread to be suspended while the I/O is in progress. Other threads can be dispatched for work while your thread is suspended. The GET and PUT macros do not return control to your program until the I/O is complete, therefore, these macros should be avoided. Use of these I/O macros in Multi-User can increase processing time and reduce throughput. Issuing a message to the user by moving the message to USRBUF is not an I/O by the exit.

Do not use EXTWAIT in EXIT00 and EXIT02. These exits occur during SYSTEM 2000 initialization before the dispatcher is ready to accept work. EXIT00 and EXIT02 can use the WAIT, GET, PUT, and CHECK macros without degrading performance.

Macros

DEXTPARM — This macro provides a DSECT of all addresses and data areas available to any exit, except for AVAILEXT and LOADLIST. These two areas are 5 and 151 fullwords in length, respectively; they are available only to EXIT00 and EXIT01. DSECTs for these two areas are in the macro EXITAREA.

EXITAREA — EXITAREA is a subset of DEXTPARM. Additionally, it establishes DSECTs for AVAILEXT and LOADLIST. These two areas are available only to EXIT00 and EXIT01. AVAILEXT (5 fullwords in length) is a bit mask that identifies available exits. LOADLIST (151 fullwords in length) is a list of addresses of exits that are marked as available in AVAILEXT.
Because labels in EXITAREA and DEXTPARM are the same, only one of these macros can be used in a routine.

EXITBGN — EXITBGN provides pseudo re-entrant code. It equates registers, chains save areas, and provides addressability. This macro has 5 optional parameters. The first 2 parameters are positional; the last 3 are keyword parameters. The default value for the storage parameter is GETMAIN. The default value for STRSIZE is 1024. If you specify EXITBGN with no parameter values, you will have 1024 bytes of storage for your save area and work space.

Syntax for the EXITBGN macro is

label EXITBGN [*length*] [, S] [, WS=*name*]

[,STORAGE=|GETMAIN][,STRSIZE=*nnnn*]
|TEMPSTOR

label is the name of the CSECT to be created.

length is a positional parameter that specifies the number of bytes to be used for working storage for the exit in addition to 72 bytes for a save area. Length plus 72 is added to the beginning of the storage area; that value (beginning of next save area) is stored in the first word of the save area. If you call a subroutine, R13 points to the calling routine's save area. The subroutine loads the first word of R13 to find its save area. The process is repeated as often as it is necessary.

S indicates a subroutine call. Space for the save area and working storage is obtained by loading the word pointed to by R13.

WS = *name* is the DSECT name used for the save area and working storage.

STORAGE = [GETMAIN] [TEMPSTOR] obtains space for the save area and working storage by issuing a GETMAIN. TEMPSTOR specifies that the area TEMPSTOR, which is identified in DUSRPRM, is to be used. The parameter is ignored if you code S as a positional parameter value. GETMAIN is the default.

STRSIZE=*nnnn* indicates the size of the area to be obtained when using GETMAIN. STRSIZE=*nnnn* is ignored if you code the S option or if STORAGE=TEMPSTOR. The default is 1024 bytes.

EXITEND restores the registers of the calling program, sets R15 to 0, and returns to the calling program. This macro has no parameters.

EXTPARM is invoked by DEXTPARM. EXTPARM has the address list area. If you specify DEXTPARM, you automatically get EXTPARM.

EXTWAIT should be invoked after each asynchronous I/O is executed by your exit and before CHECK is issued. This allows other threads to process while your thread is suspended for I/O completion. When your I/O ECB is posted, your thread is dispatched to process CHECK. EXTWAIT has the following parameters whose value identifies a register into which you place an address:

ECB=*n* identifies the register assigned to ECB that must point to a nine-word DECB for the I/O just issued.

WAITEP=*n* identifies a register that has the value stored in AEXTWT. AEXTWT contains the address of an entry point used for I/O waits. If you omit WAITEP=, EXTWAIT loads R15 with AEXTWT.

SETEXIT sets appropriate bits in AVAILEXT to show that the exit is enabled, and it stores the address of the exit in the user exit LOADLIST. It is used in EXIT00. You can load the exit, or you can use a VCON and the CALL option in the linkage step so that the exit will be linked with EXIT00. This example enables exit 7 and specifies that the LOAD module name on disk is EXIT7.

```
SETEXT 7,ON,LOAD=EXIT7

EXIT7   DC   V(EXIT7)
```

See Table 8.2 for a summary of the available exits.

Table 8.2 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.	Enable specified exits.	0, 4
01	After parsing an SCF ENABLE/ DISABLE EXITS command or after a PLEX call resulting from an ENABLE/DISABLE EXITS command.	Enable or disable specified exits.	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).	Dynamic allocation of files.	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 files.	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, 44, 48, 52
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

Table 8.2 Summary of Available Exits (c*ontinued*)

Exit Number	Logical SYSTEM 2000 Exit Point	Position Applications	Allowable Action Codes
14	SCF line of output is ready.	Modifying output line or replacing with alternate input line to other software.	0, 4, 8, 12, 20, 32, 40, 44, 48, 52
15	Parsed SCF input.	Verifying value range to user requirements.	0, 4, 8, 12, 24, 36, 40, 44, 48, 52
16	Formatted SCF output.	Modifying current output value.	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 PLEX schema value.	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 PLEX schema value for output.	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 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 ACB 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 ACB prior to physical close.	0, 4, 8, 12, 40, 44, 48, 52
25	Prior to logical open of a database. (Multi-User only)	Preparing for dynamic de-allocation.	0, 4, 8, 12, 16, 28, 40, 44, 48, 52
26-35	(Reserved for future use.)		
36	Prior to physical close of a database. (Multi-User only)	Preventing Multi-User from physically closing a database.	0, 4, 8, 12, 16, 28
37	After logical close of a database. (Multi-User only)	Companion to EXIT03 or EXIT25.	0, 4, 8, 12

Table 8.2 Summary of Available Exits (*continued*)

Exit Number	Logical SYSTEM 2000 Exit Point	Possible Applications	Allowable Action Codes
38	All authority checkpoints.	Detecting security violations	0
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	Prior to every update command.	Accept or reject command.	0, 16, 28
45	Prior to logical open of a database.	Limit access to read only.	0, 4, 8, 12, 36, 40
46-49	(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 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.

Table 8.3 Exit Parameter (Address List)

Parameter Label	Address of	Field Description	Exits
AUSRCTL	user control block	29 fullwords	
		- exit number (binary)	All
		- execution mode (binary)	All
		0 - SCF	
		1 - PLEX	
		0 - master console command	43
		1 - S2OP console command	43
		- environment (binary)	All
		0 - single-user	
		≠ - Multi-User	
		- action code (binary)	All
		- PLEX return code set by user (binary)	All
		- flag for ENABLE/DISABLE EXITS command (binary)	01
		0 - disable	
		1 - enable	
		- 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	
		- address of user work buffer (set by user exit) or	9-14
		SCF tape indicator:	22-24
		=1 if KEEP or RESTORE	50-53
		>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, 45
APASSWRD[§§]	password	1 fullword (character)	01, 03-25, 39, 45
ADBNAME[§§]	database name	4 fullwords (character)	01, 03-25, 39, 45

[§] 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 and are altered regardless of action code.

Table 8.3 Exit Parameter (Address List) (*continued)*

Parameter Label	Address of	Field Description	Exits
ACCTINFO[§§]	ACCT information (Multi-User only)	8 fullwords (character) - jobname (2) - stepname (2) - program name (2) - terminal ID (for TP only)	03-22, 25, 37, 39, 45
ACOMBLK[§§]	COMMBLOCK Control block	Address not given if program does not use COMMBLOCK	03-07, 09-12, 17-20, 25, 36-38, 45
ASCHEMA[§§]	SSR control block (PLEX only)	Address not given if program does not use SSR	05, 06, 17-20, 38, 45
AS2KDUM[§§]	S2KDUM control block (PLEX only)	See PLEX manual for S2KDUM format	01, 03-07, 09-12, 17-20, 25, 37, 38, 45
ABUFSIZ[§§§]	database page buffer size	1 fullword that contains 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 that contains byte count initially set to max size of 118 characters (binary)	01, 03-63, 45
AERMSGBF[§]	user-message buffer	30 fullwords (message length in AERMGSLN parameter)	01, 03-63 45
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

§ 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 and are altered regardless of action code.

§§§ Buffer sizes must conform to the provided buffer sizes to avoid writing outside of the buffer area.

Table 8.3 Exit Parameter (Address List) (c*ontinued)*

Parameter Label	Address of	Field Description	Exits
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
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, 45, 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

[§§] These fields are used by SYSTEM 2000 and are altered regardless of action code.

[§§§] Buffer sizes must conform to the provided buffer sizes to avoid writing outside of the buffer area.

Table 8.3 Exit Parameter (Address List) (c*ontinued)*

Parameter Label	Address of	Field Description	Exits
AUSRLOAD[§§]	user exit load list	64 fullwords One word per user exit entry point, address	00, 01
ADCB	ACB address	VSAM ACB	23, 24
AEXTGM	Reserved		
ADBIO[§§]	database I/O count	2 fullwords when segment statistics active (packed)	13, 14, 21, 45
ATIM[§§]	elapsed CPU time	1 fullword when segment statistics active (packed)	13, 14, 21, 45
ASEGIO[§§]	database I/O count for Multi-User segments	2 fullwords when segment statistics active (packed)	13, 14, 21, 45
ASEGTIM[§§]	segment elapsed CPU time	1 fullword when segment statistics active (binary)	15, 16, 19, 20, 45
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 is EXIT38	
AFILENO[§§]	database file number	1 character-file number (binary)	05, 06, 23, 24
AECB[§§]	ECB	14 fullwords (see Table A.5; start with the ECBNAM Field Description for details.)	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
ATEMPSTR2	TEMPSTR2	Temporary storage	45
AURB	URB (Multi-User only)	URB	45

[§§] These fields are used by SYSTEM 2000 and are altered regardless of action code.

EXIT00

EXIT00 occurs during single-user or Multi-User initialization. EXIT00 establishes user exit availability by setting bits in AVAILEXT and storing the exit address in the user exit load list.

The CSECT and load module name for EXIT00 must be S2KEXIN. This is the only exit that has a required name. S2KEXIN is loaded if it was not link-edited with S2EXIT. S2KEXIN establishes the available exit mask and the user exit load list to enable the exits that you want during this execution of SYSTEM 2000. 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.

S2KEXIN must be coded if the execution parameter EXITS=YES is specified, or if the S2EXIT interface routine was link-edited with SYSTEM 2000, otherwise an IBM OS/390 operation exception occurs. S2KEXIN must set up the available exit mask and user exit load list that is addressed by AVALEXIT and AUSRLOAD in the exit parameter list to enable the necessary exits and to provide the entry point addresses of the routines to be called for any enabled exits.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AVALEXIT	available exit mask
AUSRLOAD	user exit load list

The allowable action codes are: 0 and 4.

EXIT01

EXIT01 allows you to enable and disable exits during SYSTEM 2000 execution by using SCF and PLEX ENABLE/DISABLE commands. EXIT01 changes AVAILEXT and LOADLIST. If EXIT01 does change AVAILEXT and LOADLIST, the action code must be set to 4 or 12. If the command is ENABLE, ENDIS=1. If the command is DISABLE, ENDIS=0. Parameters passed to the exit provide information needed to make the change.

The label BITS is a field, two words in length, that identifies which exits were specified. Use of this area is similar to that of AVAILEXT. Counting left-to-right, the first bit position is 0, the second position is 1, and so on. Corresponding bits are on for exits specified. For example, bit 7 is on for EXIT07.

The field NMS is an 80-byte area for the specified exit names. Names are specified in the command. For SCF, each name is 8 characters in length, left-aligned, and blank filled. A maximum of 10 names can be specified. For PLEX, the 80-byte area is a copy of the 80-byte area that is passed by the PLEX program.

If the command is DISABLE, EXIT01 must update AVAILEXT by setting appropriate bits to 0 and storing 0's in the appropriate field of LOADLIST. LOADLIST can be correctly indexed by multiplying the exit number by 4 (for example, load exit number and shift left logical 2).

If the command is ENABLE, appropriate AVAILEXT bits must be set to 1 and the address of the enabled exit stored in LOADLIST.

The SETEXIT macro can be used with EXIT01.

The parameters available for this exit are

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

The allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 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 S2KUSERS file at this time.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage

Allowable action codes are: 0 and 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.

EXIT03 can set a PLEX return code.

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

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit WAIT routine
APASSWRD	password
ADBNAME	database name
ACCTINFOR	ACCT information (Multi-User only)
ACOMBLK	COMMBLOCK (PLEX only)
AS2KDUM	S2KDUM (PLEX only)
AERMSGLN	user message length
AERMSGBF	user message buffer

Allowable action codes are: 0, 8, 40, 44, 48 and 52.

EXIT04

EXIT04 can be used to de-allocate database files dynamically (thereby freeing resources not needed for an entire job). EXIT04 occurs after SYSTEM 2000 physically closes the database files.

EXIT04 can set a PLEX return code.

Note: You can also dynamically de-allocate database files by using the FREE command in SCF sessions. For details, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit WAIT routine
APASSWORD	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

Allowable action codes are: 0, 8, 40, 44, 48 and 52.

EXIT05

EXIT05 allows you to develop and use fixed encoding decryption routines or to add third-party 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. The page retrieved should be decrypted in place. If decrypted in a work buffer provided by you, the page must be moved into the buffer whose address is provided. Terminating a user at this time can result in a damaged database

A physical retrieval is an I/O to the database files; it is not a request for a page that resides in memory. Providing this user exit after physical I/Os for database files allows the user exit routine to decrypt database pages that exist on disk in an encrypted format. Changes should be accompanied by action code 4 or 12.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

Note: SYSTEM 2000 does not contain any decryption routines, nor was it designed to allow a user to call on any third party decryption routines. SYSTEM 2000 must be modified by the user, as described in EXIT05 above, use with decryption routines. If you add decryption routines and you plan to use the software outside the United States and Canada, you must obtain special export authorization from the United States Bureau of Export Administration.

EXIT06

EXIT06 allows you to use fixed encoding encryption routines or to add third-party 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 fixed encoding or third-party 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 BUFFADDR of DUSRPRM. Changes must be accompanied by action code 4 or 12 prior to returning control of S2EXIT. If the page needs no encryption, set the action code to 0. Terminating a user at this time can result in a damaged database.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit WAIT routine
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

Note: SYSTEM 2000 does not contain any decryption routines, nor was it designed to allow a user to call on any third-party decryption routines. SYSTEM 2000 must be modified by the user, as described in EXIT06 above, to be used with decryption routines. If you add decryption routines and you plan to use the software outside the United States and Canada, you must obtain special export authorization from the United States Bureau of Export Administration.

EXIT07

EXIT07 occurs after the physical opening of a database. You can change the table that specifies maximum allowable buffers per database file. Reducing the number of buffers can increase I/O's and increasing the number of buffers can decrease I/O's. Changes must be accompanied by action code 4 or 12.

You can specify a buffer size in AMHBUFSZ for the Multiple Local Holds buffer. The buffer is allocated by using a GETMAIN, rather than from a pool. If a size is not specified, the default is 23464.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 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 replaces the current unit and must be blank-filled to a fullword boundary. Changes must be accompanied by action code 4 or 12.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	password
ADBNAME	database name
ACCTINFO	ACCT information (Multi-User only)
AERMSGLN	user message length
AERMSGBF	user message buffer
AIBUFSIZ	syntactic unit size
AIBUF	current syntactic unit

Allowable action codes are: 0, 4, 8, 12, 24, 36, 40, 44, 48 and 52.

EXIT09

EXIT09 occurs before the physical opening of a Keepfile for an APPLY or a KEEP command. You can use this user exit to allocate the Keepfile dynamically, or you can let SYSTEM 2000 dynamically allocate the Keepfile by not allocating it in the JCL or in a CLIST. For more details about the Keepfile, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

If the Keepfile is a tape, TAPEOP in DUSRPRM 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. Dynamic allocation does not wait for a resource to be made available unless the program that makes 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 OS/390 Systems Programming Library, Job Management manual.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

EXIT10

EXIT10 occurs before a Savefile is physically opened for saving or restoring a database. The DDname for the Savefile must be the first 7 characters of the database name and the suffix S. The Savefile is a QSAM file. You can allocate the Savefile dynamically. You can also let SYSTEM 2000 dynamically allocate the Savefile by not allocating it in the JCL or a CLIST. For more details about the Savefile, see *SYSTEM 2000 CONTROL Language, Version 12.*

If the Savefile is a tape, TAPEOP in DUSRPRM 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. Dynamic allocation does not wait for a resource to be made available unless the program that makes 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

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

EXIT11

EXIT11 occurs after the physical closing of a Keepfile that follows an APPLY command or a KEEP command. You can de-allocate the Keepfile dynamically.

If SYSTEM 2000 dynamically de-allocated the Keepfile (not de-allocating it through JCL or in EXIT09), it de-allocates the Keepfile dynamically after the APPLY or KEEP command executes.

If the Keepfile is a tape, TAPEOP in DUSRPRM 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. Dynamic allocation does not wait for a resource to be made available unless the program that makes 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

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

EXIT12

EXIT12 occurs after the physical close of a Savefile, which occurs after a database has been saved or restored. You can dynamically de-allocate the Savefile.

If SYSTEM 2000 dynamically de-allocated the Savefile (not de-allocating it through JCL or in EXIT10), it de-allocates the Savefile dynamically after saving or restoring the database.

If the Savefile is a tape, TAPEOP in DUSRPRM 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. Dynamic allocation does not wait for a resource to be made available unless the program that makes 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

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

EXIT13

EXIT13 occurs after SYSTEM 2000 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 specific 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 pre-process an SCF input line before SYSTEM 2000 scans it. Records coming to EXIT13 from the user must have binary 0's 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 BUFFADDR of DUSRPRM. If necessary, update the input buffer size. Modifications must be accompanied by action code 4 or 12.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are as follows: 0, 4, 8, 12, 16, 28, 40, 44, 48, 52

EXIT14

EXIT14 occurs just before SYSTEM 2000 writes an SCF output line to the Message File or the 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 re-format output.

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 BUFFADDR in DUSRPRM. Residual information can 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. Modifications must be accompanied by action code 4 or 12. Terminating a user at this time can cause a damaged database.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are as follows: 0, 4, 8, 12, 20, 32, 40, 44, 48, 52

EXIT15

EXIT15 occurs after an SCF input value is parsed in the input line. These values occur in update action-clauses, in where-clauses, and in loader streams in the Data File. This user exit can be especially useful for editing input of values that must fall within specific ranges, for example, integers between a range of 5 to 9 or for allowing only specific values as input in a name. You can also create 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 currently defines. The item value can be modified only in place. Also, if the character count is changed, the item length must be updated. The value and its length must be acceptable for the database definition. Modifications must be accompanied by action code 4 or 12.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 24, 36, 40, 44, 48 and 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 create a table look-up of encoded values so that the resulting report contains readable data.

The item 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 (pointed to by AIBUFSIZ) must be set to the correct character count or else the pre-set field size is used. Modifications must be accompanied by an action code 4 or 12.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	password
ADBNAME	database name
ACCTINFO	ACCT information (Multi-User only)
AERMSGLN	user message length
AERMSGBF	user message buffer
ACOMPLN	component length
ACOMPKY	component key indicator
ACOMPFLT	component decimal specification
ACOMPVAL	component value
ACOMPNO	component number

Allowable action codes are: 0, 4, 8, 12, 20, 32, 40, 44, 48 and 52.

EXIT17

EXIT17 occurs before the processing of every PLEX command, retrieval, and update. It can also occur when there is no subschema record, for example, the CLEAR command. EXIT17 allows you to edit subschema record values to enforce a range or a security constraint. 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 must 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 that is set at the time of this exit can be overridden by using EXIT18 to set a new return code. A non-zero return code prevents any further processing of 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

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are : 0, 4, 8, 12, 24, 36, 40, 44, 48 and 52.

EXIT18

EXIT18 occurs prior to the return to a PLEX program. The subschema record that is 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 by the PLEX program. For example, COBOL can redefine the subschema record into fields that are different from those portrayed for SYSTEM 2000. Changes should be accompanied by action code 4 or 12. Putting the requested return code in PLIRTN in DUSRPRM can set a PLEX return code. Terminating a user at this time can result in a damaged database.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

EXIT19

EXIT19 occurs after an item value in a PLEX subschema record has been identified. Use EXIT19 for value verification, range editing, and security processing. The value is in a subschema record for an INSERT command , a MODIFY command, or a where-clause. The item value must be modified in place, honoring the subschema record picture size and type. Changes must 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. Putting the requested return code in PLIRTN in DUSRPRM can set a PLEX return code. 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

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

EXIT20

EXIT20 occurs after a retrieved item value is placed in the PLEX subschema record. The information provided at EXIT 20 allows 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 re-define a value. Changes must be accompanied by action code 4 or12. Putting the requested return code in PLIRTN in DUSRPRM can set a PLEX return code. Terminating a user at this time can result in a damaged database.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 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

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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

EXIT22

EXIT22 occurs after an end-of-file (EOF) in the SCF Command File for a batch SCF job or after an SCF TP end-of segment (EOS) is 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 that contains 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 by 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

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	password
ADBNAME	database name
AERMSGLN	user message length
AERMSGBF	user message buffer
AIBUFSIZ	input buffer size
AIBUF	input buffer

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

EXIT23

EXIT23 occurs after EXIT03 and before EXIT07. Use EXIT23 to inspect or modify the ACB of each database file or for security checking. EXIT23 is called once for each ACB of a database that is to be physically opened.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	password
ADBNAME	database name
AERMSGLN	user message length
AERMSGBF	user message buffer
ADCB	ACB
AFILENO	database file number

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

EXIT24

EXIT24 occurs before EXIT04. Use EXIT24 to inspect or modify the ACB of each database file. EXIT24 is called once for each ACB of a database that is to be physically closed.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	password
ADBNAME	database name
AERMSGLN	user message length
AERMSGBF	user message buffer
ADCB	ACB
AFILENO	database file number

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

EXIT25

EXIT25 occurs before 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 by setting an action code of 16 or 28, which 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 for an OPEN command as well as 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

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
APASSWORD	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

Allowable action codes are: 0, 4, 8, 12, 16, 28, 40, 44, 48 and 52.

EXIT26 through EXIT35

Reserved for future use.

EXIT36

EXIT36 occurs before a physical close of database files. This exit is for a Multi-User environment only. You can use EXIT36 to prevent Multi-User from executing a physical close of the database files by setting an action code of 16 or 28. If the close is in response to a FREE or a VARY OFFLINE command, EXIT36 is not taken.

Note: The password may belong to a database that is being opened rather than to the database that is being closed.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	password
ADBNAME	database name
ACCTINFO	ACCT information (Multi-User only)
ACOMBLK	COMMBLOCK (PLEX only)
AERMSGLN	user message length
AERMSGBF	user message buffer

Allowable action codes are: 0, 4, 8, 12, 16, and 28.

EXIT37

EXIT37 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, if enabled, is executed 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

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
APASSWORD	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

Allowable action codes are: 0, 4, 8 and 12.

EXIT38

EXIT38 occurs when a security violation is detected by SYSTEM 2000. Use EXIT38 to record security violations that the software detects. Fields such as, password, database name, job name, and terminal ID are provided for logging purposes. Here is a list of 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 authority required to access the item referenced in a PLEX statement.
- A secondary password holder issues an SCF command that contains an item the password does not have authority to use.
- A secondary password holder issues a command reserved for the DBA password holder or the master password holder. To see a list of commands that can be used by the DBA password, see *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*
- A secondary password holder has violated 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.*

- Access denied by EXIT45.

Other security violations occur when a user attempts to add or to change a password that already exists.

S2EXIT sets APTYPE, which is one of the EXIT38 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. The second halfword of the APTYPE value contains the security violation number in binary format.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWORD	password
ADBNAME	database name
ACCTINFO	ACCT information (Multi-User only)
ACOMBLK	COMMBLOCK (PLEX only)
ASCHEMA	subschema record control block (PLEX only)
AS2KDUM	S2KDUM (PLEX only)
APTYPE	processing type (used to indicate violation type)

The allowable action code is 0.

EXIT39

EXIT39 occurs prior to SYSTEM 2000 reading the Command File. EXIT39 enables you to specify an input buffer of commands and 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

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
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

Allowable action codes are: 0, 4, 8, 12, 16, 28, 40 and 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

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
APASSWRD	password
ADBNAME	database name
ACOMBLK	COMMBLOCK control block (PLEX only)
AS2KDUM	S2KDUM control block (PLEX only)
AERMSGLN	user message length
AERMSGBF	user message buffer

Allowable action codes are: 0, 8 and 44.

EXIT42

EXIT42 occurs at SYSTEM 2000 termination time. It is the last exit called. Use EXIT42 for clean-up tasks, such as freeing any storage obtained with GETMAINs, deleting modules loaded, closing files used by your user exit code, and so on. Regardless of whether you use EXIT42, SYSTEM 2000 frees memory obtained for user exits and deletes S2EXIT if it was loaded at initialization time.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AVALEXIT	available exit mask
AUSRLOAD	user exit load list

Allowable action codes are: 0 and 4.

EXIT43

EXIT43 allows you to examine operator console requests to Multi-User software. This exit occurs before Multi-User processing of the command. SYSTEM 2000 gives EXIT43 the command buffer that Multi-User will process. Any modification of the command in this buffer is passed to Multi-User.

EXIT43 can determine whether the command was issued by the master console or by using an S2OP call. If S2OP issued the command, the EMODE field in DUSRPRM is equal to 1. If the master console issued the command, the EMODE field is equal to 0 and EXIT43 has the following constraints:

- EXIT43 is not available until the first user signs on to Multi-User software.
- SYSTEM 2000 does not print any message that is contained in AERMSGBF, which is the user message buffer. You must specify your own WTO if you want to send a message to the console.

The parameters available for this exit are

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

Allowable action codes are: 0, 4, 8, 12, 16 and 28. If EXIT43 returns an invalid action code, SYSTEM 2000 sets the action code to 0, and processing continues.

EXIT44

This exit is general in that it allows you to intercept all update commands (SCF and PLEX). You define the purpose of this exit. Use it for monitoring updates, preventing updates, or collecting statistics on updates.

EXIT44 is called at the beginning of PLEX processing of a command and at the beginning of parsing of an SCF command. It has not yet been determined what components are involved in the update but some preliminary decisions have been made in both SCF and PLEX. The PLEX user has access to all PLEX control blocks passed by the program that initiates the update, and the SCF user has access to the information that identifies the user.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine address
APASSWRD	database password
ADBNAME	database name
ACCTINFO	accounting information
ACOMBLK	PLEX COMMBLOCK address
ASCHEMA	PLEX schema address
AS2KDUM	PLEX S2KDUM control block address
ABUFSIZ	database page buffer size
ADBBUF	current database page buffer address
AERMSGLN	user message length
AERMSGBF	user message buffer
AIBUFSIZ	size of current input buffer or syntax
AIBUF	address of current input buffer or syntax
ADBIO	database I/O count
ATIM	elapsed CPU time
ASEGIO	segment database I/O count

Allowable action codes are: 0, 16 and 28. In response to action code 16 or 28, a -504- message is written to the message file.

EXIT45

EXIT45 allows you to control access to each database based on USERID or other criteria you choose. You can restrict access type that is READ-only, UPDATE, or NO ACCESS, to something less than that allowed by the database password, by setting an action code that directs SYSTEM 2000 to impose the restrictions. You cannot expand access type beyond password authorities.

TEMPSTR2 is 400 bytes of storage in DUSRPRM. When EXIT45 is first called on behalf of a user, TEMPSTR2 is set to binary 0's. You can put whatever you want in TEMPSTR2, and set an action code to save it in the URA. The URA is a data area within SYSTEM 2000 that is unique for each user. When EXIT45 is invoked again on behalf of that user (switch in databases), TEMPSTR2 data saved in the URA is copied back to TEMPSTR2.

Action Codes for EXIT45 are: 0, 4, 8, 12, 36 and 40.

0 (update) No restrictions other than those established by the database password. Do not save TEMPSTR2 in the URA.

4 (update) No restrictions other than those established by the database password. Save TEMPSTR2 in the URA.

8 (read) Updates are not allowed, regardless of password authorities. There are no restrictions on READ access, other than those established by the database password. Do not save TEMPSTR2 in the URA.

You can issue a message to the user. If USRLEN (message length) is 0, a message is not issued.

12 (read) Updates are not allowed, regardless of password authorities. There are no restrictions on READ access other than those established by the database password. Save TEMPSTR2 in the URA.

You can issue a message to the user. If USRLEN (message length) is 0, a message is not issued.

36 (no access) User is not allowed access to the database. For PLEX, SYSTEM 2000 issues return code 14. For both PLEX and SCF, SYSTEM 2000 issues message 513 and calls EXIT38 if it is enabled. The job is allowed to continue. Message 513 states "An exit at your installation does not allow you to access the database."

You can issue a message to the user. If USRLEN (message length) is 0, a message is not issued.

40 (no access) User is not allowed access to the database. Terminate with SYSER 69.

Description of Example EXIT45

In the example provided, RACF USERID is retrieved for each job that executes SYSTEM 2000 code. The USERID is made available to EXIT45 where all RACF GROUPIDs to which the user is connected are retrieved and saved for the duration of the user's job. A maximum of 49 GROUPIDs can be saved.

Single-user jobs get the ACEE address from the address space extension block (ASXB). Multi-User gets the user's ACEE address using RACROUTE REQUEST=VERIFY. The USERID for Multi-User is put in URBRH1 by a dependent region exit (example EXTPLI). When a user first opens a database, EXIT45 is invoked and is passed the database name and, if Multi-User, the RACF USERID. The RACF USERID is in URBRH1, mapped with DURB. The address of the URB is in AURB in DUSRPRM.

The exit retrieves GROUPIDs and saves them in TEMPSTR2. The IDs are then scanned to determine the type of access allowed and the appropriate action code. On the first call by the user, the first word of TEMPSTR2 is 0. The number of GROUPIDs for the user and a maximum of 49 GROUPIDs are moved to TEMPSTR2 from the ACEE. The action code directs S2EXIT to save TEMPSTR2 only on the first call.

The action code applies to the master password and any secondary password. In the case of READ-only for the master password, the user can retrieve any item from the database. However, UPDATE or any activity that requires the master password is not allowed.

For Multi-User, the RACF USERID is obtained in accounting exits by getting the address of the ASCB from the PSA, the address of the ASXB from the ASCB, and the address of the ACEE from the ASXB. The ACEE contains the RACF USERID and all connected GROUPIDs. For single-user, the address of the ACEE is obtained in EXIT45. Single-user and dependent region jobs do not require any special authorization to gain access to the ACEE. In contrast, Multi-User EXIT45 obtains GROUPIDs for the user by using the RACROUTE REQUEST=VERIFY macro and, therefore, must run as an APF authorized program. APF authorization is set when EXIT45 is called in Multi-User mode; it is disabled when EXIT45 returns. APF authorization is not set in single-user mode or when using the SVC version of Multi-User.

The parameters available for this exit are

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	accounting information
ACOMBLK	COMMBLOCK (PLEX only)
ASCHEMA	schema record, PLEX only
AS2KDUM	S2KDUM (PLEX only)
AERMSGLN	user message length
AERMSGBF	user message 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
ATMPSTR2	address TEMPSTR2
AURB	address URB

Allowable action codes are: 0, 4, 8, 12, 36 and 40.

EXIT46 through EXIT49

Reserved for future use.

Dependent Region Exits

There are several exits for dependent region code. Some are for SCF; some are for PLEX; and others are for both SCF and PLEX. Additionally, some exits are only for Multi-User, while others are for both single-user and Multi-User. The term dependent region usually refers to a region that executes Multi-User interface code. Examples are SYS2KJOB and SYS2KTPI. S2KEX0 through S2KEX2 applies to both single-user and Multi-User. The term dependent region in this context refers to either interface.

Most of the exits have a specific load module and CSECT name. Exceptions are EXIT50 through EXIT53, which can be any name acceptable to the operating environment. EXIT50 through EXIT53 are enabled by S2KEXIN (EXIT00) and S2EXIT. All others are taken if they are linked with the SCF or PLEX interface module; otherwise, they are ignored.

EXIT50-EXIT53

EXIT50, EXIT51, EXIT52, and EXIT53 are for Multi-User SCF only. EXIT54 through EXIT60 are reserved for dependent region usage, but they are not yet implemented. EXIT50 through EXIT53 are enabled by using EXIT00 (S2KEXIN) and the exit interface S2EXIT, both are described in exit descriptions for mainline code. In fact, you can use the same copy of S2KEXIN and S2EXIT in both regions. References to exits that are not applicable to the region are ignored. You can use the execution parameter EXITS=YES or NO to enable or disable exit processing. Additionally, you can link S2EXIT with SYS2KJOB, SYS2KTPI, and S2KDMV6 to ensure any usage of Multi-User SCF will use exits. The execution parameter EXITS is ignored. Note that dependent region exits cannot be disabled by using EXIT01. There is no EXIT01 in the dependent region.

Descriptions of EXIT50 through EXIT53 are at the end of this section.

S2KEX0, S2KEX1, S2KEX2

Exits S2KEX0, S2KEX1, and S2KEX2 are PLEX exits for both single-user and Multi-User. The exits are linked by using the PLEX interface routine S2KPLR. The three CSECTS already exist. Therefore, to implement the exits, you use the linkage editor to replace existing CSECTS with your code.

S2KEX0 is used only to identify the amount of working storage needed by S2KEX1 and S2KEX2. It is a one-word CSECT that has the byte count of working storage that is needed. Interface module S2KPLR does a GETMAIN for the storage and stores the address in S2KSTR, a word within S2KDUM. The address of S2KDUM is passed as a parameter to S2KEX1 and S2KEX2.

S2KEX1 is invoked before the interface transfers control to SYSTEM 2000 processing code. When called, Register 1 points to S2KDUM; Register 10 points to the COMMBLOCK; and Register 11 points to the schema.

S2KEX2 is invoked when control is returned to the PLEX interface from SYSTEM 2000 processing code. When called, Register 1 points to S2KDUM; Register 10 points to the COMMBLOCK; and Register 11 points to the schema.

Accounting EXITS

Accounting exits allow you to alter job, step, and program names, which Multi-User uses to identify the job and reflects the changes in the user-termination accounting record. You also can alter job priority (URBPRTY), which is used by Multi-User for job dispatches. See PQA parameter in Chapter 14, “Execution Parameters” for details about priority dispatching. See Chapter 9, “Exits Used for Changing Fields in Accounting Log Records” for details about Accounting exits.

EXIT50

EXIT50 occurs when SYSTEM 2000 requires a new input line, just before reading the SCF Command File. EXIT50 either routes SYSTEM 2000 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, EXIT50 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 SCF input that you want. Also, its address must be set in the user work buffer-address field of the control block, even if the usual AIBUF input buffer address is used. If necessary, the input buffer size can be modified. Use action code 4 or 12.

If no input line from another source is required, EXIT50 indicates that SYSTEM 2000 should read the Command File. Indicate "no action" by setting an action code of 0 or 8.

The parameters available for this exit are

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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

EXIT51

EXIT51 occurs after SYSTEM 2000 reads 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 increased 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

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

Allowable action codes are: 0, 4, 8, 12, 16, 28, 40, 44, 48 and 52.

EXIT52

EXIT52 occurs before SYSTEM 2000 writes 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 can be included in the output unless the entire buffer (up to its LRECL size) is modified or cleared. Modifications must be accompanied by action code 4 or 12. If you change the output buffer size to exceed the LRECL size, the excess is not used.

The parameters available for this exit are

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

Allowable action codes are: 0, 4, 8, 12, 20, 32, 40, 44, 48 and 52.

EXIT53

EXIT53 occurs after SYSTEM 2000 detects an end-of-file condition in the SCF Command File. When this user exit occurs, end-of-job processing can be executed 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

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

Allowable action codes are: 0, 4, 8, 12, 40, 44, 48 and 52.

Chapter 9: Accounting

Obtaining Resource Statistics: Accounting Log

You can obtain resource usage statistics by instructing SYSTEM 2000 to write accounting information in the Accounting Log. The Accounting Log includes statistics about CPU time and I/O counts used by each SYSTEM 2000 session and each Multi-User segment. You can store the Accounting Log file on tape or disk. When the file resides on disk, you may use the ACTUTIL program to dump the file contents to a sequential file in secondary storage. This chapter provides the information you need to use the Accounting Log.

Records in the Accounting Log File

The Accounting Log file can contain six types of records. Each type of record is described in detail later in "Accounting Log Record Layouts for Version 1."

Table 9.1 Types of Records in Accounting Log File

Types of Records	Description
Multi-User Initialization	written whenever Multi-User is initialized. contain the Multi-User parameters selected, the time and date of Multi-User initialization, and the Multi-User region size.
User-Termination	written when a SYSTEM 2000 user session terminates. contain the following: date and time of start and termination of a user job user's Job Name, Step Name, and Program Name terminal ID (if TP user) SYSTEM 2000 release number CPU time used Multi-User region size, user priority, number of I/Os to Database files number of I/Os to Scratch files, total I/Os for Database files, Scratch files, Sort files, and user files.

Table 9.1 Types of Records in Accounting Log File *(continued)*

Types of Records	Description
Multi-User Termination	contain the time and date when Multi-User terminates.
Multi-User Segment	written whenever you specify CPU-time or I/O count thresholds and these limits are met or exceeded. You can request information about PLEX segments, batch SCF segments, and SCF TP segments.
	contain the same information as a User-Termination record but the data pertains to a specific Multi-User segment.
Header	contain a date-and-time stamp that indicates when ACTUTIL started dumping the contents of a data set to a sequential file in secondary storage.
Trailer	contain a date-and-time stamp that indicates when ACTUTIL finished dumping the contents of a data set to a sequential file in secondary storage.

Using the Accounting Log

The Accounting Log can be implemented on tape or disk. On tape, you need to specify only the execution parameters that determine which accounting statistics are compiled. Multi-User writes the records that contain 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 requirement for a new tape. However, tape implementations 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 recommended practice. The discussion later in this chapter about execution parameters in "Generating Accounting Data: SYSTEM 2000 Execution Parameters" refers to both implementations.

Note: If you decide to use the Accounting Log, you must add the necessary DD statements to the JCL that executes Multi-User.

If the Accounting Log is implemented on disk, add

```
//S2KMANX   DD   DSN=dsnacl-X,DISP=SHR
//S2KMANY   DD   DSN=dsnacl-Y,DISP=SHR
```

If the Accounting Log is implemented on tape, add

```
//S2KMANX   DD   DSN=dsnacl,DISP=(NEW,CATLG),
//              DCB=BLKSIZE=blksize,UNIT=TAPE
```

In both instances, *dsnacl* is the data set name for the Accounting Log data set described later in this chapter in "Initializing the Accounting Log Data Set(s)".

To calculate the block size, use this formula:

BLKSIZE = (252 × blocking factor) + 4

That is, multiply the LRECL of the largest record by the blocking factor and add 4 bytes for the block descriptor word.

Initializing the Accounting Log Data Set(s)

The only initialization necessary for the Accounting Log data set(s) is to decide what block size you are going to use for the variable blocked data set. Before specifying the block size to use, consider these factors:

- A small block size means more I/Os to the disk but fewer records are lost if the system fails.
- A large block size means fewer I/Os to the disk but more records are lost if the system fails.
- The types of Accounting Log records that will be written to the disk. The most common records are the User-Termination records that have an LRECL of 104 bytes or the Multi-User segment records that have an LRECL to a maximum of 252 bytes.
- The track size of the disk. You want to minimize wasted space on the disk.

Sample JCL for allocating disk data sets:

```
//JCLACT JOB (ACCTINFO),
//       S2K,REGION=0M
//*
//*************************************************************
//* ALLOCATE DATA SETS FOR ACCOUNTING LOG FILES
//*************************************************************
//*
//ACTBLK   EXEC PGM=IEFBR14
//MANX     DD   DSN=S2K.V1.ACCOUNT.MANX,
//              DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//              VOL=SER=DISK01,
//    DCB=(RECFM=VB,LRECL=252,BLKSIZE=9076,DSORG=PS),
//              SPACE=(CYL,(10))
//MANY     DD   DSN=S2K.V1.ACCOUNT.MANY,
//              DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//              VOL=SER=DISK01,
//    DCB=(RECFM=VB,LRECL=252,BLKSIZE=9076,DSORG=PS),
//              SPACE=(CYL,(10))
//*
```

Sample JCL for a tape file:

```
//S2KMANX  DD DSN=S2K.V1.ACCOUNT.MANX,DISP=(,KEEP),UNIT=CART,
//            DCB=(RECFM=VB,DSORG=PS,LRECL=252,BLKSIZE=2524),
//            VOL=SER=123456,LABEL=(,SL)
```

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 to 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.

The corresponding messages for S2KMANY are S2K1304 and S2K1307.

When using Accounting Logs on disk, Multi-User automatically switches files when one file fills up. If, in the process of switching files, it is determined that both are full, the accounting process is suspended and the following WTO's are sent to the operator:

S2K1311/*sid*- NO AVAILABLE ACCOUNTING LOG FILES –
S2K1314/*sid*- ACCOUNTING SUSPENDED, ALL FILES FULL

After these WTO's are sent, the operator is required to enter the following Multi-User operator console command to re-start accounting:

START ACCOUNTING

Upon receiving this operator command, Multi-User turns off the suspend flag and allows accounting to proceed.

After an Accounting Log has been identified as being full, the ACTUTIL utility must be run against that file to dump the data to secondary storage. If the data is not dumped, the file will not be marked as available for use. As long as both files are unavailable, accounting remains suspended. While accounting is suspended, a count of lost records is kept and the following WTO is sent for every 1024 records lost:

S2K1309/*sid- nnnnnn*K ACCNT RECORDS LOST DUE TO FULL ACCOUNTING LOG FILE- PLEASE DUMP

When Multi-User is shut down, the following WTO is issued to indicate the total number of records lost during this session:

S2K1313/*sid- nnnnnn* RECORDS LOST FOR ENTIRE SESSION

To prevent the loss of data, dump each data set as it becomes full. That is, dump the data set after WTO 1304 or 1305 is issued. If you attempt to dump a data set that Multi-User has open and is currently writing to (the active log), the following message is issued from the ACTUTIL utility:

DUMP OF ACTIVE ACCOUNTING LOG NOT ALLOWED

After you dump the data, you can execute a program to process the data. We have supplied the source code for a program named ARPMAIN that will process your Accounting Log. You may use this program as is, or you can modify it to fit your needs. The main program for this process is written in COBOL, and its use is described in the section "Using ARPMAIN to Process the Accounting Log**."**

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)
```

In this JCL,
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 × blocking factor) + 4

That is, multiply the LRECL of the largest record by the blocking factor and add 4 bytes for the block descriptor word.

The following sample JCL uses a blocking factor of 25:

```
//DUMPFILE  DD   DSN=S2K.V1.ACCOUNT.DUMP,DISP=(NEW,CATLG,DELETE),
//               UNIT=CART,
//               DCB=(LRECL=252,BLKSIZE=6304,RECFM=VB)
```

After your initial creation of the dump file, you can append additional accounting files using DISP=MOD. The following sample JCL is used to dump accounting files.

```
//JCLACDMP JOB (ACCTINFO),
//      S2K,REGION=0M
//*
//*************************************************************
//* DUMP ACCOUNTING LOG
//*************************************************************
//*
//ACTBLK   EXEC PGM=ACTUTIL,
//              PARM='FUNCTION=DUMP'
//STEPLIB  DD  DSN=S2K.V1.LOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=&SYSOUT
//ACCTFILE DD  DSN=S2K.V1.ACCOUNT.MANX,DISP=SHR
//*ACCTFILE DD  DSN=S2K.V1.ACCOUNT.MANY,DISP=SHR
//DUMPFILE DD  DSN=S2K.V1.ACCOUNT.DUMP,DISP=MOD,
//              UNIT=CART
//*
```

See *SYSTEM 2000 Messages and Codes, Version 1, First Edition* for information about the WTO and SYSPRINT messages that might be issued when executing ACTUTIL.

Generating Accounting Data: SYSTEM 2000 Execution Parameters

You can use the following four SYSTEM 2000 execution parameters to generate accounting data.

ACCT Parameter After you have defined the two disk data sets, you must set the ACCT execution parameter to YES so that Multi-User will start writing accounting data to the Accounting Log. The YES or NO status of the SYSTEM 2000 ACCT execution parameter cannot be changed from the console.

ACCT = YES|NO

YES generates accounting data. When ACCT=YES, the following Accounting Log records are generated: Multi-User Initialization, User-Termination, and Multi-User Termination. NO stops generating accounting data. The default is NO.

NLSEG, TPSEG, and PLSEG Parameters If you also require Multi-User segment accounting data, you must specify one or more of the following execution parameters: NLSEG, TPSEG, and PLSEG.

NLSEG obtains accounting data for batch SCF segments. TPSEG obtains accounting data for SCF TP segments. PLSEG obtains accounting data for PLEX Multi-User segments.

You can control how much accounting data is generated for Multi-User segments by specifying CPU-time and I/O-count thresholds as execution parameter options. The syntax of NLSEG, PLSEG, and TPSEG also allows you to delay, interrupt, or stop segment data from being generated until Multi-User receives a command from the console.

|NLSEG = *CPU-time/I/O-count*[/DELAY]
|PLSEG
|TPSEG

CPU-time is the threshold time in 100^{ths} of a second, specified as an integer in the 0 to 999999 range. Multi-User produces accounting data segments that meet the I/O criterion and whose CPU-time equals or exceeds the specified time.

I/O-count is the threshold number of database I/O's 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 criterion.

DELAY delays Multi-User segment-level accounting until a START request is received from the console. If DELAY is not specified, Multi-User segment-level accounting starts immediately.

You can also issue an operator console command to control segment-level accounting. The syntax for that command is

|MODIFY *jobname,* |START STAT |NL
|F |STOP STAT |PL
|TP
|ALL

jobname refers to the SYSTEM 2000 job name. START tells Multi-User to start writing segment-level data into the Accounting Log file. This applies if the writing was delayed because DELAY was specified in the execution parameter or stopped because of a previous STOP specification.

NL controls writing of the accounting data for batch SCF segments. PL controls writing of the accounting data for PLEX Multi-User segments. TP controls writing of the accounting data for SCF TP segments. ALL controls writing of the accounting data for all three types of Multi-User segments.

SYSTEM 2000 Execution Options

You can use the following four SYSTEM 2000 execution options.

OPT004

includes system-related time in Types 5, 6, and 8 records or segment-level accounting records.

Syntax

OPT004=YES| NO

Details

YES includes system-related (SRB) time with problem-related (PRB) time in Accounting Log Types 5, 6, and 8. NO means that only PRB time is specified in Accounting Log Types 5, 6, and 8 records. The default is NO.

OPT031

includes the number of user calls in theType 4 User-Termination record.

Syntax

OPT031=YES|NO

Details

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 means that the User-Termination record (Accounting Log Type 4) is not altered. The default is NO.

OPT037

places the synchpoint ID in the ACCTALT field of the user's URB.

Syntax

OPT037=YES|NO

Detail

YES places the synchpoint ID for a logical unit of work (LUW) in the ACCTALT field of the user's URB. NO means that the ACCTALT field of the user's URB is not affected. The default is NO.

OPT051

keeps the elapsed CPU time in segment records in 1000^{ths} of a second, instead of 100^{ths} of a second.

Syntax

OPT051=YES |NO

Details

YES means that the CPU times in the Accounting Log segment records are kept in 1000^{ths} of a second. NO means that the CPU times in the Accounting Log segment records are kept in 100^{ths} of a second. The default is NO.

Using ARPMAIN to Process the Accounting Log

The Accounting Log Report Program, ARPMAIN, provides better access to your Accounting Log data without your having to invest time in program development. The report program is flexible enough to allow you to generate reports that contain only the information you want or all of the information that is in the Accounting Log. ARPMAIN is parameter driven and allows you to print the data that contains a specific field value or a range of field values from the Accounting Log records. The report program accepts as its input file either your dumped Accounting Log or a qualified data set built by the AQUMAIN program, which is discussed later in this chapter in "Using AQUMAIN to Subset the Accounting Log."

The output generated by ARPMAIN is a sequential data set in SYSOUT format. This output is usually directed to a printer but automatically adjusts to an 80- or 132-byte record size. The parameters that control the execution of ARPMAIN are

JOBNAME= |*(xxxxxxxx,xxxxxxxx)*
|*(,xxxxxxxx)*
|*(xxxxxxxx)*
|*xxxxxxxx*
|*xxx**

STEPNAME= |*(xxxxxxxx,xxxxxxxx)*
|*(,xxxxxxxx)*
|*(xxxxxxxx)*
|*xxxxxxxx*
|*xxx**

PROGRAM= |*(xxxxxxxx,xxxxxxxx)*
|*(,xxxxxxxx)*
|*(xxxxxxxx)*
|*xxxxxxxx*
|*xxx**

TERMINAL= |*(xxxxxxxx,xxxxxxxx)*
|*(,xxxxxxxx)*
|*(xxxxxxxx)*
|*xxxxxxxx*
|*xxx**

DATE= |*(MM/DD/YYYY,MM/DD/YYYY)*
|*(,MM/DD/YYYY)*
|*(MM/DD/YYYY)*
|*MM/DD/YYYY*

TIME= |*(hh:mm:ss.cc,hh:mm:ss.cc)*
|*(,hh:mm:ss.cc)*
|*(hh:mm:ss.cc)*
|*hh:mm:ss.cc*

CPUTIME= |*(nnnnnnnn,nnnnnnnn)*
|*(,nnnnnnnn)*
|*(nnnnnnnn)*

DBIO= |*(nnnnnnnn,nnnnnnnn)*
|*(,nnnnnnnn)*
|*(nnnnnnnn)*

SCRIO= |*(nnnnnnnn,nnnnnnnn)*
|*(nnnnnnnn)*
|*(,nnnnnnnn)*

TOTALIO= |*(nnnnnnnn,nnnnnnnn)*
|*(nnnnnnnn)*
|*(,nnnnnnnn)*

SEGLEV= YES|NO|ONLY

TEXT= YES|NO

MUINFO= YES|NO

TYPE= *nn*

DATABASE= |*xxxxxxxx*
|*xxx**

The **JOBNAME, STEPNAME, PROGRAM**, and **TERMINAL** parameters can have several different input formats. When you specify a range of values, enclose the values in parenthesis. The values specified for a range cannot be generic. The values that you specify are the low value to be accepted and the high value to be accepted, in that order. This is true of all range values for all of the parameters that allow a range. You may specify a low value (*,xxxxxxxx*) only, or a high value (*xxxxxxxx*) only. In either case, the value must be enclosed in parenthesis to indicate that a range value is being specified. Generic values are not allowed for this type of specification.

If you want to qualify a single value only, omit the parenthesis and specify a generic, *xxx** or a full *xxxxxxxx* value. Qualified records must contain this value in the appropriate field to qualify for the report. If any of these four parameters (JOBNAME, STEPNAME, PROGRAM, and TERMINAL) is omitted, the corresponding field in the accounting records are not considered in the qualifying process. Values for the various parameters can be numeric or alphanumeric, or they can contain special characters except the asterisk (*). The asterisk (*) is used to denote a generic value and is not valid as part of a data value.

The **DATE, TIME, CPUTIME**, **DBIO, SCRIO,** and **TOTALIO** parameters limit the report to records that contain values that are within the specified range in these fields. If only the first value in the range is specified for any of these parameters, it is considered to be the low value. All records in the Accounting Log that contain values greater than or equal to this low value will qualify for the report. If only the second value in the range is specified, it is considered the high value, and only records that have a value less than or equal to this value will qualify for the report.

The **SEGLEV** parameter specifies whether segment-level accounting records are included in this report. YES indicates that segment-level records are to be included in the qualification process. NO indicates that segment-level records are not to be included in the qualification process. ONLY indicates that only segment-level records are to be included in the qualification process.

The **TEXT** parameter specifies whether the text information from a segment-level record is displayed. YES specifies that an additional line of output is included with each segment-level record for the display of text information. NO specifies that no text information from the SCF and PLEX segment-level records is included in the report. If SEGLEV=NO, the TEXT parameter is ignored.

The **MUINFO** parameter specifies whether the information in the Multi-User Initialization record is displayed. YES indicates that information from the Multi-User initialization should be displayed in the report. NO indicates that no Multi-User initialization information should be displayed in the report.

The **TYPE** parameter allows you to select a specific type of record for display. The valid types are: 04, the user-termination record; 05, the PLEX segment-level record; 06, the SCF TP segment-level record; and 08, the batch SCF segment-level record.

The **DATABASE** parameter can be used with the SEGLEV parameter to qualify only PLEX segment-level accounting records in which the database name matches this DATABASE parameter value. If SEGLEV=NO, the DATABASE parameter is ignored.

ARPMAIN displays all pertinent information from each record in the Accounting Log in a one- or two-line display. If segment-level text will be displayed, a third line is used. Multiple records are displayed on each page and a separate page is used for displaying Multi-User initialization information. The beginning page of the report has a section that displays the qualifying parameters for the report. Each page has a heading line to identify the individual fields. In the 132-byte format, the whole record content is displayed on one line. In the 80-byte format, the record is displayed on two lines of output. If TEXT=YES, the text portion of the segment-level records is displayed on a line by itself for the 132-byte format and, possibly, on two lines for the 80-byte format.

The driver program for the ARPMAIN process is ARPMAIN. You can use the source code for the ARPMAIN program as it is when distributed, or you can modify it to fit your specific needs. The ARPMAIN program is a COBOL program that is compatible with both COBOL II and COBOL for OS/390 and VM.

Sample JCL for Executing ARPMAIN

The following sample JCL executes the ARPMAIN program using a Multi-User Accounting Log as input. In this sample JCL, the qualification parameters are specified in-line by using the DD * statement.

```
//JCLARPRU JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//*********************************************************
//* EXECUTE ARPMAIN PROGRAM
//*********************************************************
//ARP      EXEC PGM=ARPMAIN
//STEPLIB  DD DSN=S2K.V1.LOAD,DISP=SHR
//ARPINPT  DD DSN=S2K.V1.ACCOUNT.MANY,DISP=SHR
//ARPPARM  DD *
MUINFO=NO
/*
//ARPOUTP  DD SYSOUT=A
//ARPRPRT  DD SYSOUT=A
//ARPRP80  DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSOUT   DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//*
```

The following definition for the ARPPARM data set can also be used:

//ARPPARM DD DSN=*user.pds*(*member*),DISP=SHR

member contains the same parameter values as those contained in the ARPMAIN DD * statement in the sample above. Also, in this sample,

- The ARPINPT data set points to a Multi-User Accounting Log. You could have pointed this DD statement to a file that contains the output from an AQUMAIN run.
- The ARPRPRT data set is written as 132-byte printer output.
- The ARPRP80 data set is written as 80-byte printer output.

Using AQUMAIN to Subset the Accounting Log

The Qualify Utility, AQUMAIN, accepts the same parameters that ARPMAIN accepts. The SEGLEV, MUINFO, and TEXT parameters are accepted by AQUMAIN but don't impact the qualification. The SORT parameter is accepted by AQUMAIN only. The SORT parameter directs AQUMAIN to sort the qualified data in specified order as the last step. The syntax for the SORT parameter is

SORT=((*field1*,*order*),(*field2*,*order*),…,(*fieldn*,*order*))

field can have one of the following values:

JOBN	(Job Name)
STEP	(Step Name)
PROG	(Program Name)
TERM	(Terminal Name)
DATE	(Start Date)
TIME	(Start Time)
CPUT	(CPU Time)
DBIO	(Database I/O)
SFIO	(Scratch File I/O)
TOTA	(Total I/O)
TYPE	(Record Type)

order can be A, for ascending; or D, for descending. The order in which the fields are specified dictates the priority of the fields in the sort process. In the following example, the values specified for SORT cause the qualified records to be sorted, first, by Job Name in ascending order, and second, by CPU Time in descending order.

SORT=((JOBN,A),(CPUT,D))

The AQUMAIN program qualifies the following record types only: 4, User-Termination records; 5, PLEX segment-level records; 6, SCF TP segment-level records; and 8, SCF Batch segment-level records.

The data set that results from running the AQUMAIN program can be used as input to the ARPMAIN program or as input to a sort process of your own design. You might want to use AQUMAIN to create a data set that is used as input by one of your own programs.

The driver program for the AQUMAIN process is AQUMAIN. You can use the source code for the AQUMAIN program as it is provided, or you can modify it to fit your specific needs. The AQUMAIN program is a COBOL program that is compatible with both COBOL II and COBOL for OS/390 and VM.

Sample JCL for Executing AQUMAIN

The following sample JCL executes the AQUMAIN program using a Multi-User Accounting Log as input.

```
//JCLAQURU JOB (ACCTINFO),
//     S2K,REGION=0M
//*
//*********************************************************
//* EXECUTE AQUMAIN PROGRAM
//*********************************************************
//AQU       EXEC PGM=AQUMAIN
//STEPLIB   DD DSN=S2K.V1.LOAD,DISP=SHR
//AQUINPT   DD DSN=S2K.V1.ACCOUNT.MANY,DISP=SHR
//AQUOUTP   DD DSN=S2K.V1.ACCOUNT.QUALIFY,DISP=SHR
//AQUPARM   DD *
MUINFO=NO
SORT=((JOBN,A),(TERM,A),(CPUT,D))
/*
//AQUREPT   DD SYSOUT=A
//AQURE80   DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SORTIN    DD DSN=S2K.V1.ACCOUNT.QUALIFY,DISP=SHR
//SORTOUT   DD DSN=S2K.V1.ACCOUNT.QUALIFY,DISP=SHR
//*
```

In this sample JCL, the AQUPARM data set is defined as a member in a user PDS. The parameters can be defined in-line to the JCL by using the AQUPAPM DD * statement. Also, in this sample,

- AQUINPT must point to a Multi-User Accounting Log.
- AQUOUTP is a variable blocked sequential data set in which LRECL=252 and BLKSIZE=(252 * *n*) + 4.
- AQUREPT is the 132-byte report data set that displays information about the execution of AQUMAIN.
- AQURE80 is the 80-byte report data set.

If you use the SORT parameter, you must specify the SORTIN and SORTOUT data sets for sorting purposes. Both SORTIN and SORTOUT (also, AQUOUTP) point to the same file because the sort is executed in place in the data set.

Accounting Log Record Layouts for Version 1

If you currently have programs that process the Accounting Log records, you might have to make changes to those programs to accommodate the new record formats. The changes made to the record layouts were necessary to allow sorting of these records by standard Sort programs.

Table 9.1 Multi-User Initialization Record

Offsets Dec	Hex	Field Name	Length	Type	Description
0	0	AR0LEN	2	BINARY	Record Length
2	2		2	BINARY	Reserved
4	4	AR0RID	2	BINARY	Record ID (0)
6	6	AR0SID	2	CHARACTER	System ID
8	8	AR0RNO	4	CHARACTER	Release Number (1.0)
12	C	AR0STDT	4	PACKED DECIMAL	Start Date in 0CYYDDD format
16	10	AT0STTM	4	BINARY	Start Time in 100^{ths} of seconds
20	14	AR0RGSZ	4	BINARY	Region Size
24	18	AR0#THR	2	BINARY	Number of Threads
26	1A	AR0SDBS	2	BINARY	Number of Small Databases
28	1C	AR0SDSZ	2	BINARY	Small Database Size
30	1E	AR0LDBS	2	BINARY	Number of Large Databases
32	20	AR0LDSZ	2	BINARY	Large Database Size
34	22	AR0SID	2	BINARY	System ID
36	24		4	BINARY	POOL0 . . . . .
40	28		4	BINARY	POOL1 :
44	2C		4	BINARY	POOL2 :
48	30		4	BINARY	POOL3 size/number
52	34		4	BINARY	POOL4 of buffers
56	38		4	BINARY	POOL5 :
60	3C		4	BINARY	POOL6 :
64	40		4	BINARY	POOL7 . . . . .
68	44		4	BINARY	COPYAREA1 . . . . .
72	48		4	BINARY	COPYAREA2 :
76	4C		4	BINARY	COPYAREA3 size/number
80	50		4	BINARY	COPYAREA4 of buffers
84	54		4	BINARY	COPYAREA5 :
88	58		4	BINARY	COPYAREA6 . . . . .
92	5C	AR0LCNT	2	BINARY	Log Count
94	5E	AR0USER	2	BINARY	Number of Users
96	60	AR0TPTH	2	BINARY	Number of TP Threads
98	62	AR0TPSC	2	BINARY	Number of TP Run Units
100	64	AR0ACCT	2	BINARY	Accounting Flag
102	66	AR0OPI	2	BINARY	Operator Interface Flag
104	68		2	BINARY	USER ACCOUNTING EXITS
106	6A		2	BINARY	ACCOUNTING FILE USAGE
108	6C	AR0LLEV	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

Table 9.1 Multi-User Initialization Record (*continued*)

Offsets Dec	Hex	Field Name	Length	Type	Description
110	6E		2	BINARY	STAE Error Trapping
112	70		2	BINARY	Diagnostic Snaps
114	72	AR0RWTR	2	BINARY	Report Writer Flag
116	74		4	BINARY	Reserved
120	78	AR0LIST	2	BINARY	List Parameter
122	7A		2	BINARY	Reserved
124	7C	AR0TSO	2	BINARY	TSO Flag
126	7E		2	BINARY	Reserved
128	80	AR0STTP	2	BINARY	Start TP Value
130	82	AR0FPTP	2	BINARY	Fetch Protected TP Flag
132	84		8	BINARY	Reserved
140	8C	AR0LHLD	2	BINARY	Local Hold Flag
142	8E		2	BINARY	Reserved
144	90	AR0PLCPU	4	BINARY	PLEX Segment CPU-Time
148	94	AR0PLIO	4	BINARY	PLEX Segment I/O Count
152	98	AR0TPCPU	4	BINARY	SCF TP Segment CPU-Time
156	9C	AR0TPIO	4	BINARY	SCF TP I/O Count
160	A0	AR0NLCPU	4	BINARY	Batch SCF CPU-Time
164	A4	AR0NLIO	4	BINARY	Batch SCF I/O Count
168	A8	AR0EXITS	2	BINARY	Exits Parameter
170	AA	AR0PQA	4	BINARY	Priority Queueing
174	AE	AR0CVRT	2	BINARY	Convert Parameter
176	B0	AR0CORC	2	BINARY	Coordinated Recovery Flag
178	B2	AR0PADS	64	BINARY	16 of 2-byte Number of AUs 2-byte Bytes per AU
242	F2		10		Reserved

Table 9.2 Multi-User Termination Record

Offsets Dec	Hex	Field Name	Length	Type	Description
0	0	AR12RID	2	BINARY	Record ID
2	2	AR12SID	2	CHARACTER	System ID
4	4	AR12REL	4	CHARACTER	Release Number
8	8	AR12SDT	4	PACKED DECIMAL	Stop Date
12	C	AR12STM	4	BINARY	Stop Time

The User-Termination and Segment records share a common layout for Version 1. The common format of accounting record Types 4, 5, 6, and 8 is shown in Table 9.3.

Table 9.3 User Records

Offsets Dec	Offsets Hex	Field Name	Length	Type	Description
0	0	ARRECLEN	2	BINARY	Record Length
2	2		2	BINARY	Reserved
4	4	ARRECID	2	BINARY	Record ID 4 for User-Termination 5 for PLEX segment 6 for SCF TP segment 8 for SCF batch segment
6	6	ARSYSID	2	CHARACTER	System ID
8	8	ARRELNO	4	CHARACTER	Release Number (1.0)
12	C	ARSTPDT	4	PACKED DECIMAL	Stop Date in 0CYYDDDF format
16	10	ATSTPTM	4	BINARY	Stop Time in 100^{ths} of seconds
20	14	ARJOBNM	8	CHARACTER	Job Name
28	1C	ARSTONM	8	CHARACTER	Step Name
36	24	ARPGMNM	8	CHARACTER	Program Name
44	2C	ARTRMNM	8	CHARACTER	Terminal Name
52	34	ARPRTY	2	BINARY	User Priority
54	36		14	CHARACTER	Reserved
68	44	ARSTRDT	4	PACKED DECIMAL	Start Date
72	48	ARSTRTN	4	BINARY	Start Time
76	4C	ARCPUTM	4	BINARY	CPU Time in 100^{ths} of seconds
80	50	ARDBIO	4	BINARY	Database File I/O Count
84	54	ARSFIO	4	BINARY	Scratch File I/O Count
88	58	ARTOTIO	4	BINARY	Total I/O Count
92	5C		4	BINARY	Reserved
96	60	ARREGSZ	4	BINARY	Region Size
100	64		4	BINARY	Reserved

The layout in Table 9.3 is the same for all of the mentioned record types (4, 5, 6, 8) and represents the first 104 bytes of all these record types. For User-Termination records (Type 4) this is the entire record. For Types 5, 6, and 8, the remainder of the record (a maximum of 252 bytes) is laid out as shown in Tables 9.4 and 9.5.

Table 9.4 Type 5 (PLEX Segment Level)

Offsets Dec	Hex	Field Name	Length	Type	Description
104	68	AR5OPCD	4	BINARY	Operation Code
108	6C	AR5DBNM	8	CHARACTER	Database Name
116	74	AR5SCHN	30	CHARACTER	Schema Name
146	92	AR5DWLN	4	BINARY	Dynamic Where Length
150	96	AR5DWCL	102	CHARACTER	Dynamic Where-Clause

Table 9.5 Type 6 and Type 8 Records (SCF TP and Batch)

Offsets Dec	Hex	Field Name	Length	Type	Description
104	68	AR6INLN	4	BINARY	Command length
108	6C	AR6INPT	144	CHARACTER	Input SCF command up

Note: In Accounting Log records prior to Version 1, the I/O count fields were packed decimal. In Version 1, the fields are now 4-byte binary values.

Note: SYSTEM 2000 no longer has a Lost Record record. Version 1 has a WTO that is displayed at the end of the Multi-User session to indicate the number of records lost during that session.

Using Exits to Change Fields in Accounting Log Records

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: The user abend code must not be one that is already in use. Abend codes currently in use are described in *SYSTEM 2000 Messages and Codes, Version 1, First Edition.*

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 jobname, stepname, program name, and user priority (URBPRTY). To modify values for these fields, 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 shown in Table 9.6.

Table 9.6 URB Fields

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 "Accounting Log Record Layouts for Version 1." Prior to modification, the jobname field contains the OS/390 jobname; the stepname field contains the OS/390 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 Chapter 14, "Execution Parameters**."**

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 priority in the Multi-User control blocks is set based on this value. Therefore, after the accounting exit is taken, the user priority cannot be changed. TP users can change priorities for each command segment, but they cannot change priorities during a specific command segment.

If the jobname field in the URB is modified, Multi-User recognizes a specific 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.

Several interfaces to Multi-User are provided by SAS Institute. Individual user exits can be linked into each interface. The interface protocol is identical for each user exit. An external reference to a specific 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 to the interface. If an exit is present, it is called one time by using standard OS/390 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 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. The initialization-time user exit for SCF is named SCFUEX4 for CICS. For PLEX programs, it is named PLXUEX1. Each user exit is expected to be a callable CICS command-level program. The operator-command facility is the same as SCF. User exits available in the SYSTEM 2000 interface to CICS are described in *SYSTEM 2000 Interface to CICS, Version 1, First Edition.*

The other four Multi-User interfaces are non-overlayed load modules (as members of the standard SYSTEM 2000 load library) that all customers receive. 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 (shown in Table 9.7). 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 Table 9.7. The interface routine GETS2K, which calls the user exit, resides in both SYS2KJOB and MUPLINT. S2KDMV6 is the SAS Version 6, SCF interface invoked by executing PROC QUEST.

Table 9.7 Interface Modules and User Exits

Executable Load Module Name	User Exit Name	CALL Routine in Interface	Entry Point
SYS2KJOB	EXJOB	GETS2K	MSYS2K
MUPLINT	EXPLI	GETS2K	MSYSTM2
SYS2KTPI	EXTSO	SCFTP2I	SCFTP2I
S2KDMV6	EXTSO	S2KDMV6	S2KDMV6

The following example shows linkage-editor JCL and control statements for including the appropriate user exit in SYS2KJOB. Copies of SYS2KJOB, MUPLINT, and SYS2KTPI must be saved before any re-linking is attempted, so that the functional version still exists if problems occur.

Sample Link-Edit JCL that Includes EXJOB in SYS2KJOB

```
//JCLEXACC JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//*************************************************************
//* RELINK DEPENDENT MODULES WITH ACCOUNTING USER EXITS
//*************************************************************
//*
//S2KLINK  PROC SYSOUT=A,WRKUNIT=SYSDA
//LKED     EXEC PGM=IEWL,
//SYSLMOD  DD  DSN=S2K.V1.LOAD,DISP=SHR
//LOAD     DD  DSN=S2K.USER.LOAD,DISP=SHR
//SYSLIN   DD  DDNAME=SYSIN
//SYSPRINT DD  SYSOUT=&SYSOUT
//SYSUT1   DD  UNIT=&WRKUNIT,SPACE=(1024,(400,20))
//         PEND
//*
//         EXEC S2KLINK
//SYSIN DD *
 INCLUDE LOAD(EXJOB)
 INCLUDE SYSLMOD(SYS2KJOB)
 ENTRY MSYS2K
 NAME SYS2KJOB(R)
/*
//*  THE ABOVE JCL RELINKS SYS2KJOB.  FOR MUPLINT:
//*
//*  INCLUDE LOAD(EXPLI)
//*  INCLUDE SYSLMOD(MUPLINT)
//*  ENTRY MSYS2TM2
//*  NAME MUPLING(R)
//*
//*  FOR SYS2KTPI:
//*
//*  INCLUDE LOAD(EXTSO)
//*  INCLUDE SYSLMOD(SYS2KTPI)
//*  ENTRY SCFTP2I
//*  NAME SYS2KTPI(R)
//*
//*  FOR S2KDMV6:  ADDITIONALLY, S2KDMV6 NEEDS TO BE AMODE=31, RMODE=24
//*   PARM=(MAP,LIST,'AMODE=31','RMODE=24')
//*
//*  INCLUDE LOAD(EXTSO)
//*  INCLUDE SYSLMOD(S2KDMV6)
//*  ENTRY S2KDMV6
//*  NAME S2KDMV6(R)
```

This example shows a simple Assembler Language user exit for batch SCF.

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 OS/390 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)
            BR      14                     RETURN TO GETS2K
USERSTEP    DC     CL8'XXXXXXXX'
END         EXJOB
```

Chapter 10: Analyzing Multi-User Events: The Diagnostic Log

Overview

The Diagnostic Log captures event data about database use, thread use, and user job activity that occurs in the Multi-User environment. Each message in the Diagnostic Log contains information about an event that took place during a Multi-User session. This data can help you when you are determining buffer sizes, scratch pad sizes, Update Log page sizes, and pool sizes. For PLEX programs, the Diagnostic Log can serve as a debugging tool for observing program logic. User jobs get suspended in a thread if there are conflicts between global or local holds on a database. Wait time, user suspension, and thread suspension can cause performance degradation and inefficient use of resources. The Diagnostic Log can help identify these problem areas.

The Diagnostic Log is written in the S2KDIAG data set. It can be allocated to disk, tape, or printer. Usually, disk is the preferred medium. If the Diagnostic Log is allocated to tape or disk, you can use the LOGDUMP and 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. Diagnostic Log message contents are discussed in "Format of Diagnostic Log Records" later in this chapter.

Displaying the Diagnostic Log

This discussion shows ways of displaying Diagnostic Log data and suggests some guidelines for using and interpreting the data.

After data has been written to the Diagnostic 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.
- your own code to obtain customized reports.

Specify the LOGLEVEL and LOGCOUNT parameter values that are appropriate to capture the type(s) and amount of data needed. For details, see "Generating Diagnostic Log Data: Execution Parameters" later in this chapter.

Sequential Listings

The LOGDUMP utility produces an unformatted, sequential listing of the messages written to the Diagnostic Log. This listing contains all events or any of the following subsets in any combination:

- a specific time period
- one or more specific user jobs
- one or more user IDs
- one or more thread numbers
- specific messages that are identified by message number.

For details, see "How to Use the LOGDUMP Utility" later in this chapter.

The sequential listing 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 user ID, you might be able to isolate potential problem areas that can cause poor performance or inefficient use of resources.

Database Activity Detail Report

The LOGLEVEL parameter values that are required for collecting data for this report are: LOPEN, POPEN, UNIT, USPND.

Use DIAG2000 to produce the Database Activity Detail Report. (For details, see "How to Use the DIAG2000 Utility"). The Database Activity Detail Report shows the following statistics for each database that is 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 obtain the status of locks, holds, retrievals, updates, and rollbacks on the database
- 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).

An excessive number of logical opens might indicate that 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 higher than normal, it might indicate a Multi-User database contention problem. Output 10.1 shows a sample report.

Output 10.1 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

The LOGLEVEL parameter values that are required for collecting data for this report are: LOPEN, POPEN, TSPIO, and USPND.

Use DIAG2000 to produce the Database Activity Summary Report that shows the following statistics for each database that is accessed during the Multi-User session:

- the total number of physical opens for the database
- the total number of logical opens for the database
- the total number of user suspensions for any status on that database
- the number of reads and writes for each file that is associated with the database.

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 higher I/O counts than expected.

The I/O numbers in this report are based on the number of times that message 211 occurred in the Diagnostic Log. Message 211 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. Output 10.2 shows a sample report.

Output 10.2 Database Activity Summary Report

```
                        TOTAL       TOTAL       TOTAL
DATA BASE NAME       PHYS-OPENS  LOGL-OPENS  SUSPENDS

LIBRARY                   01          156        00


LIBRARY7    01R      OOW   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

The LOGLEVEL parameter values that are required for collecting data for this report are: TSPIO, TSPND, TSTRT, UINIT.

Use DIAG2000 to produce the Thread Activity Detail Report that shows the number of times each thread was initialized and the number of times each thread had to wait for

- buffers
- databases
- tape
- I/O
- scratch pads
- Update Log(s).

This report also shows the total for all threads for each category that had waits. The number of waits for databases, tapes, and scratch pads should be low; 0 is the best. The waits for I/O should be the highest number. A problem with excessive I/O will probably not show up here. Output 10.3 shows a sample report.

Output 10.3 Thread Activity Detail Report

```
THREAD      INIT      B-WAIT     D-WAIT     O-WAIT     T-WAIT     IO-WAIT      P-WAIT     U-WAIT
 NBR        COUNT      COUNT      COUNT      COUNT      COUNT      COUNT        COUNT      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             00
TOTAL IO WAITS                  10
```

Thread Activity Summary Report

The LOGLEVEL parameter values that are required for collecting data for this report are: TSPIO, TSPND, TSTRT, UINIT.

Use DIAG2000 to produce the Thread Activity Summary Report, which shows the following statistics for each thread:

- the total time thread was active (initialized for a user)
- the percentage of total active time spent in the thread (CPU time and wait time while user was in the thread).
- the total wait time used by the thread
- the percentage of active thread time spent in the wait state
- the percentage of thread wait time spent waiting for buffers, databases, tapes, I/Os, scratch pads, and Update Logs.

All time is based on elapsed time written to the Diagnostic Log by Multi-User. The Thread Activity Summary Report can help you identify threads that are active a small percentage of the time and are probably not necessary. Wait time should form a small percentage (10 to 20%) of a thread's active time. I/O waits should be the highest percentage of the wait time. Numbers that do not conform to these patterns might indicate a problem. Output 10.4 shows a sample report.

Output 10.4 Thread Activity Summary Report

THREAD NBR	TOTAL ACTIVE TIME	PERCENT ACTIVE	TOTAL WAIT TIME	PERCENT WAIT	WAIT TIME ANALYSIS 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

The LOGLEVEL parameter values that are required for collecting data for this report are: LOPEN, POPEN, TSPIO, TSPND, TSTRT, UINIT.

Use DIAG2000 to produce the User Job Activity Report, which shows the following statistics for each SYSTEM 2000 session:

- job name
- start and stop times, as written to the Diagnostic Log
- number of physical database opens
- number of logical database opens
- number of thread initializations
- 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.

These statistics might help in isolating specific jobs that are causing excessive contention. You can also use this report to further identify a job that is shown to be causing performance problems on another report. Output 10.5 shows a sample report.

Output 10.5 User Job Activity Report

```
JOBNAME     START        STOP         PHYSICAL   LOGICAL    THREAD    THREAD      USER        IO      UPD LOG
             TIME         TIME          OPENS     OPENS      INITS   SUSPENDS   SUSPENDS   SUSPENDS   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

This section discusses the advantages and disadvantages of the various device types (tape, disk, and printer) and gives sample JCL statements for describing the S2KDIAG data set.

Allocating the S2KDIAG Data Set

The S2KDIAG file contains variable-length records that have a maximum logical record length (LRECL) of 36 bytes. Using a large blocksize to reduce disk and tape I/O to the S2KDIAG file is recommended. Subtracting 36 (a 32-byte header plus a 4-byte record length field) derives the message length (length of the variable data) from your defined LRECL. If no LRECL is specified for the S2KDIAG file, SYSTEM 2000 assigns a default LRECL of 4092. The default block size is 4096 and the maximum message length is 4060.

The use of a tape drive requires that the drive be allocated for the entirety of the Multi-User session. If the tape fills up, additional tape mounts are requested.

The value specified in the LOGCOUNT parameter indicates how many records your S2KDIAG disk file will support. After that number of records has been written to the disk file, the file is closed and then opened again to re-start at the beginning. In this instance, new records will overwrite existing records. The LOGDUMP utility re-initializes the disk file after dumping the output to a printer. If Multi-User is still actively using the S2KDIAG file when you run LOGDUMP against it, the file is initialized and the data on it is lost.

S2KDIAG DD Statement

If LOGLEVEL= NO, the S2KDIAG DD statement is not necessary. The JCL to describe the S2KDIAG file must be set up at your site. Here are some example DD statements:

```
disk          //S2KDIAG DD DSN=S2K.DIAG,DISP=(,CATLG),UNIT=3380,
(new)           SPACE=(blksz,(pri,sec))
                        (blocksize, (primary, secondary))

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 execution parameters when you initialize Multi-User.

The LOGLEVEL parameter defines the kinds of events that are recorded in the Diagnostic Log. These variables can be modified dynamically during a Multi-User session if the LOGLEVEL parameter is not originally set to NO.

The LOGCOUNT parameter defines the number of records that are written to the S2KDIAG data set before the file is re-set. This value cannot be modified dynamically during a Multi-User session.

For information about the types of events that can be logged, see "How to Use the DIAG2000 Utility" later in this chapter. Depending on what events you want to capture, set your LOGLEVEL parameter accordingly. If you write your own programs, make sure you are capturing the correct data. Table 10.1 shows the different values for LOGLEVEL and the type of data that each causes to be captured.

Table 10.1 LOGLEVEL Parameter Values

Value	Specifies
NO	no logging. If NO is specified, you cannot alter LOGLEVEL parameter during a Multi-User session.
LOPEN	the logical open or close of a database.
PDATE	the date and time in packed format. If PDATE is not specified, the default is time only.
POPEN	the physical open or close of a database.
SDATE	the date and time in SAS TODSTAMP format. SDATE is faster than the SVC-based time stamp.
TSPIO	that the thread is suspended or dispatched (I/O).
TSPND	that the thread is suspended or dispatched (except for I/O).
TSTRT	the thread start or stop.
UINIT	Multi-User initialization, user start or stop. If the LOGLEVEL execution parameter is not specified, the default is UNIT.
USEGM	the command segment start. USEGM is not used by DIAG2000.
USPND	user suspension or acquittal.

Follow these rules for coding the LOGLEVEL parameter:

- 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 = NO.
- Each message in the Diagnostic Log is time stamped. If you specify PDATE, the time and the date are combined in packed format.

  SDATE gives you an alternate method of storing the date/time field at displacement 5 in the Multi-User Diagnostic Log records. SDATE writes the date and the time in SAS TODSTAMP format and runs faster than the SVC-based time stamp. If you specify SDATE and the Diagnostic Log is disk or tape, a 'Store Clock' instruction is issued. The results are stored in the right-most 8 bytes of the 10-byte date/time field, that is, bytes 5 through 14. (SDATE uses the 14th byte, which is usually 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 an *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 format, which is an 8-byte time-of-day stamp.

- Use SDATE if you want to read the Diagnostic Log with SAS.
- The SDATE option and the PDATE option are mutually exclusive. If you try to specify both of them in the LOGLEVEL execution parameter, the WTO message S2K0111/00 is issued.

Note: The LOGDUMP and DIAG2000 utility programs do not recognize an SDATE time stamp. You must use either SAS or your own program to read disk or tape Diagnostic Log records that are created by using the SDATE format.

If the original LOGLEVEL parameter is set to a value other than NO, the operator can change the level of logging during a Multi-User session. The commands to do this are

F S2K, LOGLEVEL=*variable* [*/variable*]...
F S2K, DISPLAY LOGLEVEL
F S2K, D LOGLEVEL

The same LOGLEVEL variables that are available when using the operator command are available in the execution parameter. When modifying variables, the operator must specify the full set of variables that are to remain in effect. If LOPEN, POPEN, and PDATE are in effect and PDATE is to be turned off, you must specify LOPEN and POPEN in the operator command.

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. The format of the LOGCOUNT parameter is

LOGCOUNT=*n*

n is a positive integer that specifies the number of 1000-message increments that can be written to the Diagnostic Log. The range is from 1 to 32767. The default is 1 (that is, 1000 messages). The operator cannot modify the LOGCOUNT parameter.

When the S2KDIAG file is assigned to disk, LOGCOUNT dictates when the S2KDIAG log is re-set. To use the maximum disk space possible, set LOGCOUNT high and SYSTEM 2000 will adjust the LOGCOUNT automatically if an end-of-file condition occurs.

How to Use the LOGDUMP Utility

The LOGDUMP utility produces a sequential, unformatted listing of the messages in the Diagnostic Log. You can display all of the messages or a selected subset of them by specific job, user ID, thread, message type, or time interval. LOGDUMP reads the messages from the S2KDIAG data set and prints them to the SYSPRINT data set.

LOGDUMP is a stand-alone Assembler language program. It reads

- commands from the SYSIN data set that specify the types of Diagnostic Log messages to display.
- the disk or tape S2KDIAG data set.

Specifying LOGDUMP Commands

To specify LOGDUMP commands, use either of the following methods:

- place each command in a separate SYSIN record beginning in column 1. Do not use a comma after the command.
- place several commands in a SYSIN record beginning in column 1. Use a comma between commands.

The commands and parameters are the same for either method. If you do not specify a command, the defaults are as shown in Table 10.2

Table 10.2 LOGDUMP Commands, Command Defaults, and Parameter Values

Command	Default	Parameter	Parameter Meaning
START	beginning of the log	*yydd,hh.mm.ss*	year, day, hour.minute.second
STOP	end of the log	*yyddd,hh.mm.ss*	year, day, hour.minute.second
JOBNAME	all job names	*jjjjjjj*[,*jjjjjjjj*] . . .	job name
USER	all user IDs	*uuuu,*[,*uuuu*] …	user ID
THREAD	all threads	*tt*[,*tt*]…	thread number
MESSAGE	all messages	*iiii*[,*iiii*]…	message ID

The following rules apply to LOGDUMP commands:

- You can specify commands in any order.
- Embedded blanks are not allowed. If a blank is found, scanning for that record stops.
- If a LOGDUMP command appears more than one time, only the last command is used.
- The number of parameters that can be specified is not limited.
- To list events for a specific time period that might span several days, enter 0's for the *yyddd* value and a valid time for the *hh.mm.ss* parameter in the START and STOP commands.
- If a syntax error is found in a command, the letter 'C' is printed below the command in error, and the following message is issued

* * * SYNTAX ERROR IN COMMAND * * *

The following is 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                  *
//*  *    DSP = DISPOSITION FOR INPUT DATA SET                         *
//*                                                                   *
//*********************************************************************
//*
//LOGDUMP     PROC   INDEX=S2K,RLSE=V1,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
```

The LOGDUMP utility produces an unformatted listing in which the date appears on a line by itself, and each log message begins with the time. The first four characters of each record in the log is the record length field, which is not printed by LOGDUMP. See "Format of Diagnostic Log Records" for information about the contents of messages. See *SYSTEM 2000 Messages and Codes, Version 1, First Edition* for error messages and warnings issued by the LOGDUMP utility.

How to Use the DIAG2000 Utility

You can use the DIAG2000 utility to organize information from the Diagnostic Log into detailed and summary reports. These reports can help in analyzing Multi-User resource contention, performance, and interaction of thread, database, and job activities. You can select any or all of the following reports when you execute DIAG2000:

- Database Activity Detail Report
- Database Activity Summary Report
- Thread Activity Detail Report
- Thread Activity Summary Report
- User Job Activity Report.

All of the reports require minimum values to be specified in the LOGLEVEL parameter. See "Displaying the Diagnostic Log" for more details about report contents and parameters. Each time you execute DIAG2000, the DIAG2000 commands are displayed with any error messages or anomalies in the S2KDIAG data set. DIAG2000 also displays the count of messages in the Diagnostic Log. DIAG2000 is a stand-alone COBOL program, and the source for this program is provided on the SYSTEM 2000 delivery media.

Specifying the Input to DIAG2000

DIAG2000 requires the following input:

- the Diagnostic Log on disk or tape
- an execution time parameter passed by means of the JCL
- a set of DIAG2000 commands in the INPUT DD statement.

S2KDIAG Data Set

Table 10.3 provides a cross-reference between LOGLEVEL variables and DIAG2000 reports. If the required LOGLEVEL variables are not specified, the totals shown in the reports will probably be 0 or the report will fail completely.

Table 10.3 DIAG2000 Reports and LOGLEVEL Variables

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 indicates some totals may be incorrect if this variable is not specified.
R indicates this variable is required to generate this report.
W indicates a warning message is issued if this variable is not specified.
Z indicates invalid zeros are reported if this variable is not specified.

Note: The USEGM variable is used only for listings. It is not used by DIAG2000.

DIAG2000 commands

The DIAG2000 commands specify which reports are to be produced. These commands are specified in 80-byte records in the INPUT data set. Each DIAG2000 command must have a '1' in column 1 and a keyword in columns 2 through 5. A sample command is shown in the first column in Table 10.4. Possible values for the keyword in columns 2 through 5 are shown in the third column of Table 10.4.

Table 10.4 Summary of DIAG2000 Comments

DIAG2000 Command	Report Requested	Keywords	Meaning	Report Type
1THRD	Thread Activity Detail Report	THRD	request a Thread Activity Report	D
1DB	Database Summary Report	DB	request a Database Activity Report	S
1JOB	Job Activity Report	JOB	request a Job Activity Report	
1ALL	All Reports	ALL	request all reports	

Column 6 of Table 10.4 shows whether the command specifies a detailed (D) or a summary (S) report when the keyword is THRD or DB; otherwise, the field has no meaning.

The following rules apply when specifying JCL for executing 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.
- The INPUT DD statement defines a file for the DIAG2000 commands. This must be an 80-byte, fixed record length file (LRECL=80 and RECFM=F or FB). The file can be either inline (DD*) or a disk data set.
- The PRINT DD statement defines the main output file (LRECL=133 and RECFM=F or FB). This specifies where DIAG2000 puts the report(s).
- The ERRORS DD statement defines the print file for error messages (LRECL=133 and RECFM=F or FB).
- The SPOOL DD statement defines the scratch file used by DIAG2000 (LRECL=52 and RECFM=F or FB). This file should be about one-fourth the size of the Diagnostic Log.

Output from DIAG2000

The reports produced by DIAG2000 are discussed in "Displaying the Diagnostic Log." Additional output for the DIAG2000 execution includes a listing of the commands, error messages (if any), and a count of the number of messages in the Diagnostic Log. Output 10.6 shows a sample of this last portion of the report.

Output 10.6 Sample Report Produced by DIAG2000

```
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
```

DIAG2000 Messages and Codes

DIAG2000 issues the following types of error message:

- 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. These messages may indicate discrepancies in the S2KDIAG file.
- messages for fatal errors, that is, errors that stop the execution of DIAG2000.

DIAG2000 error messages and codes are discussed in *SYSTEM 2000 Messages and Codes, Version 1, First Edition.*

Format of Diagnostic Log Records

The Diagnostic Log messages are of variable length and have a maximum record length of 126 characters. Each message records the occurrence of an event in the Multi-User session. If the S2KDIAG data set has been re-initialized (max LOGCOUNT reached), the message

LOG INIT.xxxx TIMES

is written in columns 15 through 35 of the first log record. Table 10.5 shows the record layout of the Diagnostic Log messages.

Table 10.5 Diagnostic Log Messages Record Layout

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	Job name	Character
23-26	4	User ID	Character
27-28	2	Thread ID	Character
29-32	4	Message ID	Character
33-	Varies	Variable data (if any)	Character

Table 10.6 LOGLEVEL Messages

LOGLEVEL Parameter	Log Message Number	Variable Data (Length, if any)	Message Description
UINIT	0001	LOGLEVEL	Initial settings Multi-User Initialization started
	0002		Accounting initialized
(logged when	0003		Multi-User initialization complete
any level	0004		Sign on non-TP user
except NO	0005		Sign on TP user
is specified or	0006		Multi-User terminated during initialization
when requested			
by itself)	0007		TP user request cancelled
	0008		TP user table full; request cancelled
	0009		No memory for user control blocks
	0010		SCF session not allowed; not enough threads to start SCF session
	0011		User died for POST
	0012		Error in physical open of database
	0013		Terminate non-TP user
	0014		Terminate SCF 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 (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 (*xxx* is user priority; *yyy* is number of dispatches since user was added to queue)
	0201		Terminate thread
TSPIO	0211	*DDname*I*nnnn*	Physical I/O thread
		*DDname*O*nnnn*	Suspend thread I/O
		followed by	*DDname* identifies the file being accessed; I and O indicate WRITE and READ
		bufage clock	*nnnn* is the page number, in decimal relative to 0 (BDAM files only)
		bufaddr	*bufage* is the page age in hex, relative to x'-1'; (x'-1' indicates the oldest page; 0 indicates a freed page)
			clock is the buffer manager clock in hex, relative to x'-1'; (x'-1' indicates the earliest time)
			bufadr is the page address in hex
	0212	*xxx yyy*	Dispatch thread I/O

(*continued*)

Table 10.6 LOGLEVEL Messages (*continued)*

LOGLEVEL Parameter	Log Message Number	Variable Data (Length, if any)	Message Description
USPND	0303	LOCK *DBname* HOLD *DBname* RETR *DBname* OPEN *DBname* FRAME *DBname* ROLBK *DBname* NLSES *DBname* NKUPD *DBname* HOLDL *DBname* LHSES *DBname* RESETR *Dbname*	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
	0203	database request (6 char)	Wait for database
	0204	(4 char)	Wait for overlay
	0205		Wait for tape
	0210	ddname page #	Wait for I/O
			Suspend thread until the resource is released
	0206	*xxx yyy*	Buffer or I/O
	0207	*xxx yyy*	Database
	0208	*xxx yyy*	Overlay
	0209	*xxx yyy*	Tape
			Dispatch thread that was waiting for the specified resource
	0214		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 is now dispatched to continue processing
	0216		User suspended in thread for 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 is now dispatched
USEGM	0300	*opcode dbn ssr*[1]	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 opcodes 07 and 12)
	0399	*opcode dbn ssr*	Secondary database names on FRAME

[1] In message 300 above, *opcode* is usually displayed as the internal number of the operation. For example, 91 represents a START S2K command. By using the OPT052 execution parameter, you can change this field to character representations of the PLEX operation codes instead of numerical representations. When OPT052=NO or is not specified, *opcode* shows up in the 300 message as 91 for a START S2K command. When OPT052=YES, *opcode* shows up in the 300 message as START S2K.

Format of LOGLEVEL Setting Display

The Multi-User 0001 initialization record (initial LOGLEVEL settings) and the message 0031 (modified LOGLEVEL settings) have a 2-byte variable data field that starts in column 29. The bit pattern of the 2-byte field is displayed. The first 4 bits are not used. Bits 5 and 6 indicate the type of date format; these bit settings are useful when displaying the date and time. The date/time format is based on the LOGLEVEL setting and is initially set by S2KPARMS. The remaining bits indicate other LOGLEVEL settings, as shown next

```
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
```

Chapter 11: Multi-User Console Operator Commands

Introduction

The console operator communicates with Multi-User by using the MODIFY (or F) command. The alternate console operator uses the S2OP command instead of the MODIFY command. The commands and their results, which are the same for both console commands and the Multi-User responses, are written to the console log and to the Multi-User job log. Console input does not need Multi-User TP to be enabled, but S2OP uses some TP functions for dependent region communications and Multi-User dispatches.

Master console commands are submitted to Multi-User by using the operator command MODIFY. If your Multi-User job name is S2K, the syntax to submit a DISPLAY ACTIVE (D A) command is F S2K, D A. The MODIFY (or F) command and the job name must accompany each command.

The alternate console command S2OP can be called as a TSO program, a batch program, or a CICS transaction. The functionality is the same in all environments.

Batch execution of S20P requires allocations of S2KMSG and S2KCOMD files. S2KMSG can be a SYSOUT data set, or it can have a DCB with RECFM=VB,LRECL=133,BLKSIZE=137. S2KCOMD can be a SYSIN data set, or it can have a DCB with RECFM=FB,LRECL=80. If you do not allocate the S2KMSG file, it is dynamically allocated. If you do not allocate S2KCOMD, user abend 517 occurs. An end-of-file (EOF) in the S2KCOMD file terminates batch S2OP. A NULL command terminates S2OP in all other environments.

The S2KCOM file must be allocated for XMS Multi-User when using S2OP. S2KCOM is not dynamically allocated.

Syntax

|MODIFY |F	*job name , text*	(*master console)*
S2OP	*text*	(*alternate console)*

job name is the SYSTEM 2000 Multi-User job name. In the examples given in this section, the job name is S2K. *text* contains the values that are described in the following sections. *text* can contain embedded blanks.

Use the plus sign (+) to repeat a previous console command from an alternate terminal. The plus sign (+) must be the only character on the line. This option is not available for the master console.

Console Commands

Use console commands to perform the functions that are described in the following sections.

Display Status of Open Databases

DISPLAY STATUS (D S) displays the status of each open database in Multi-User. The retrieval and local-hold status are not displayed; either of these statuses, or no status, has a hyphen (-) in the status column and blanks in the job column. The four statuses that are displayed show the job name and the status for that job name in the job column. See Table 11.1.

Table 11.1 Status-Column Code in Output for *job name*

Status Indicator	Function	Means that…
L	Lock	the database is opened for exclusive use. The job name for that exclusive use is listed under JOB.
R	Rollback	the job that is listed in the job column is in rollback.
H	Hold	this is a global hold. Retrievals are allowed. No other type of hold, including local or update status, is allowed while the global hold is in effect. Activities that cause a global hold are GETx/HOLD in PLEX queue mode, the where-clause phase of an SCF update command; FRAME; TERMINATE of Q/T session that has SCF updates; SAVE; KEEP; and re-set of the Rollback Log.
U	Update	a PLEX or an SCF update is in progress.

To request the status of all databases, use the command `F S2K, D S.` Output 11.1 shows the results. A line-by-line explanation of the results follows the output.

Output 11.1 Results of Status Request for All Databases

```
      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/-1- DBFOUR       001   001   H     JOBPQR       -
      S2K1430/01- DBFIVE       002   001   R     JOBABC       -
```

The first line of output contains the column headings. PLX is the PLEX user count, and SCF is the SCF user count, which includes any SCF TP users who are signed on to the system. In Output 11.1,

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

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

DBTHREE has user JOB123, who has update status for the database.

DBFOUR has user JOBPQR who has a HOLD on the database.

DBFIVE has user JOBABC who is performing a rollback on the database.

If there are no active databases, the display output is `'NO ACTIVE DATABASES'.`

Display Jobs Signed On to Multi-User

DISPLAY ACTIVE (D A) lists each job that is signed on to Multi-User. Output shows whether the job is SCF or PLEX, and usually shows what the job is doing.

There are five activity indicators and several resource indicators that expand the scope of the activity. For example, the message `W. UPDATE EMPLOYEE` means that the user is waiting with no thread to update the EMPLOYEE database. The job is suspended while the user waits to acquire update status.

Table 11.2 Multi-User Activity Indicators

Activity Indicator		Function
E		executing
N		user has a global hold on a database but is not currently active in the Multi-User region. For example, a FRAME has executed successfully, but the program has not yet returned to execute another PLEX command. The database remains in HOLD status.
WT		waiting in a thread
	|TAPE	SAVE, RESTORE, KEEP, or APPLY needs a tape
	|PAD	user needs an allocation unit
	|BUFFER	user needs a buffer
	|UPDLOG	user needs access to an Update Log
W.		waiting (suspended) with no thread
.	(period)	not executing and not waiting for resources (no database hold)

Table 11.3 Multi-User Resource Indicators

Resource Indicator	Function
EXECUT	indicates the database is not in retrieval, hold, or update mode.
FRAME	causes the job to wait to FRAME a database. The job must wait until all current updates and global holds are finished.
HOLD	allows global hold retrievals and prevents updating until the hold is relinquished.
LHOLD	causes the job to wait to obtain a local hold on a specific record. The job must wait until the user, who is currently holding the record, relinquishes it or a HOLD is dropped.
LOCK	requests exclusive use of a database, and issues a return to TP if use is not granted Use the LOCK command in PLEX; use the EXCLUSIVE DBN command in SCF.
LOCKR	requests exclusive use of a database, and issues a return code to PLEX user programs if use is not granted. Use the LOCKR command in PLEX; not applicable in SCF.
NKUPD	causes the job to wait to do a non-key update. The job must wait until all updates and global holds are finished.
NO ACT	indicates no SYSTEM 2000 activity is requested. The user program is currently active in the dependent region.
OPEN	requests non-exclusive use of a database, and issues a return to TP if use is not granted. Use the OPEN command in PLEX; use the DBN command in SCF.
OPENR	requests non-exclusive use of a database, and issues a return code to PLEX user programs if use is not granted. Use the OPENR command in PLEX; not applicable in SCF.
RETREV	puts the database in retrieval mode.
ROLBK	causes the job to wait to get control of the database to perform recovery. The job must wait until all current updates, retrievals, and global holds are finished.
UP ACT	indicates user-partition activity. A thread is tied up on behalf of the user, but control has passed to the user partition. For example, this could happen if a message for the S2KMSG data set is returned to a batch user.
UPDATE	updates the database. The duration is the completion of the command.

To request the status of all active jobs, use the command `F S2K,DISPLAY ACTIVE` Output 11.2 shows the results. A line-by-line explanation follows the output.

Output 11.2 Results of Status Request for All Active Jobs

```
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
```

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 job name 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 job name 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 job name or terminal ID
	000	=	job priority
	W.	=	waiting in the input queue
	LOCK	=	resource request pending on DBTHREE
TER123	TP	=	inactive TP job using database DBTWO

If one of the active users is an alternate console, the user type is ALT CON. To request the status of all active jobs, Use the command `F S2K,D A,S`, which includes the STATUS operand. 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 is listed with the job status. S2K1429 is a column heading. Output 11.3 shows the results.

Output 11.3

```
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   -
```

Display Status of a Specific Job

To display the status of a specific job, use the following syntax. The optional STATUS operand displays the name and status of each database that the job has opened.

|MODIFY |F	S2K,	|DISPLAY |D	blank	|*terminal - ID* |*job name*	| ,STATUS | ,S

To display the status of a specific SYSTEM 2000 job (in this example, JOBABC), use the command `F S2K,DISPLAY JOBABC`, which does not include the STATUS operand. Output 11.4 contains the results, which show that JOBABC is a PLEX program that is executing database updates.

Output 11.4 Results of Request

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

To display the status of the SYSTEM 2000 job JOBABC, use the command `F S2K,D  JOBABC,STATUS`, which includes the STATUS operand. Output 11.5 shows the results. This output is the same as Output 11.4 except that this shows that JOBABC has two databases open and is updating database DBFIVE. The total number of SCF and PLEX users for each database is also shown.

Output 11.5 Results of Request to Display Status of a Specific Job (with STATUS operand)

```
S2K1432/00- JOBABC   PLX      000 E  UPDAT  -
S2K1429/00-       DBN        PLX SCF S    JOB    -
S2K1431/00- DBONE            001 002 -
S2J1430/00- DBFIVE           002 001    JOBABC   -
```

Change the PQA Setting

To change the Priority Queuing Algorithm (PQA) execution parameter, use the following syntax. Output 11.6 shows the results.

|MODIFY	S2K,PQA=	|PRTY [/*n*/*y*]
|F		|FIFO

PRTY [/*n* /*y*] orders user queuing by user priority. *n* is the number of requests dispatched from the dispatch queue before the software examines the queue for priority adjustments. If *n* is not specified, the default is 0. 0=*n*=999. *y* is the number that becomes the increment when a user priority is being increased. Multi-User examines the dispatch queue every *n* dispatch. Any user not dispatched in the last *n* dispatches has his priority increased by *y* to a maximum of 255. 0 < *y* = 255. FIFO orders user queuing on a first-in, first-out basis. FIFO is the default.

Output 11.6 PQA Output Message

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

Display the Current PQA Setting

To display the current PQA setting, use the following syntax. For example, the command `F S2K, D PQA` results in one of the replies shown in Output 11.7.

|MODIFY	S2K,	|DISPLAY	PQA
|F		|D	

Output 11.7 Possible Replies Resulting from the Current PQA Setting

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

Activate SCF TP Facility

To activate the SCF TP facility after Multi-User has been initialized, use the following syntax. For example, the `F S2K, START TP` command activates SCF TP. Output 11.8 shows the results if the request was successfully executed.

|MODIFY	S2K,	|START	*blank*	TP
|F		|S		

Output 11.8 Results of Successful Request to Activate SCF TP

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

Output 11.9 Results of Successful Request to Quiesce SCF TP

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

Output 11.10 Results of Unsuccessful Request

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

Change Level of Diagnostic Log Messages

To change the level of diagnostic log messages, re-set the LOGLEVEL execution parameter settings by using the command `F S2K,LOGLEVEL=UINIT/TSTRT`. When changing levels of messages, you must re-specify all levels that are to be in effect. For example, use either of the following commands to request a display of the active settings of the LOGLEVEL parameter: `F S2K,DISPLAY LOGLEVEL` or `F S2K,D LOGLEVEL`

Change Status of Segment Statistics

To dynamically change the status of Multi-User Accounting Log segment statistics, use the following syntax. For example, to start segment statistics for PLEX jobs, use the command `F S2K,START STAT PL`. However, to start and stop logging of statistics, commands cannot re-set the thresholds that were set by the execution parameters. For more details about segment statistics, see the chapter "Generating Accounting Data: SYSTEM 2000 Execution Parameters" earlier in this book.

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

Cancel Specific Job with an Optional Dump

To cancel a specific job and generate a dump, use the following syntax:

```
|MODIFY    S2K    |CANCEL    blank    job name    [ , DUMP]
|F                |C
```

job name is the name of the specific job or terminal ID to be cancelled, and DUMP generates a dump of the Multi-User region before the specified job name is cancelled. If the job name does not exist, a dump is still taken, and a message appears, which informs you that the job was not found. Specifying the DUMP option when cancelling a job is equivalent to issuing the DUMP command followed by a CANCEL *job name* command.

Canceling a specific job does not damage a database. The job is allowed to finish the current command if updating a database; otherwise, cancellation is immediate. A CANCEL causes user abend code 551. For example, the command `F 2K,CANCEL JOBABC` cancels job JOBABC. Output 11.11 shows the results if job JOBABC is updating a database.

Output 11.11 Results of Cancel Request of a Specific Job During Update

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

Output 11.12 shows the results if the job is not found.

Output 11.12 Results of Cancel Request if Job Is Not Found

```
S2K1404/00- JOB job name NOT FOUND -
```

Cancel More Than One Job

To cancel more than one job at a time or to cancel a specific terminal ID for a specified job name, use the following syntax. If an UPDATE command is being processed, the job is marked for cancellation and is not cancelled until the update finishes. The CANCEL JOB command is convenient for CICS jobs because it allows you to use a single command to cancel all jobs included under a specified job name. For example, CANCEL JOB=CICSPROD,* cancels all jobs associated with CICSPROD.

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

job name is the name of the job. The asterisk (*) means to cancel all jobs, including duplicates, that are active under the specified job name. *termid* means cancel a specific TP terminal ID for that job name, including duplicates.

Dump without Terminating Multi-User Session

To request a dump without terminating the Multi-User session, use the following syntax. The DUMP command generates a dump of the Multi-User region without disturbing any Multi-User activities.

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

Terminate Multi-User or TP Session

To terminate Multi-User or TP, use the following syntax. The CANCEL S2K command allows you to terminate a Multi-User or a TP session without risk of damage to database(s). After the command is issued, no new SYSTEM 2000 users (or SCF TP users if TP was specified) are allowed in the system. (The default is WHEN, without a dump). Multi-User regards SCF TP as being active when any terminal user is still signed on to the system. Therefore, the CANCEL S2K command does not terminate Multi-User if any TP user is logged on. S2K, W or C TP,W prevents any new TP users from signing on. TP users who attempt to sign on after a cancellation has been requested receive a message that the system is being quiesced. When TP has terminated, WTO message S2K1424/00 is issued. When the Multi-User session is terminated, S2K1215/00 is issued. Exactly when Multi-User or TP terminates is dependent on the WHEN, NOW, and FORCE options. Output 11.4 shows all the options that can be used in the CANCEL S2K command and their functions.

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

Table 11.4 Options that Terminate Multi-User or TP

Option	Function
W WHEN	ends Multi-User when all current users have finished their individual sessions and have exited from Multi-User. No new users are allowed to sign on.
N NOW	cancels, immediately, all jobs that are not in update status. Jobs in update status are allowed to finish. When all jobs (or SCF TP jobs) have been cancelled, Multi-User (or SCF TP) terminates.
FORCE	causes abends to Multi-User without doing any cleanup. It is equivalent to an OS/390 cancel and disables all ESTAE processing. Therefore, try to cancel Multi-User by using the WHEN or the NOW option before using the FORCE option. The FORCE option causes a SNAP dump with ID of 61. **Note:** The FORCE option can cause damaged databases, and it can only be used with the S2K option. It cannot be used to cancel SCF TP.
DUMP	causes a SNAP dump. The DUMP option must be preceded by appropriately positioned parameters, that is, C S2K,DUMP is invalid but C S2K,W,DUMP is valid.

Table 11.5 shows the syntax for various forms of the CANCEL (or C) command and the message that is issued for each command.

Table 11.5 CANCEL Commands and WTO Messages

Sample Commands	WTO Message
F S2K, C 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 user.)
F S2K,C TP,NOW	S2K1420/00- SCF TP ACTIVITY BEING TERMINATED - (cancels each SCF TP user 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 users have exited the system.)

Use START TP to resume TP activity after a CANCEL TP command was issued. SCF TP also starts at Multi-User initialization time if the execution parameter STARTTP=YES is specified.

Vary Databases Offline or Online

To vary a database offline or online, use the following syntax. VARY OFFLINE allows you to force the physical close of a database when the usage count is 0 (no users on database). Also, if the files are dynamically allocated by SYSTEM 2000, the files are de-allocated.

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

database is the name of the database to be flagged offline (de-allocated) or online (allocated).

If the database is listed in S2KDBCNT, the offline lock is set to 2. This prevents the files from being re-allocated until VARY ONLINE is invoked. The lock also serves as a pending indicator when users are working in the database, and it cannot be closed. When a close is pending , users who have the database open can continue; new users are denied access. If the database is not listed in S2KDBCNT, you can vary it offline, but it is not under protection of database lockout. A database that is varied offline is unavailable until you vary it online.

Start Accounting

To re-start accounting after it has been suspended, use the following syntax. When both accounting files fill up, the Accounting process is suspended. This command should be issued after one or both of the Accounting files have been dumped.

```
|MODIFY        S2K,        START ACCOUNTING
|F
```

Cancel All Users on a Database

To cancel all users working in a database, use the following syntax. This console command is useful in conjunction with the VARY OFFLINE console command because 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 but is not cancelled until the update finishes. If the requested database is not open, the following WTO message 1447 is issued: S2K1447/*sid-database* DATABASE NOT OPEN -

```
|MODIFY        S2K,        |CANCEL DATABASE=        database
|F                         |C
```

database is the name of a database.

MUSTATS Console Operator Commands

The Multi-User status (MUSTATS) commands (see Table 11.6) 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. You will find details about the individual commands later in this chapter. The commands are presented in alphabetical order. 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.

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.

This example illustrates a command that requests that the MUSTATS DBNS command be repeated.

```
#DBNS
     …          …          …          …
+
     …          …          …          …
```

MUSTATS Command Syntax

Here is the general format for MUSTATS console commands. The pound sign (#) is an alias for SEND MUSTATS.

```
|SEND MUSTATS
|#
```

If you submit MUSTATS commands from the master console under OS/390, you must precede each command with one of the following. Alternate consoles do not need these identifiers.

```
|MODIFY job name,
|F job name
```

In the following examples, $MU is the Multi-User job name. The first two commands are used on a master console under OS/390.

```
F $MU, #DBNU=
MODIFY $MU , SEND MUSTATS DBNU=
#DBNU=                                  alternate console
SEND MUSTATS DBNU=                      alternate console
```

Table 11.6 Summary of the MUSTATS commands

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

BUFFERS

Displays information about current buffer usage.

Syntax

#BUFFERS

Details

The BUFFERS command has no parameters. The output shows one line of information for each defined buffer.

Output 11.13 Output from the BUFFERS Command

```
- BUFFERS                                          I L S D M L
-                                                 O O C B 0 0
- 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 8-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 re-assignment 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 has been modified. Y means YES.

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

DBNS

Displays the names of all databases currently open in the Multi-User environment.

Syntax

#DBNS

Details

The DBNS command has no parameters. The DBNS output shows a list of database names, which appears in the format shown here.

```
-DBNS        ①          -2                1          -2
```

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

② indicates whether the database is defined as small (S) or large (L).

DBN=

Displays information about a specific database that is open in the Multi-User environment.

Syntax

#DBN=*database*

Details

The DBN= command requires one parameter that specifies the name of a database. The output shows one line for each database file. Six lines always appear for database Files 1 through 6. Seven lines appear if File 7 (update logging) is activated. Eight lines appear if File 8 (rollback logging) is enabled.

```
-DBN=    ①
 # CISIZE    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 CISIZE 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 software is enabled. If XBUF 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 software is enabled. If XBUF 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 is re-set. 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 the total pages in File 8. If this threshold is exceeded, the Rollback Log is re-set. The default is 50%.

⑭ is a threshold number of synchpoints. If this threshold is exceeded, the Rollback Log is re-set. 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=

Displays users accessing a specified database in the Multi-User environment.

Syntax

#DBNU=*database*

Details

The DBNU= command requires one parameter that specifies the name of a database. The output shows one line of information for the specified database.

```
-DBNU= ①
 ②          ②          ②          ②          ②
```

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

② is the ID of the user who has this database open.

DBSTAT=

Displays the various database permissions currently in effect for the specified database.

Syntax

#DBSTAT=*database*

Details

The DBSTAT= command requires one parameter that specifies the name of a database. The output from the DBSTAT command is shown next.

```
-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 the count of SCF users who have this database open.

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

④ is the count of users doing retrievals.

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

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

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

⑧ is the ID of the user who has exclusive use of this database. (The PLEX syntax is OPEN/LOCK.)

⑨ is the ID of the user who has rollback permission on this database, that is, the ID of the user who began the rollback processing.

⑩ is the ID of the user who has a global hold on this database.

⑪ is the ID of the user who has update permission on this database.

⑫ is the ID of the user who has non-key update permission on this database.

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

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

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

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

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

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

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

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

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

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

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

㉔ indicates if a user is waiting to open this database for exclusive use (The PLEX syntax is OPEN/LOCK.)

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

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

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

㉘ is the ID of the user who 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 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 that the user who is in field ㉘ is waiting for a local hold on the record that is identified in field ㉙. A blank means that the record identified in field ㉙ is the last record held by the user identified in field ㉘.

HELP

Displays a list of available MUSTATS commands.

Syntax

#HELP

Details

The HELP command has no parameters. The output from the HELP command is shown next.

```
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
USERS=  -- DISPLAY INFORMATION ABOUT ALL USERS
WHY=    -- DISPLAY REASON FOR USER WAIT
```

MLH=

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.

Syntax

#MLH=*database*

Details

The MLH= command requires one parameter that specifies the name of a database. The output shows a list of the records held and the ID of the user who is holding each record.

```
-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 Return Code 110 appears.

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 ID of the user who has a hold on this record.

⑤ is the ID of the user who has a hold on 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

Displays information about scratch pads.

Syntax

#PADS

Details

The PADS command has no parameters. The output from the PADS command for each scratch pad that is defined is shown next.

```
-PADS
  PAD ①  USE       ②         ③ %  HI     ④        ⑤ %
      CISIZE    # AU'S   BLKS/AU      AU'S   PAGES
         ⑥         ⑦        ⑧           ⑨      ⑩
     PRIMARY    SECONDARY  UNIT       EXT
        ⑪         ⑫         ⑬          ⑭
```

① the scratch pad number.

② the number of allocation units in use.

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

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

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

⑥ the CISIZE for scratch pads.

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

⑧ the number of blocks per allocation unit.

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

⑩ the number of pages formatted.

⑪ the primary allocation, expressed in the units shown in field ⑬.

⑫ the secondary allocation, expressed in the units shown in field ⑬.

⑬ 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 the numeral 0 or 1. A 0 means no secondary extents have been used. A 1 means one (or more) secondary extents have been used.

POOLS

Displays information about pool definitions and usage.

Syntax

#POOLS

Details

The POOLS command has no parameters. Each line of output displays information about one defined pool.

```
-POOLS #  CISIZE  NUM FREE USE     POOLSZ   RPM   WPM   DPM
       ①      ②     ③    ④  ⑤          ⑥    ⑦     ⑧     ⑨
```

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

② the CISIZE of the buffers in the pool.

③ the number of buffers defined for the pool.

④ the number of unassigned buffers in the pool.

⑤ the usage defined for the buffers in the pool. Position 1 is S or D for scratch files or database files. 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 the size of the pool in bytes, that is, CISIZE (field ②) times the number of buffers in the pool (field ③).

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

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

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

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

QUEUES

Shows how many users are in the four Multi-User work queues.

Syntax

#QUEUES

Details

The QUEUES command has no parameters. The output from the QUEUES command contains the information shown next.

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

① 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 a count of users in the internal dispatch queue. These users have completed an I/O operation or were suspended and are 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 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 a count of users who are waiting for an I/O operation to complete.

THREADS

Displays current and high thread usage.

Syntax

#THREADS

Details

The THREADS command has no parameters. The output from the THREADS command contains the information shown next.

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

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

② the code that shows the current thread usage.

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

USER=

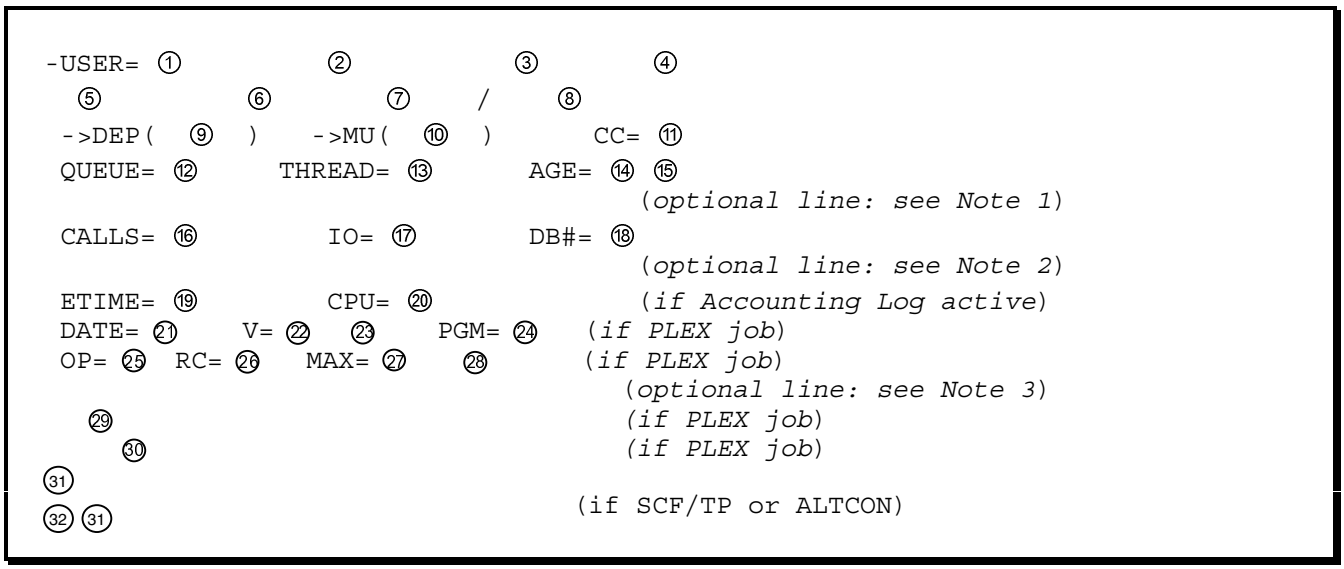
Displays information about a specified user.

Syntax

#USER=*userid*

Details

The USER= command requires one parameter that specifies a user ID. The output from the USER= command is shown next.



① the user's job name.

② the user's step name.

③ the user's program name.

④ the user's terminal name.

⑤ 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 indicates TSO if the user is running under TSO.

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

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

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

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

⑪ the condition code in hexadecimal if the user has been terminated.

⑫ the work queue that the user is on (for example, SUSPEND, SVC, INTERNAL, I/O or NONE).

⑬ the number of the thread assigned to this user.

⑭ 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 the resource that this user is waiting for if the user is on the suspend queue.

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

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

⑱ the number of databases that this user has open.

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

⑳ the amount of CPU time charged to this user (for PLEX users). The line that contains 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. (PLEX jobs only)

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

㉓ the language that was used in this PLEX program: PLI, PLIF, COB, FORT, or ASM; for PL/I, COBOL, FORTRAN, or Assembler, respectively (for PLEX jobs only.)

㉔ the alternate interface name that was recognized by the PLEX processor, if present. (PLEX jobs only)

㉕ the operation code (in hexadecimal) of the last command sent to Multi-User software. It comes from the S2KDUM control block in the user PLEX program. (for PLEX jobs only)

㉖ the last return code returned to S2KDUM in the user PLEX program. (PLEX jobs only)

㉗ the number of Locate Files requested by the PLEX program in the START S2K command. (PLEX jobs only)

㉘ 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. (PLEX jobs only)

㉙ the database name for the last PLEX command, which is taken from the COMMBLOCK in the PLEX program. (PLEX jobs only)

㉚ the subschema name for the last PLEX command, if present. (PLEX jobs only)

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

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

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

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

PQA=PRTY JOB PRTY= *x* DISP= *y* ADJ= *z*

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

Note 3: If the operation code that Multi-User is processing differs from the code in S2KDUM, the following line appears after the OP= line (PLEX jobs only). 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 opcode INSERT, but OP in the previous line of output is the command in the other stack.

PIOP= *opcode text*

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

User swapped out For an SCF/TP user who is swapped out to the S2KUSERS file, the output from the USER= command is shown next.

```
-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 user ID that you specified in the USER= command cannot be found, the output appears as shown next.

```
-USER= ①                        USER NOT FOUND
```

① is the user's job name.

USERS

Displays one line of data for each user found.

Syntax

#USERS

Details

The USERS command has no parameters. The output from the USERS command is shown next.

```
USERID  TYP  ACT  CALL  #I/0  CPUS   LAST COMMAND  RC
  ①      ②   ③     ④    ⑤    ⑥          ⑦       ⑧
```

① the 8-character user's job name

② the 2-character code that shows the type of job. It can be any of the following:
PL - PLEX
SC - SYS2KJOB
TP - SCF/TPI
AC - ALTCON

③ the 3-character code that shows user activity. It can be any of the following:
UPA - user partition activity
NOA - no activity
INA - inactive
EXE - executing

④ the 4-character code that shows the number of calls that the user has made to Multi-User software.

nnnn is the number of calls between 0 to 9,999.

nnnK is the number of calls between 10,000 and 999,999. The three right-most digits are truncated. For example, 39,999 is displayed as 39K.

nnnM is the number of calls greater than 999,999. The six right-most digits are truncated. For example, 42M.

⑤ the 4-character code that shows the number of I/Os performed for this user. The format is the same as that for CALL (field ④).

⑥ the 6-character value that specifies the amount of CPU time charged to this PLEX user if the Accounting Log is enabled. The format is the same as that for CALL (field ④), except that values less than 10,000 have one decimal point so that they display tenths of a second.

⑦ the 14-character operation code for PLEX or the 18-character operation code for SCF that displays the opcode as a mnemonic for PLEX or the last command for SCF.

⑧ the 3-character code that is the last return code from the user's PLEX program (PLEX only).

Here are some examples of output from the #USERS command.

```
S2K1916/01- MUSTATS/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\ -
S2K1916/01- MUSTATS- USERID TYP ACT CALL #I/0  CPUS   LAST COMMAND  RC  -
S2K1916/01- MUSTATS-DCNTJW   AC EXE   1    0    0.00 #USERS             -
S2K1916/01- MUSTATS-DCNTJW2  PL EXE   8  340    0.73 LOCATE DYNAMIC 000 -
S2K1916/01- MUSTATS><><><><><><><><><><><><><><><><><><><><><><><><><><>-
```

```
S2K1916/01- MUSTATS/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\-
S2K1916/01- MUSTATS- USERID TYP ACT CALL #I/O  CPUS   LAST COMMAND  RC -
S2K1916/01- MUSTATS-DCNBWC1  TP EXE   2   37    0.13 TALLY C2 WH C1 EQ -
S2K1916/01- MUSTATS-DCNTJW   AC EXE   1    0    0.00 #USERS            -
S2K1916//H- MUSTATS-DCNTJW2  PL NOA  12  531    1.11 GET S2KCOUNT   000-
S2K1916//H- MUSTATS><><><><><><><><><><><><><><><><><><><><><><><><><><-
```

```
S2K1916/01- MUSTATS/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\-
S2K1916/01- MUSTATS- USERID TYP ACT CALL #I/O  CPUS   LAST COMMAND  RC -
S2K1916/01- MUSTATS-DCNTJW   AC EXE   1    0    0.00 #USERS            -
S2K1916/01- MUSTATS-DCNTJW2  PL EXE  20  745    1.56 LOCATE DYNAMIC 000-
S2K1916/01- MUSTATS-DCNBWC1  SC UPA   2    4    0.04 USER,DEMO:DBN IS E-
S2K1916/01- MUSTATS><><><><><><><><><><><><><><><><><><><><><><><><><>>-
```

WHY=

Displays the reason for this user wait.

Syntax

#WHY=*userid*

Details

The WHY command requires one parameter, which specifies the ID of the user who is waiting. Ouput from the WHY= command is shown next.

```
-WHY=  ①          ②              ③          ④
     ⑤                          ⑥
```

① is the ID of the user who 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 *userid* is waiting for the permission shown in field ④.

⑥ is more information about the reason shown in field ⑤, such as a *userid* or a count of local holds.

Chapter 12: SAS/ACCESS Interface to SYSTEM 2000 Software

Introduction

This chapter introduces you to SAS/ACCESS software and some of its terminology and briefly describes how to use the interface. Complete documentation for using SAS/ACCESS software, including the use of access descriptors, view descriptors, and SAS data files, can be found in *SAS/ACCESS Interface to SYSTEM 2000 Data Management Software*.

SAS/ACCESS Software

SAS/ACCESS software provides an interface between SAS software and SYSTEM 2000. The SAS/ACCESS interface lets you

- create SAS/ACCESS descriptor files using the ACCESS procedure.
- directly access data in SYSTEM 2000 databases from within a SAS program by using the descriptor files created with the ACCESS procedure.
- extract SYSTEM 2000 data and place it in a SAS data file by using the ACCESS procedure or the DATA step.
- load data into a SYSTEM 2000 database by using the DBLOAD procedure.
- update data in SYSTEM 2000 databases by using the SQL procedure, SAS/FSP, or the APPEND procedure.
- directly access data in SYSTEM 2000 databases by using SYSTEM 2000 statements in the QUEST procedure.

The SAS/ACCESS interface consists of four parts:

- the ACCESS procedure, which you use to define the SAS/ACCESS descriptor files
- the interface view engine, which enables you to use SYSTEM 2000 data in SAS programs in much the same way that you use SAS data files
- the DBLOAD procedure, which enables you to create and load SYSTEM 2000 databases using data from SAS data sets
- the QUEST procedure, which enables you to access SYSTEM 2000 and issue SYSTEM 2000 statements from SAS.

The ACCESS procedure enables you to describe SYSTEM 2000 data to SAS by creating access descriptors and view descriptors. After creating these descriptor files, you can use them in SAS programs in the same way that you would use SAS data files. You can print, plot, and chart the data described by the descriptor files, use the files to create other SAS data files, and so on.

The interface view engine is an integral part of the SAS/ACCESS interface, but you do not have to work directly with the engine. SAS automatically interacts with the interface view engine when you use SYSTEM 2000 data in your SAS programs, so you can use SYSTEM 2000 data just as you would use SAS data.

You might need to combine data from several SYSTEM 2000 databases; external databases, such as DB2; Version 6 SAS data sets; and Version 7 SAS data sets. Such combinations are not only possible, but they are also easy to perform. SAS can distinguish between SAS data files, SAS/ACCESS descriptor files, and other types of SAS files, and then use the appropriate access method.

SAS/ACCESS Descriptor Files

SAS/ACCESS descriptor files are the tools that SAS/ACCESS software uses to establish a connection between SAS and SYSTEM 2000. To create these files, you use the ACCESS procedure.

There are two types of descriptor files: **access descriptors** and **view descriptors**. Creating and editing these files is explained in detail in the chapter "Defining SAS/ACCESS Descriptor Files" in *SAS/ACCESS Interface to SYSTEM 2000 Data Management Software*.

Note: When you create or load data into a database, the DBLOAD procedure creates these descriptor files for you.

Access Descriptor Files

Access descriptor files have the member type ACCESS. Each access descriptor holds essential information about one SYSTEM 2000 database that you want to access, for example, the database name, the item names, and the item types. It also contains corresponding SAS System information, such as the SAS variable names, formats, and informats. Usually, you have one access descriptor for each SYSTEM 2000 database, or you can have one access descriptor for each of several SYSTEM 2000 passwords. However, you cannot create a single access descriptor that references two SYSTEM 2000 databases.

View Descriptor Files

View descriptor files are sometimes called SAS data views because their member type is VIEW. In this book, view descriptor is used to distinguish these files from views that are created by the SQL procedure.

Each view descriptor can define all of the data or a specific subset of the data that is described by one access descriptor (and, therefore, one SYSTEM 2000 database). A view descriptor defines a single path in the database. If your database has three disjoint paths, you need to create three view descriptors to define the entire database. In addition, you might want to use only three or four possible items and only some of the values stored in the items. The view descriptor allows you to do this by selecting the items that you want to use, and by specifying selection criteria to collect only the specific data that you want. Usually, you have several view descriptors selecting subsets of data for each access descriptor that you created.

The SQL Procedure

You can join SYSTEM 2000 databases by using the SQL procedure. The SQL procedure can join SAS data files, PROC SQL views, and SAS/ACCESS view descriptors.

The DBLOAD Procedure

The DBLOAD procedure can be used to create and load SYSTEM 2000 databases using SAS data. The DBLOAD procedure enables you to

- create a new database definition only.
- create a new database definition and load data.
- load new logical entries into an existing database.
- insert new data records into existing logical entries.

The QUEST Procedure

The QUEST procedure provides SAS with interactive access to a SYSTEM 2000 database by using SCF syntax. Commands are submitted from the SAS Program Editor, output is viewed in the Output window, and messages are viewed in the Log window.

Chapter 13: XBUF Data Space Feature

Overview

I/O processing remains a costly bottleneck for many sites. Version 1 of SYSTEM 2000 offers the Extended Buffer Manager (XBUF) feature, which allows you to maintain large volumes of database data in individual data spaces.

Note: An XBUF data space is similar in concept to an OS/390 data space created by the DPSERV macro.

XBUF data space resources are backed by a combination of central and auxiliary storage. Instead of reading from DASD, most I/O requests are read from central or auxiliary storage. In effect, this technique can eliminate many I/O operations and reduce execution time. In addition, XBUF gives you control over which databases or database files are to use the individual data spaces.

Implementing XBUF

SYSTEM 2000 has three macros to help you implement XBUF. Before using these macros, you should be familiar enough with IBM Assembler language to know about column restrictions, spacing, capitalization, and so forth. Also, you should know how to assemble and link the resulting table into a load library. The three macros are

- The XB macro, which starts and ends a set of specifications.
- The XBMEMORY macro, which defines a data space.
- The XBDD macro, which defines the databases that will use the data space.

Although XBUF can allocate up to 2 gigabytes for all data space usage, your site might have restrictions on virtual storage usage that could reduce the effectiveness of XBUF.

The XB Macro

The XB macro starts and ends a sequence of XBUF macros.

XB |START
|END

START begins generating the XBUFTBL table of control blocks.

END finishes generating the XBUFTBL table.

Note: The XB START macro must be the first macro in the sequence of macros, and the XB END macro must be the last.

The XBMEMORY Macro

The XBMEMORY macro starts the data space definition. You must specify the buffer size and the number of buffers needed for this data space. (See "The XBDD Macro" for instructions about how to specify what databases to include with this data space definition.) After ending each data space definition by using the XBMEMORY END statement, you can start another data space definition.

XBMEMORY |START,CISIZE=*cisize*, COUNT=*count*
|END

START starts a set of macros that define a data space. START is a positional parameter.

END ends a set of macros that define a data space. END is a positional parameter.

cisize is the buffer size that composes the data space. The control-interval size for the databases specified by the XBDD macros must equal this value. The value of CISIZE must be an acceptable SYSTEM 2000 CISIZE.

count is the number of buffers that compose the data space.

Note: You must specify the CISIZE and the COUNT in the XBMEMORY START macro for each data space that you define.

The XBDD Macro

The XBDD macro specifies a database or database file that will use the data space. The XBMEMORY macro requires at least one XBDD macro.

XBDD DDNAME=*filename*

filename is the DDname of a database file. You can specify either 7or 8 characters in the DDname. For 7 characters, XBUF assumes that the characters represent a valid database name (possibly padded with X's) and sets up Files 1 through 6 by appending the numbers 1 through 6 to the specific DDname. An 8-character DDname designates one specific database file. Files 7 and 8 are not valid in the XBDD macro.

The DDname must follow the conventions for database file names. The DDname should consist of the first 7 characters of the database name and, optionally, a suffix of 1 through 6 as the 8th character. Replace embedded blanks and trailing blanks with X's. You must code at least one XBDD macro for each XBMEMORY macro. If you use more than one XBDD macro in a sequence, all files specified by the DDnames belong to the same data space, and their CISIZE must match the CISIZE specified for the data space. If a database CISIZE does not match the data space CISIZE when the file is opened, XBUF issues WTO S2K2028, and the database file is not included in the data space.

```
example  +S2K2028/01- XBUF  FOUND CISIZE 26624 FOR LIBRARY5,
         +S2K2028/01- XBUF  BUT DATAS1   CISIZE IS 04096
         +S2K2028/01- XBUF  SETTING LIBRARY5 RUNTYPE = 'NONE'
```

Examples

To define a data space, specify

- the XBMEMORY START macro.
- at least one XBDD macro, which gives the DDname of a database file that will make up the data space. If you want more than one database file in the data space, specify consecutive XBDD macros between the XBMEMORY START and XBMEMORY END macros. You can include Files 1 through 6 for any database.
- the XBMEMORY END macro after the last XBDD macro.

Example 1 This example illustrates using the XBMEMORY and XBDD macros to set up one data space of 1000 buffers for two different databases that contain Files 1 through 6 of the EMPLOYEE database and File 2 of the PERSONNEL database (these files must have a CISIZE of 4096).

```
XB  START
XBMEMORY START,CISIZE=4096,COUNT=1000
XBDD  DDNAME=EMPLOYE
XBDD  DDNAME=PERSONN2
XBMEMORY END
XB  END
```

Example 2 This example illustrates using the XBMEMORY and XBDD macros to set up two data spaces. Data space 1 has 1000 buffers of 22528 bytes each and contains only File 6 of the EMPLOYEE database (File 6 must have a CISIZE equal to 22528). Data space 2 has 500 buffers of 26624 bytes each and contains Files 1through 6 of the LIBRARY database (these files must have a CISIZE equal to 26624).

```
XB  START
XBMEMORY START,CISIZE=22528,COUNT=1000
XBDD  DDNAME=EMPLOYE6
XBMEMORY END
XBMEMORY START,CISIZE=26624,COUNT=500
XBDD  DDNAME=LIBRARY
 XBMEMORY END
 XB  END
```

Note: Storage for each data space can be calculated by multiplying CISIZE × COUNT. SYSTEM 2000 acquires additional storage for control blocks; approximately, 12 × COUNT bytes per data space.

Assembling and Linking Procedures

After the XBUF macros are coded, the resulting table must be assembled and then linked into the SYSTEM 2000 run-time load library. Sample JCL is provided in your CNTL library as member JCLXBUF. The resulting load module name must be of the format XBUFTBL[*suffix*]. The optional suffix distinguishes one XBUF configuration from another when you generate several XBUF load modules. During initialization, SYSTEM 2000 will attempt to load XBUFTBL[*suffix*] which must exist in order to use the XBUF data space feature.

SYSTEM 2000 Execution Parameters

The two execution parameters that control XBUF processing are: XBUF and XBUFSUF. The XBUF parameter determines whether XBUF is active. XBUF turns the XBUF software on or off. The XBUFSUF parameter specifies a suffix that indicates which XBUFTBL load module you want to use.

XBUF= YES|NO

YES means that XBUF is active and data spaces will be created for database files specified by your XBUF macros. NO means that XBUF is not activated. The default is NO.

If XBUF = YES, SYSTEM 2000 looks for member XBUFTBL[*suffix*] in its run-time load libraries, which are allocated by using the STEPLIB or JOBLIB DD statements. The delivery tape contains a sample XBUFTBL module, which creates data spaces for the EMPLOYEE and LIBRARY databases that are created for validation purposes at installation.

If you have created more than one XBUFTBL member, you must indicate which configuration you want to use by specifying the XBUFSUF execution parameter. SYSTEM 2000 loads the members, and the XBUF software formats the data spaces. The XBUFSUF execution parameter allows you to activate a specific configuration of XBUF data spaces.

XBUFSUF=*suffix*

s*uffix* is a unique character appended to the XBUFTBL load module name that indicates the configuration to activate. s*uffix* can be any valid national character that is allowed in a PDS member name. The member name for the XBUFTBL module must be in the format XBUFTBL[*suffix*]. Optionally, you can create several configurations of XBUFTBL, each represented by a different module in the load library. In this instance, you must specify the SYSTEM 2000 execution parameter XBUFSUF.

XBUF Console Command Syntax

Here is the general format for XBUF console commands. The asterisk (*), which may be used instead of SEND XBUF, provides a fast, abbreviated format.

```
|SEND XBUF  D
|*          D
```

If you submit XBUF commands from the master console, you must precede each command with one of the following:

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

jobname specifies the job name of the Multi-User system. Alternate consoles do not need these identifiers. For example,

```
MODIFY $MU, SEND XBUF D EMPLOYE* *      master console
|F $MU, *D EMPLOYE* *                   master console
SEND XBUF D EMPLOYE* *                  alternate console
*D EMPLOYE *  *                         alternate console
```

Displaying XBUF I/O Information

I/O information can be displayed for data spaces and the databases that use them. Data spaces can only be displayed by using the asterisk (*) because data space names are internally generated. The RATES parm (which provides the rates of reads, writes, and drops-of-buffers per minute) and the PAGES parm (which provides the buffer or page counts) help you determine the effectiveness of XBUF. The syntax for XBUF display commands in a Multi-User environment is issued in this order: command, filename, parm. For example,

Command	*file name*	*parm*
|SEND XBUF D	|*	|*
|* D	|DDname	|RATES
		|PAGES

A DDname that has an asterisk (*) as a suffix means display all files that begin with the specified characters. Using an asterisk instead of a DDname means display all files and data spaces. Using an asterisk instead of RATES or PAGES produces both reports.

I/O rates vary during normal workloads, and XBUF intentionally keeps only approximations. In effect, each rate is rounded to a precision of one event per minute. Rounding minimizes the CPU time and the main memory overhead for monitoring.

Example

SEND XBUF D * *

```
S2K2001/01- XBUF RATES: DATAS0 ① RPM=0010 ②    WPM=***** ③    DPM=***** ④
S2K2001/01- XBUF PAGES: DATAS0    CI=04096 ⑤   TOTAL=001000 ⑥  BU=000040 ⑦
S2K2001/01- XBUF
S2K2001/01- XBUF RATES: EMPLOYE1 DBRPM=*****⑧   DBWPM=***** ⑨
S2K2001/01- XBUF                 XBRPM=0006 ⑩   XBWPM=***** ⑪  XBDPM=***** ⑫
S2K2001/01- XBUF PAGES: EMPLOYE1 TOTAL=00050 ⑬  PC=00011 ⑭     STATUS=O ⑮
S2K2001/01- XBUF
S2K2001/01- XBUF RATES: EMPLOYE2 DBRPM=*****   DBWPM=*****
S2K2001/01- XBUF                 XBRPM=0004    XBWPM=***** XBDPM=*****
S2K2001/01- XBUF PAGES: EMPLOYE2 TOTAL=00050   PC=00029      STATUS=O
```

```
S2K2001/01- XBUF RATES: 1   RPM= 2     WPM= 3    DPM= 4
S2K2001/01- XBUF PAGES: 1   CI=  5     TOTAL=6   BU= 7
S2K2001/01- XBUF
S2K2001/01- XBUF RATES: 1   DBRPM= 8   DBWPM= 9
S2K2001/01- XBUF            XBRPM= 10  XBWPM= 11 XBDPM= 12
S2K2001/01- XBUF PAGES: 1   TOTAL= 13  PC= 14    STATUS= 15
```

① is the database file name or data space name. (Data spaces begin with DATAS as the prefix)
② is the reads-per-minute for the data space.
③ is the writes-per-minute for the data space.
④ is the drops-per-minute for the data space. (The oldest buffer is dropped to make room.)
⑤ is the size of the buffers that compose the data space. The CISIZE is equal to the control interval size of the databases whose data will reside in the data space.
⑥ is the number of buffers allocated to this data space.
⑦ is the number of buffers currently in use for this data space.
⑧ is the database file reads-per-minute.
⑨ is the database writes-per-minute.
⑩ is the reads-per-minute issued against data space on behalf of this file.
⑪ is the writes-per-minute issued against the data space on behalf of this file.
⑫ is the database pages dropped from data space.
⑬ is the number of pages in the database file.
⑭ is the number of database pages that currently reside in a data space.
⑮ is the file status. O means the file is open; C means the file is closed.

Chapter 14: Execution Parameters

Overview

Single-user and Multi-User execution parameters are set for an entire session. SYSTEM 2000 obtains execution parameters from one of the following sources, listed in order of priority, from highest to lowest.

1. Operator replies – In a Multi-User environment, the console operator can change parameter values if the OPI parameter equals YES. Console operator values take precedence over any previous values.

2. EXEC statement in the job – parameters in the EXEC statement override S2KPARMS data set values.

3. S2KPARMS data set – If S2KPARMS is not present or it fails to open, default values are used for all parameters that are not set in the EXEC statement and not overridden by the operator.

4. SYSTEM 2000 default value – These values have the lowest priority.

Execution Parameters

Table 14.1 contains a summary of the SYSTEM 2000 execution parameters in alphabetical order. The syntax and details about each parameter are presented (also in alphabetical order) following the table.

Table 14.1 SYSTEM 2000 Execution Parameters

Name	Values	Default
ALLOC=*option*	YES, NO, TBL	YES
ACCT=*option*	YES, NO	NO
COPYAREA*n*=*size*K/*m*	*n*=1 to 6 *size*=1K to 65K *m*=1 to 4096	*n*=1 *size*=1K *m*=4
CORECOV=*option*	YES, NO	NO
DBBUFN=*n*	*n*=1 to 255	*n*=5
EXITS=*option*	YES, NO	NO
EXPSAVE=*yyyyddd*	*yyyyddd*-valid year and day	NONE
FREE=*option*	YES, NO	NO

Table 14.1 SYSTEM 2000 Execution Parameters (*continued*)

Name	Values	Default
FPTPSYS=*option*	YES, NO	YES
KEEPUNT=*unit*	*unit*=disk unit	SYSDA
LDBS=*blocks*	*blocks*=0 to 32	0
LDBSIZE=*n*	*n*=0 to 10000	1000
LHOLD=*option*	YES, NO	YES
LIST=*option*	YES, NO	YES
LOGCOUNT=*n*	*n*=1 to 32767 (in thousands)	1 (=1000)
LOGLEVEL=*option*[/*option*]…	LOPEN, PDATE, POPEN, TSPIO, TSPND, TSTRT, UINIT, USEGM, USPND, NO	NO
NLSEG=*hsec/nio*[/DELAY]	*hsec*=NO, 0 to 999999 *nio*=NO, 0 to 999999	NO NO
OPI=*option*	YES, NO	NO
OPT000=*option*	YES, NO	NO
OPT001=*option*	YES, NO	NO
OPT002=*option*	YES, NO	NO
OPT003=*option*	YES, NO	NO
OPT004=*option*	YES, NO	NO
OPT005=*option*	YES, NO	NO
OPT006=*option*	YES, NO	NO
OPT007=*option*	YES, NO	NO
OPT008=*option*	YES, NO	NO
OPT009=*option*	YES, NO	NO
OPT010=*n*	*n*=0 to 32767	0

Table 14.1 SYSTEM 2000 Execution Parameters (*continued*)

Name	Values	Default
OPT011=*n*	*n* = 0 to 4095	0
OPT012=*n*	*n* = 4096 to 26624	4096
OPT013=*n*	*n* = 0 to 5	0
OPT014 through OPT029 are reserved for future use		
OPT030=*option*	YES, NO	NO
OPT031=*option*	YES, NO	NO
OPT032=*option*	YES, NO	NO
OPT033=*option*	YES, NO	NO
OPT034=*option*	YES, NO	NO
OPT035=*option*	YES, NO, ALL	NO
OPT036 is reserved for future use		NO
OPT037=*option*	YES, NO	NO
OPT038 is reserved for future use		
OPT039=*option*	YES, NO, OFF	OFF
OPT040=*option*	YES, NO, OFF	OFF
OPT041 is reserved for future use		
OPT042=*option*	YES, NO	NO
OPT043=*option*	YES, NO	NO
OPT044=*option*	YES, NO	NO
OPT045=*option*	YES, NO	NO
OPT046=*option*	YES, NO	NO
OPT047=*option*	YES, NO	NO
OPT048=*option*	YES, NO	NO
OPT049=*option*	YES, NO	NO

Table 14.1 SYSTEM 2000 Execution Parameters (*continued*)

Name	Values	Default
OPT050 is reserved for future use		
OPT051=*option*	YES, NO	NO
OPT052=*option*	YES, NO	NO
OPT053=*option*	YES, NO	NO
OPT054=*option*	YES, NO	NO
OPT055=*option*	YES, NO	NO
OPT056 through OPT999 are reserved for future use		
PAD*nn*=*x*/*y* |YES |NO	*nn*=00 to 15 *x*=allocation units *y*=blocks per unit	See details in text
PADPRI=*n*	*n*=1 to 32767	1
PADSEC=*n*	*n*=0 to 32767	1
PADSPACE=*option*	CYL, TRK	CYL
PADVOL=*volume*	See details in text	NONE
PLSEG=*hsec*/*nio*[/DELAY]	*hsec*=NO, 0 to 999999 *nio*=NO, 0 to 999999	NO NO
POOL*n*=*bufs*/*nbufs*[/usage]	See details in text	See details in text
PQA=FIFO or PQA=PRTY[/*n*/*y*]	*n*=0 to 999 *y*=0 to 225	FIFO 0/0
PREFIX=*prefix*	See details in text	See details in text
RETPRD-*nnnn*	*nnnn*=number of days	NONE
RW=*option*	YES, NO	NO
SAME=*option*	YES, NO	YES
SAVEUNT=*unit*	*unit*=disk unit	SYSDA
SDBS=*blocks*	*blocks*=0 to 32	1
SDBSIZE=*n*	*n*=1 to 10000	430

Table 14.1 SYSTEM 2000 Execution Parameters (*continued*)

Name	Values	Default
SFPRI=*n*	*n*=1 to 32767	1
SFSEC=*n*	*n*=0 to 32767	1
SFSPACE=*option*	CYL, TRK	CYL
SFUNIT=*unit*	See details in text	SYSDA
SID=*n*	*n*=0 to 99	1
STAE=*option*	YES, NO, NODUMP	YES
STARTTP=*option*	YES, NO	YES
STAX=*option*	YES, NO	YES
SYSOUT=*x*	*x*=1 character	A
S2KMSGL=*nnn*	*nnn*=44 to 256	80
THREADS=*n*	*n*=1 to 63	1
TPSCRUN=*n*	*n*=1 to 10000	180
TPSEG=*hsec/nio*[/DELAY]	*hsec*=NO, 0 to 999999 *nio*=NO, 0 to 999999	NO NO
TPTHREADS=*n*	*n*=0 to 230	5
TPVOL=*volume*	See details in text	NONE
USERS=*n*	*n*=0 to 230	16
XBUF=*option*	YES, NO	NO
XBUFSUF=*suffix*	*suffix*=1 character	NONE
YEARCUTOFF=*nnnn*	*nnnn*=year 1500 to 9900	1920

ALLOC=*option*

allows dynamic allocation of database files

Syntax

ALLOC=YES| NO| TBL

Details

The ALLOC parameter controls dynamic allocation of database files and varying databases offline and online. The default is YES, which means that dynamic allocation or de-allocation of new or existing database files is allowed. Users can issue the CONTROL language commands, ALLOC and FREE, or they can dynamically access databases specified in the S2KDBCNT table. SYSTEM 2000 automatically searches for existing database files if there is no ALLOC command or S2KDBCNT table. ALLOC=YES also allows the console operator to vary databases offline and online in a Multi-User environment.

ALLOC=NO means that dynamic allocation or de-allocation of database files is not allowed. The CONTROL language commands ALLOC and FREE are not honored, and the S2KDBCNT table is not searched for database file allocations. The database files must be allocated in the start-up job stream or allocated with user exits.

ALLOC=TBL restricts dynamic allocation or de-allocation to those databases that are defined in the S2KDBCNT table. TBL is useful for a Multi-User environment in which you want to have strict control of database access. Users cannot issue the CONTROL language command ALLOC to create new databases dynamically or to override the information in the S2KDBCNT table.

ACCT=*option*

activates or de-activates the Accounting Log

Syntax

ACCT = YES|NO

Details

The ACCT parameter controls the writing of accounting data to the Accounting Log. The ACCT parameter cannot be changed from the console. The default is NO, which means that writing accounting data to the Accounting Log is disabled.

NOTE: See the **NLSEG**, **TPSEG** and **PLSEG** parameters.

ACCT=YES generates accounting data, Multi-User initialization records, User Termination records, and Multi-User Termination records.

COPYAREA*n*=*size*K/*m*

Assigns an interregional copy area for systems requiring the S2KCOPY routine in the Multi-User SVC

Syntax

COPYAREA*n*= *size***K/***m*

Details

The COPYAREA*n* parameter specifies a common area (or inter-region copy area) for data transfer between Multi-User and a dependent region user. A 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. The default is COPYAREA1=1K/4.

n is the inter-region copy area number. The range is 1 to 6. If you specify multiple COPYAREA*n* parameters, the numbers must be consecutive and start with 1. The default for *n* is 1. *size* is the size of each area in K notation (for example, 1K or 2K, where 1K=1024 bytes). The range is 1K to 65K. Each larger value of *n* requires a larger value of *size*. The default is 1K. *m* is the number of areas of this size in this copy area. The range is 1 to 4096. The default for *m* is 4.

Each SCF user (TP or batch) requires a 1K copy area. You should pre-allocate enough 1K copy areas for all concurrent SCF users. PLEX programs require a variable size based on the specific PLEX command and the size and number of parameters in the call to SYSTEM 2000; for example, S2KDUM, COMMBLOCK, SUBSCHEMA RECORD, where-clause, and other parameters. PLEX commands do not execute in a 1K copy area. Multi-User dynamically acquires a larger copy area so the PLEX user can continue processing if the established copy areas are insufficient. When this occurs, message 0017 is written to the Diagnostic Log. If this message occurs often, you

should increase the current copy areas in either size or number or both. If Multi-User cannot dynamically obtain more space, the user who is involved is terminated with a user abend.

To obtain the copy area size 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) × 124 bytes
- SUBSCHEMA RECORD
  - SUBSCHEMA Control Block (SCB): 56 bytes
  - Item Control Blocks (ECBs): SCBNUM × 56 bytes; SCBNUM is the number of items in the subschema record, the sixth field in the SCB.
  - length of the subschema record I/O area in bytes. This length is in SCBBUF, the third field in the SCB.
- Where-clause list
  - parameter list:
    (number of items and WORKING-STORAGE values in the where-clause + 1) × 4 bytes
  - Operator Control Block (OCB): (number of operators + 1) × 4 bytes
  - sum of the picture sizes of all items and the WORKING-STORAGE values specified in the where-clause. Each value is rounded to a word boundary.
  - number of items in the where-clause × 56 bytes
  - number of AT operators × 4 bytes

- ORDER BY list
  - parameter list: (number of ordering items + 1) × 4 bytes
  - Operator Control Block (OCB): same as the length of the parameter list.

CORECOV=*option*

determines whether only the primary or all databases are recovered by Coordinated Recovery

Syntax

CORECOV=YES|NO

Details

The CORECOV parameter determines the scope of Coordinated Recovery in single-user and Multi-User environments. CORECOV=YES causes SYSTEM 2000 to attempt a full Coordinated Recovery in the primary and secondary databases. Recovery includes all user transactions in all databases in the logical units of work. 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 default is NO, which means that SYSTEM 2000 recovers only the primary database and includes committed updates from all users of the database. SYSTEM 2000 does not check availability of secondary databases or their Update Logs.

DBBUFN=*n*

alters the number of QSAM buffers for database files during SAVE and RESTORE processing

Syntax

DBBUFN=*n*

Details

The DBBUFN parameter specifies the number of VSAM buffers to be used for database files while saving and restoring databases. The VSAM buffers are acquired for the duration of the SAVE or RESTORE process and then released. To specify the QSAM buffers for the Savefile, use the BUFNO option in the DD statement. The range is 1 to 255. The default is 5.

DITTO=*option*

controls the use of the DITTO command across SCF TP segments

Syntax

DITTO=YES|NO

Details

The DITTO parameter allows the use of the DITTO command in a Multi-User environment. The DITTO command repeats the action clause of the last SCF command. DITTO=YES means that SCF users can use the DITTO command to refer to the last action clause specified in the previous segment. The DITTO command can be used without $LOCK, which ties up a thread. The default is YES.

NOTE: DITTO is always available in single-user sessions, however, it is a parameter-controlled command under Multi-User.

EXITS=*option*

controls the overall execution of all user exits

Syntax

EXITS=YES|NO

Details

The EXITS parameter controls the overall execution of all user exits. If the S2EXIT interface routine has been link-edited with SYSTEM 2000, the EXITS parameter is ignored and the user exits capability is enabled. EXITS=YES enables user exits. The default is NO, which means that user exits cannot be used unless the S2EXIT routine has been linked-edited with SYSTEM 2000.

EXPSAVE=*yyyyddd*

specifies an expiration date for your Savefile or Keepfile

Syntax

EXPSAVE=*yyyyddd*

Details

The EXPSAVE parameter allows you to specify an expiration date for your dynamically allocated Savefile or Keepfile. *yyyy* is a 4-digit year value that ranges from 1500 through 9999. *ddd* is a 3-digit day value that ranges from 1 through 366. When this parameter is not specified, no expiration date is specified when the file is allocated. There is no default value.

FREE=*option*

controls the de-allocation of database files

Syntax

FREE=YES|NO

Details

The FREE parameter controls whether SYSTEM 2000 de-allocates database files when it terminates. FREE=YES means that SYSTEM 2000 de-allocates database files that were dynamically allocated. The default is NO, which means that de-allocation will not take place at exit time.

FPTPSYS=*option*

determines whether system areas are obtained for fetch-protected systems to allow TP access from different address spaces

Syntax

FPTPSYS=YES|NO

Details

The FPTPSYS parameter determines if system areas are obtained when Multi-User is initialized. This allows TP access from different OS/390 address spaces. The FPTPSYS parameter should be YES if the host operating system is fetch-protected, such as OS/390. The FPTPSYS parameter is directly related to the SYSTEM 2000 Type 2 SVC macro parameter OS=MVS. When FPTPSYS=NO, the TP system must be executed in the same address space as Multi-User. The default is FPTPSYS=YES, which means that system areas are obtained when Multi-User is initialized.

KEEPUNT=*unit*

specifies the disk unit type for a dynamically allocated Keepfile

Syntax

KEEPUNT=*xxxxxxxx*

Details

The KEEPUNT parameter specifies the type of disk unit to request when dynamically allocating a Keepfile. *xxxxxxxx* is a maximum of 8 characters of disk type (3380, 3390, DISK). If this parameter is omitted, the disk type defaults to SYSDA.

LDBS=*blocks*

specifies the initial number of Database Definition Blocks to use for large databases

Syntax

LDBS=*blocks*

Details

The LDBS parameter specifies the number of large databases that SYSTEM 2000 can have open at one time without having to acquire additional definition blocks. The range is 0 to 32. The default is 0.

LDBSIZE=*n*

specifies the maximum number of Database Definition Blocks for large databases

Syntax

LDBSIZE=*n*

Details

The LDBSIZE parameter specifies the maximum number of items that a large database can contain. The range is 0 to 10,000. The default is 1000.

LHOLD=*option*

controls the use of the PLEX option HOLD

Syntax

LHOLD=YES|NO

Details

The LHOLD parameter determines if PLEX programs that run 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. LHOLD=NO means that a local hold is not required prior to a PLEX update in the Multi-User session. The default is YES, however, LHOLD=YES is ignored if the database has the Rollback Log enabled, has a global hold, or is under exclusive use.

LIST=*option*

controls the display of the execution parameter settings

Syntax

LIST=YES|NO

Details

The LIST parameter enables or disables a display of SYSTEM 2000 execution parameter settings. Only the first 21 characters of each parameter setting are displayed. The listing is written to both the job log and the operator console. The first character in each display line indicates whether the value is a default or a specified value. The following list contains the meaning of each special character:

♭	a universal default. (The indicator is a blank.)
@	the default is specific to your site.
*	the value was changed by the S2KPARMS file.
=	the value was change by the EXEC statement.
-	the value was changed by the operator (OPI=YES).

LIST=NO produces an abbreviated list that shows only numeric parameters where the default is non-zero. This is ignored if the software finds an unrecognizable parameter, and processing continues as though LIST=YES. The default is YES, which produces a listing of parameter values specified in the S2KPARMS file and of default values that apply to the session.

LOGCOUNT=*n*

specifies the maximum number of records that can be written to the Diagnostic Log

Syntax

LOGCOUNT=*n*

Details

The LOGCOUNT parameter sets the maximum number of records (event messages) that can be written to the Diagnostic Log (S2KDIAG data set). This parameter applies only when the Diagnostic Log is written to disk. If the maximum for LOGCOUNT is reached, the file is re-set to the beginning of the disk file. *n* is an integer that specifies the maximum number of messages in units of 1000 that can be logged . The LOGCOUNT parameter cannot be dynamically modified after Multi-User has been initialized. The range for *n* is 1 to 32767. The default is 1, which means a maximum of 1,000 messages.

LOGLEVEL=*option*[/*option*]...

specifies what types of messages are written to the Diagnostic Log

Syntax

LOGLEVEL=NO|LOPEN|PDATE|POPEN|SDATE|TSPIO|TSPND|TSTRT|UINIT|USEGM|USPND

Details

The LOGLEVEL parameter determines which types of event data are written to the Diagnostic Log (S2KDIAG data set). Specifying one or more LOGLEVEL variables activates the Diagnostic Log. When you specify multiple LOGLEVEL variables, use a slash (/) to separate the variables. The LOGLEVEL variables can also be modified dynamically during a Multi-User session if you do not set the LOGLEVEL parameter to NO. If NO is specified, you cannot alter the LOGLEVEL parameter. The default is NO.

Here is a list of variables and their functions:

LOPEN
: enables a logical open or close of a database.

PDATE
: is the date and time in packed format. Time, only, is the default if PDATE is not specified.

POPEN
: enables a physical open or close of a database.

SDATE
: is the date and time in SAS TODSTAMP informat. SDATE is faster than the SVC-based time stamp.

TSPIO
: enables thread suspended or dispatched I/O.

TSPND
: enables thread suspended or dispatched (except for I/O).

TSTRT
: enables thread start or stop.

UINIT
: enables Multi-User initialization and USER start or stop. UINIT is the default if the LOGLEVEL parameter is not used.

USEGM
enables command segment start or stop. USEGM is for listings on the printer or with LOGDUMP. It is not used by DIAG2000.

USPND
enables user suspension or acquit.

NOTE:

- Embedded blanks are not allowed.
- All LOGLEVEL variables must be specified in 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. PDATE or SDATE determines the format of the time-stamp.
- PDATE and SDATE are mutually exclusive. If both are specified, WTO message S2K0111/00 is issued.

NLSEG=*hsec/nio*[/DELAY]

controls which SCF batch job data, if any, is written to the Accounting Log

Syntax

NLSEG=NO|CPU-*time*/NO|*I/O-count*[/DELAY]

Details

The NLSEG parameter allows you to control how much accounting data is generated for Multi-User segments. The default is NO, NO.

CPU-time is the threshold time in 100^{ths} of a second, specified as an integer. The range is 0 to 999999. Multi-User produces accounting data for segments that meet the *I/O* criterion and whose *CPU-time* equals or exceeds the specified time. If NO is specified, the CPU-*time* is not used as a criterion.

I/O-count is the threshold number of database I/O's specified as an integer. The range is 0 to 999999. Multi-User produces accounting data for segments that meet the *CPU-time* criterion and whose *I/O-count* equals or exceeds the specified number. If NO is specified, the *I/O-count* is not used as a criterion.

DELAY postpones Multi-User segment accounting 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. The syntax for that command is

MODIFY *jobname,*	**START**	**STAT**	**NL**|**ALL**
	STOP	**STAT**	**NL**|**ALL**

jobname refers to the SYSTEM 2000 job name. STOP tells Multi-User to stop writing accounting data in the Accounting Log file. STOP applies if the writing started because DELAY was not specified or because of a previous START command. NL controls writing of accounting data for batch SCF segments, and ALL means NL, PL, and TP.

OPI=*option*

determines whether the console operator can override or specify execution parameter settings

Syntax

OPI=YES|NO

Details

The OPI parameter determines whether the software will allow the console operator to select execution parameters during the Multi-User initialization step. OPI=YES means that a message that requests parameters or a U appears on the operator console. When all the necessary parameters have been entered, the operator must enter a U, which tells Multi-User to finalize initialization. OPI=YES is the highest priority level for overriding execution parameter settings. OPI =NO means that the console operator is not allowed to enter execution parameters. The default is NO.

OPT000=*option*

checks the system separator when unloading CHAR, TEXT, and UNDEFINED values

Syntax

OPT000=YES|NO|OFF

Details

The OPT000 parameter determines how extensively SYSTEM 2000 checks for the system separator when unloading CHARACTER, TEXT, or UNDEFINED item values. OPT000=OFF tells the software not to perform any checks for the system separator character in unloaded data. OFF saves time during the unload process because 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. OPT000=YES tells the software to check whether any characters that are allowed as system separators occur in any of the CHARACTER, TEXT, or UNDEFINED item values being unloaded. The unload process uses more time when OPT000=YES. The default is NO, which means that SYSTEM 2000 checks for the current system separator only. If the HEX ON format option is in effect, the check for the system separator is not performed for UNDEFINED items because those values are in HEX notation in the output.

Frequently, UNDEFINED values might 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 that was used when the data was loaded, or because the data was loaded by using 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. OPT000=YES produces message -291- after message -290-. The characters displayed in message -291- did not occur in any unloaded value, therefore, those characters 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.

OPT001=*option*

determines whether SCF commands are echoed

Syntax

OPT001=YES|NO

Details

The OPT001 parameter sets the default for echoing SCF commands. OPT001=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. The default is NO, which means ECHO ON.

OPT002=*option*

determines whether you can save a damaged database

Syntax

OPT002=YES|NO

Details

The OPT002 parameter prevents users from saving damaged databases. OPT002=YES causes an 0C1 to occur if an attempt is made to save a damaged database. The default is NO, which means that the database will be saved regardless of its condition.

OPT003=*option*

disables or enables clearing for QUEUE sessions

Syntax

OPT003=YES|NO

Details

The OPT003 parameter disables clears in a QUEUE or a TERMINATE session. OPT003=YES disables clears for a QUEUE or a TERMINATE session in order to increase performance. The default is NO, which means that clears are enabled. (OPT003 replaces Special Zap 34.)

OPT004=*option*

includes system-related time in Type 5, 6, and 8 Accounting Log records

Syntax

OPT004=YES|NO

Details

The OPT004 execution parameter includes system-related time in Types 5, 6, and 8 records. OPT004=YES includes SRB (system-related) time and PRB (problem-related) time in Accounting Log Types 5, 6, and 8 records. The default is NO, which means that only PRB time is specified in the Accounting Log Types 5, 6, and 8 records.

OPT005=*option*

sets the default for the ZERO or the ZERO SUPPRESS format option

Syntax

OPT005=YES|NO

Details

The OPT005 parameter sets the default for the ZERO or the ZERO SUPPRESS format option, which is used in SCF commands. OPT005=YES specifies the ZERO format option. The default is NO, which means ZERO SUPPRESS. (OPT005 replaces Special Zap 103.)

OPT006=*option*

sets the default for the NULL or the NULL SUPPRESS format option

Syntax

OPT006=YES|NO

Details

The OPT006 parameter sets the default for the NULL or the NULL SUPPRESS format option, which is used in SCF commands. OPT006=YES specifies NULL. The default is NO, which sets NULL SUPPRESS. (OPT006 replaces Special Zap 109.)

OPT007=*option*

sets the default for the REPEAT or the REPEAT SUPPRESS format option

Syntax

OPT007=YES|NO

Details

The OPT007 parameter sets the default for the REPEAT or the REPEAT SUPPRESS format option, which is used in SCF commands. OPT007=YES specifies REPEAT SUPPRESS. The default is NO, which means REPEAT. (OPT007 replaces Special Zap 110.)

OPT008=*option*

sets the default for the NAME or the NUMBER format option

Syntax

OPT008=YES|NO

Details

The OPT008 parameter sets the default for the NAME or the NUMBER format option, which is used in SCF commands. OPT008=YES specifies NAME. The default is NO, which means NUMBER. (OPT008 replaces Special Zap 111.)

OPT009=*option*

determines whether database Files 2 through 6 are cleared for RELOAD and RELEASE

Syntax

OPT009=YES|NO

Details

The OPT009 parameter allows clearing of all database files for a RELOAD or a RELEASE command. OPT009=YES means that database Files 2 through 6 (in addition to File 1) are cleared when the database is reloaded or released. The default is NO, which means that only database File 1 is cleared for RELOAD or RELEASE. (OPT009 replaces Special Zap 121.)

OPT010=*n*

sets a limit on selected records for updates and retrievals

Syntax

OPT010=*nnnnn*

Details

The OPT010 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. The range is 0 to 32767. For example, if you specify 1200, an error message appears when the number of selected records for any user retrieval or update exceeds 1200. If you do not specify a number for OPT010, the default is 0, which means no limits are set. (OPT010 replaces Special Zap 3.)

OPT011=*n*

specifies the maximum number of pages to be written to the primary scratch file for where-clause processing

Syntax

OPT011=*nnnn*

Details

The OPT011 parameter indicates the maximum number of pages written to the primary scratch file for where-clause processing. The range is 0 through 4095. The default is 0, which means no limit. (OPT011 replaces Special Zap 20.)

OPT012=*n*

sets the default block size for database files

Syntax

OPT012=*nnnnn*

Details

The OPT012 parameter specifies the default CISIZE for dynamically allocated new database files. The value for OPT012 must be a valid CISIZE. If this value is greater than the largest CISIZE specified for any pool, error messages will appear. The minimum value of *nnnnn* is 4096, and the maximum value is 26624. The default is 4096. (OPT012 replaces Special Zap 50.)

OPT013=*n*

changes the default date format. A default of 0 means *mm/dd/yyyy*

Syntax

OPT013=*n*

Details

The OPT013 parameter changes the default date format. The range is 0 to 5. The default is 0. OPT013 replaces Special Zap 4, which sets the following date formats:

0	*mm/dd/yyyy*
1	*mm/dd/yy*
2	*dd/mm/yyyy*
3	*dd/mm/yy*
4	*yyyy/mm/dd*
5	*yy/mm/dd*

OPT014 through OPT029 are reserved for future use

OPT030=*option*

disables clearing of PLEX stacks and Locate Files when an LUW is committed

Syntax

OPT030=YES|NO

Details

The OPT030 parameter disables clearing of PLEX stacks and Locate Files. OPT030=YES disables clearing of all PLEX stacks and eviction of all Locate Files. These processes are usually performed when a logical unit of work (LUW) is committed. The default is NO, which means that the PLEX stacks and Locate Files are cleared. (OPT030 replaces Special Zap 130.)

OPT031=*option*

places the number of user calls in Type 4 Accounting Log records

Syntax

OPT031=YES|NO

Details

The OPT031 execution parameter includes the number of user calls in Type 4 records. OPT031=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. The default is NO, which means that the user termination records (Type 4) are not altered .

OPT032=*option*

sets the TALLY command option to EACH or ALL

Syntax

OPT032=YES|NO

Details

The OPT032 parameter changes the EACH or the ALL option for the TALLY command. OPT032=YES specifies ALL. The default is NO, which means EACH.

OPT033=*option*

sets the mode for clearing updated pages

Syntax

OPT033=YES|NO

Details

The OPT033 parameter alters the default mode for clearing updated pages at your site. OPT033=YES sets the mode to CLEAR AUTOMATICALLY, which clears the modified pages automatically for each update command. The default is NO, which means that modified pages are cleared only when the system needs a buffer or when the user issues the CLEAR command. (OPT033 replaces Special Zap 185.)

OPT034=*option*

allows QueX software to update a record without requiring a HOLD

Syntax

OPT034=YES|NO

Details

The OPT034 parameter determines if QueX users are required to get a local hold before updating a record. OPT034=YES means that QueX users can update a record without a local hold. The default is NO, which means that a local hold is required.

OPT035=*option*

determines the type of display from the LIST execution parameter

Syntax

OPT035=YES|NO|ALL

Details

The OPT035 parameter affects the display of SYSTEM 2000 execution parameters. OPT035 always works in conjunction with the LIST execution parameter. LIST=YES means that the value of OPT035 determines the type of listing that is displayed. The default is NO, which means that the execution parameters are not displayed, and OPT035 is ignored.

OPT035=NO and LIST=YES displays only the execution parameters that are specified in the S2KPARMS file, that is, the parameters that were modified.

OPT035=YES and LIST=YES displays all execution parameters. The listing shows the default values and the modified values.

OPT035=ALL and LIST=YES displays all execution parameters. The listing shows the execution parameter settings and includes the defaults and the modified values.

OPT036 is reserved for future use.

OPT037=*option*

places the synchpoint ID for an LUW in the ACCTALT field of the user URB

Syntax

OPT037=YES|NO

Details

The OPT037 execution parameter places the synchpoint ID in the ACCTALT field of the user URB. OPT037=YES places the synchpoint ID for a logical unit of work (LUW) in the ACCTALT field of the user URB. The default is NO, which means that the ACCTALT field of the user's URB is not affected.

OPT038 is reserved for future use

OPT039=*option*

affects floating point precision in PLEX data if padding is needed

Syntax

OPT039=YES|NO|OFF

Details

The OPT039 parameter affects how you want floating point data precision to be handled in PLEX programs if padding is needed. OPT039=YES means that you want data precision for single- and double-precision numbers. Return Code 17 is issued if the subschema definition does not match the database definition. OPT039=NO means that 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. The default is OFF, which means that you are not concerned with data precision, and Return Code 98 is not issued as a warning. Padding occurs if needed, and Return Code 00 is issued.

OPT040=*option*

affects floating-point precision in PLEX data if truncation is needed

Syntax

OPT040=YES|NO|OFF

Details

The OPT040 parameter affects how you want floating-point data precision to be handled in PLEX programs if truncation is needed. OPT040=YES means that you want data precision for single- and double-precision numbers. Return Code 17 is issued if the subschema definition does not match the database definition. OPT040=NO means that 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. The default is OFF, which means that you are not concerned with data precision, and Return Code 99 is not be issued as a warning. Truncation occurs, if needed, and Return Code 00 is issued.

OPT041 is reserved for future use

OPT042=*option*

disables the setting of condition codes for the COND parameter in the JCL EXEC statement

Syntax

OPT042=YES|NO

Details

The OPT042 parameter affects the condition code setting for non-informative error messages. OPT042=YES means that the condition code is bypassed, so it will always be 0. The default is NO, which means that each error message sets the condition code.

OPT043=*option*

disables uppercase translation

Syntax

OPT043=YES|NO

Details

The OPT043 parameter affects uppercase translation. OPT043=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. The default is NO, which enables uppercase translation. (OPT043 replaces Special Zap 188.)

OPT044=*option*

enables or disables the processor prompt feature

Syntax

OPT044=YES|NO

Details

The OPT044 parameter enables or disables the processor prompt feature. OPT044=YES means that the system displays the current processor, for example, ACCESS, CONTROL, DEFINE, or REPORT. The default is NO, which means the system prompt is three dashes.

OPT045=*option*

enables or disables the processor lookup feature

Syntax

OPT045=YES|NO

Details

The OPT045 parameter enables or disables the processor lookup feature. OPT045=YES means that if you are using the QUEST or the 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. The default is NO, which means that no lookup is done, and a syntax error occurs.

OPT046=*option*

sets the default KEY or NON-KEY status when defining new items in a definition

Syntax

OPT046=YES|NO

Details

The OPT046 parameter changes the default KEY or NON-KEY status of new items that the user is defining without specifying KEY or NON-KEY. OPT046=YES means that the default status is NON-KEY. The default is NO, which means that the default status is KEY. (OPT046 replaces Special Zap 277.)

OPT047=*option*

disables validity checking of user-specified packed decimal data

Syntax

OPT047=YES|NO

Details

The OPT047 parameter disables or enables validity checking for user-specified packed decimal data. OPT047=YES means that SYSTEM 2000 will not validate user-specified packed decimal data. The default is NO, which means that validity checking will occur. (OPT047 replaces Special Zap 164.)

OPT048=*option*

determines where dynamic allocation (SVC99) error messages are written

Syntax

OPT048=YES|NO

Details

The OPT048 parameter determines where dynamic allocation (SVC 99) error messages are written. OPT048=YES specifies that messages are written to the job log. The default is NO, which specifies that messages are written to the S2KMSG file.

NOTE: If allocation of S2KMSG fails, messages are written to the job log. Multi-User messages are written to the Multi-User log.

OPT049=*option*

controls database availability

Syntax

OPT049=YES|NO

Details

The OPT049 parameter controls database availability by using the S2KDBCNT table and the destination of SVC 99 messages. OPT049=YES specifies that database availability is to be checked. For each unavailable database, the WTO message S2K1448*/sid -database* DATABASE HAS BEEN TAKEN OFF LINE is written to the log. No SVC 99 messages are written. The default is NO, which specifies that the S2KDBCNT table is not to be checked for database availability and that SVC 99 messages are written to the S2KMSG file.

OPT50 is reserved for future use

OPT51=*option*

keeps elapsed CPU time in the Accounting Log records in 1000^{ths} of a second

Syntax

OPT051=YES|NO

Details

The OPT051 parameter causes the elapsed CPU time in the Accounting Log segment records to be kept in 1000^{ths} of a second instead of 100^{ths} of a second. OPT051=YES means that CPU times in the segment records are kept in 1000^{ths} of a second. The default is NO, which means that the CPU times in the segment records are in 100^{ths} of a second.

OPT052=*option*

displays the Diagnostic Log record operation code in text format

Syntax

OPT052=YES|NO

Details

The OPT052 parameter causes the operation code in the Diagnostic Log 300 record to be in a display text format instead of an internal format of associated numerical value. For example, the START S2K operation code in the internal format is displayed as the number 91. In display format, it is displayed as START S2K. OPT052=YES converts internal operation code to a display text value. The default is NO, which means that the operation code is displayed in its internal numerical format.

OPT053=*option*

allows synchronization of a shadow database with the primary database

Syntax

OPT053=YES|NO

Details

The OPT053 parameter allows a shadow database to be synchronized with the primary database. OPT053=YES means the shadow database can be updated by using the Keepfile from the primary database. The default is NO, which means normal processing. (Special Zap 329 is required for this parameter.)

OPT054=*option*

executes a dummy save

Syntax

OPT054=YES|NO

Details

The OPT054 parameter allows the execution of a dummy save. The dummy save bypasses the normal save process but updates File 1 with information that indicates that the save occurred but the Savefile is not created. OPT054=YES means that File 1 is updated but the Savefile is not created. SCF messages are generated that inform the user that no Savefile was created. The default is NO, which means that normal save processing is performed, File 1 is updated, and the Savefile is created.

NOTE: This parameter is intended to assist users who have external save processes. Save processing SHOULD NOT be bypassed unless a backup of the databases has been performed.

OPT055=*option* (Multi-User only)

closes databases in Multi-User when the usage count drops to 0

Syntax

OPT055=YES|NO

Details

The OPT055 parameter can only be used in Multi-User. OPT055=YES causes databases in Multi-User to be closed when the usage count drops to 0, and the files were dynamically allocated by SYSTEM 2000 and the database is not listed in S2KDBCNT. This parameter has no effect in single-user. The default is NO.

OPT056 through OPT099 are reserved for future use

PAD*nn=x /y* |YES |NO

defines one or more scratch pads or asks software to assign them

Syntax

PAD*nn*=*number-of-allocation-units / blocks-per-allocation-unit***|YES|NO

Details

The PAD*nn* parameter explicitly allocates scratch pads. The value for *nn* is the pad number. The range is 00 to 15. You must define at least one scratch pad and no more than 16 scratch pads. PAD*nn* must be defined consecutively, starting with PAD00. Each scratch pad can reside in either a temporary or a permanent data set with a DDname of S2KPAD*nn*. For compatibility, *number-of-allocation-units and blocks-per-allocation-unit* numbers are ignored, and these values are processed the same as PAD*nn*=YES. PAD*nn*=YES calculates the number of allocation units and the number of blocks per allocation unit based on the size of the allocated file. PAD*nn*=NO means that the specified scratch pad parameter and any other subsequent scratch pad parameters that have higher pad numbers are ignored. You can allocate only the scratch pads that are needed for a specific 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.

PADPRI=*n*

alters scratch pad primary space

Syntax

PADPRI=*n*

Details

The PADPRI parameter specifies the primary space when scratch pads are dynamically allocated. 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 de-allocates the file at the end of the single-user job or the Multi-User session. The range for *n* is 1 to 32767. The default is 1.

PADSEC=*n*

alters scratch pad secondary space

Syntax

PADSEC=*n*

Details

The PADSEC parameter specifies the secondary space when scratch pads are dynamically allocated. 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 de-allocates the file at the end of the single-user job or the Multi-User session. The range for *n* is 0 to 32767. The default is 1.

PADSPACE=*option*

alters S2KPAD space unit

Syntax

PADSPACE=CYL|TRK

Details

The PADSPACE parameter specifies the PADPRI and PADSEC units, which are either cylinder or tracks. The default is CYL.

PADVOL=*volume*

specifies the volume where dynamically defined S2KPAD files reside

Syntax

PADVOL=*xxxxxx*

Details

The PADVOL parameter value *xxxxxx* specifies the volume where S2KPAD files reside when they are dynamically defined. This parameter is required for dynamic S2KPAD file allocations. There is no default value.

PLSEG=*hsec*/*nio*[/DELAY]

controls which PLEX job data, if any, is written to the Accounting Log

Syntax

PLSEG=NO|*CPU-time*/NO|*I/O-count*[/DELAY]

Details

The PLSEG parameter controls how much accounting data is generated for Multi-User segments. *CPU-time* is the threshold time in 100^{ths} of a second, specified as an integer. The range is 0 to 999999. Multi-User produces accounting data for segments that meet the *I/O* criterion and have *CPU-time* that equals or exceeds the specified time.

I/O-count is the threshold number of database I/O's specified as an integer. The range is 0 to 999999. Multi-User produces accounting data for segments that meet the *CPU-time* criterion and have an *I/O-count* that equals or exceeds the specified number.

DELAYpostpones Multi-User segment accounting until a START request is received from the console. If DELAY is not specified, Multi-User segment accounting starts immediately. NO means that the *CPU-time* or the *I/O-count* is not used as a criterion. The default is NO, NO.

You can also issue the following operator console command to control segment data generation:

|MODIFY *jobname*, |START STAT PL|ALL
|F |STOP STAT PL|ALL

where *jobname* refers to the SYSTEM 2000 job name. STOP tells Multi-User to stop writing accounting data into the Accounting Log file. STOP applies if the writing started because DELAY was not specified or because of a previous START. PL controls the writing of accounting data for PLEX Multi-User segments and ALL means NL, PL, and TP.

POOL*n=bufs/nbufs*[/usage]

specifies the buffer size, number of buffers, and use of the buffer pools for database files and for scratch files

Syntax

POOL*n*=*buffer-size/buffers***[/*usage*]

Details

The POOL*n* parameter specifies the attributes of each I/O buffer pool that you need. *n* is a number that designates the pool. The range is 0 to 7 (for example, POOL0, POOL1, and so on). If you specify more than one POOL*n* parameter, the pool numbers must start with 0 and be consecutive. *buffer-size* specifies the number of bytes in each buffer within a pool. The range is 4096 to 26624. The buffer size for POOL*n* must be less than or equal to the buffer size for POOL*n*+1, and so on. *buffers* is the number of I/O buffers in the specified pool. The range is 0 to 999.

usage designates the use of the I/O buffers in the pool. Use one of the following to specify *usage:*

`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` is only valid for POOL7. If specified for POOL 0 through 6, `S` is changed to `D`.

If you do not specify pool parameters, defaults are assigned. The default buffer-size for Database (POOL) and Scratch (POOL) is 26624; the number of buffers is USERS * 3. A minimum of ten buffers is assigned to each pool. The number of buffers that are assigned for S2KUSERS (POOL0) is TPTHREADS + 1; the default size is 12288. If you specify one or more pool parameters, no defaults are used, even if your specification is inadequate.

PQA=FIFO or PQA=PRTY[/*n*/*y*]

determines the priority queuing algorithm

Syntax

PQA=FIFO
PQA=PRTY[/*n*/*y*]

Details

The PQA parameter determines the priority queuing algorithm. The default is FIFO, which means that user queuing is executed on a first-in, first-out basis. All users have a priority of 0 regardless of the value in the user request block. PRTY[/*n*/*y*] executes user queuing by user priority that is set by specifying *n* and *y*. 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. If you do not specify *n* and *y* when PQA=PRTY, the values for *n* and *y* are 0, and Multi-User does not examine or adjust the queue.

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 the PQA parameter value is altered, Multi-User must change the priority of the queue entry for each active user. The accounting exit in dependent region interfaces allows modification of the additional field URBPRTY, which can have a value that ranges from 0 to 255. This exit is available during sign on. The URBPRTY has no effect on FIFO queuing. URBPRTY is a 1-byte field at displacement X'1F' into the URB. Here is the relationship between the PQA parameter value and the priority of a user:

- If PQA=FIFO, the priority of the user, which is shown by 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 the user is chained into queues in priority order. The value is 0 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 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 value from PRTY to FIFO, each user priority is set to 0 in his Multi-User queue entry. His URBPRTY remains unchanged.

The Database Administrator is responsible for ensuring the appropriate exits are linked with the Multi-User interfaces. User priority queuing is written to the Accounting Log, and the Diagnostic Log thread start message shows user priorities.

Multi-User maintains a Dispatch Queue (DQ) in priority order or FIFO, based on the value of the PQA execution parameter. The Multi-User Scheduler always takes the first entry from 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.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 to 255. The lowest priority is 0, which is the default.

PREFIX=*prefix*

specifies the default user prefix to a maximum of 35 characters

Syntax

PREFIX=*prefix*

Details

The PREFIX parameter value precedes the S2KPAD*nn*, SF*nn*, Savefile, Keepfile, S2KDBCNT, and S2KPARMS files when they are dynamically allocated. *prefix* must be 1 to 35 characters in length and must be 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 use the DSN= option on an SCF ALLOC command to specify the database data set name (enclosed in quotation marks). The PREFIX parameter must be specified in the EXEC statement if you want the software to dynamically allocate the S2KPARMS file.

NOTE: 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.

RETPRD=*nnnn*

specifies a retention period for dynamically allocated Savefiles and Keepfiles

Syntax

RETPRD=*nnnn*

Details

The RETPRD parameter specifies a retention period for dynamically allocated Savefiles and Keepfiles. *nnnn* is a 4-digit value that indicates the number of days that the file is retained. The value can range from 1 to 9999. There is no default value.

RW=*option*

determines whether users can access the REPORT processor

Syntax

RW=YES|NO

Details

The RW parameter determines whether users can access the REPORT processor. RW=YES means that the REPORT processor is available. For an SCF job, the work area requires 61K bytes (9K more than usual). In a Multi-User environment, each Thread Work Area requires 61K bytes. The default is NO, which means that the REPORT processor is not available, and 9K fewer bytes are needed per work area.

SAME=*option*

controls the use of the SAME operator across SCF TP segments

Syntax

SAME=YES|NO

Details

The SAME parameter controls the use of a previously specified where-clause across transmission 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. SAME=NO means that the results of a where-clause are not saved across segments. If a user tries to use the SAME operator, SYSTEM 2000 issues an error message. The default is YES.

SAVEUNT=*unit*

specifies the type of disk unit for a dynamically allocated Savefile

Syntax

SAVEUNT=*xxxxxxxx*

Details

The SAVEUNT parameter specifies the type of disk unit to request when dynamically allocating a Savefile. *xxxxxxxx* is a maximum of 8 characters of disk type (3380, 3390, DISK). If this parameter is omitted, the disk type defaults to SYSDA.

SDBS=*blocks*

specifies the initial number of Database Definition Blocks to be used for small databases

Syntax

SDBS=*blocks*

Details

The SDBS parameter specifies the number of small databases that SYSTEM 2000 can have open at one time without having to do a GETMAIN to acquire memory for additional definition blocks. 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. The range is 0 to 32. The default is 1.

SDBSIZE=*n*

specifies the maximum number of components that can be defined in a small database

Syntax

SDBSIZE=*comp*

Details

The SDBSIZE parameter specifies the maximum number of items that a small database can contain. SYSTEM 2000 uses the value for SDBSIZE to determine the amount of memory to allocate for each small Database Definition Block. The range is 1 to 10000. The default is 430.

SFPRI=*n*

alters the Sort file primary space

Syntax

SFPRI=*n*

Details

The SFPRI parameter specifies the primary space when dynamically allocating Sort files. Dynamic allocation of Sort files does not create any permanent data sets; the data sets are temporary files. If the software dynamically allocates a Sort file, it dynamically de-allocates the file at the end of the single-user job or the Multi-User session. The range is 1 to 32767. The default is 1.

SFSEC=*n*

alters Sort file secondary space

Syntax

SFSEC=*n*

Details

The SFSEC parameter specifies the secondary space when dynamically allocating Sort files. Dynamic allocation of Sort files does not create any permanent data sets; the data sets are temporary files. If the software dynamically allocates a Sort file, it dynamically de-allocates the file at the end of the single-user job or the Multi-User session. The range is 0 to 32767. The default is 1.

SFSPACE=*option*

specifies the Sort file space unit

Syntax

SFSPACE=CYL|TRK

Details

The SFSPACE parameter specifies the space unit when dynamically allocating Sort files. Dynamic allocation of Sort files does not create any permanent data sets; the data sets are temporary files. If the software dynamically allocates a Sort file, it dynamically de-allocates the file at the end of the single-user job or the Multi-User session. The default is CYL.

SFUNIT=*unit*

specifies the Sort file UNIT value

Syntax

SFUNIT=*xxxxxxxx*

Details

The SFUNIT parameter specifies the unit device when dynamically allocating Sort files. Dynamic allocation of Sort files does not create any permanent data sets; the data sets are temporary files. If the software dynamically allocates a Sort file, it dynamically de-allocates the file at the end of the single-user job or the Multi-User session. The default is SYSDA.

SID=*n*

establishes the System ID

Syntax

SID=*n*

Details

The SID parameter sets the System ID that appears in the Multi-User Accounting Log and write-to-operator (WTO) messages. The range is 0 to 99. The default is 1.

STAE=option

specifies how error trapping is to be handled

Syntax

STAE=YES|NO|NODUMP

Details

The STAE parameter specifies how error trapping should be handled. STAE=YES causes SYSTEM 2000 to format the Extended Specify Task Abnormal Exit (ESTAE) snapshot when an unrecoverable error occurs. The formatted ESTAE work area appears in the users Message File (S2KMSG). A full snapshot is written to the S2KSNAP file. For a Multi-User abend, the formatted ESTAE work area is written to the Diagnostic Log. The default is YES.

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. The ID of the full snapshot is 50 for SYSTEM 2000 Error Code 820, or 51 for SYSTEM 2000 Error Code 826. If an abend has not occurred, but SYSTEM 2000 detects an unrecoverable error, then another 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 DDname is hard coded in the snapshot data control block (DCB) 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.

STAE=NO means that 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 a system abend are sent to the SYSUDUMP or SYSABEND files. STAE=NO does not disable any ESTAE trapping for dependent region PLEX sessions in a Multi-User environment.

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

NOTE: SYSTEM 2000 dynamically allocates an S2KSNAP file if necessary and if it is not already allocated in the JCL or CLIST. The S2KSNAP file default values are SYSOUT=A and HOLD.

STARTTP=*option*

initiates the SCF TP facility

Syntax

STARTTP=YES|NO

Details

The STARTTP parameter enables the SCF TP facility during system initialization. YES means that the operator does not have to enter the START TP command. NO delays SCF TP initialization until the console operator gives the START TP command. Use NO to allow critical SYSTEM 2000 sessions to be executed with reduced interaction from the SCF TP facility. The default is YES.

STAX=*option*

specifies whether an attention interrupt exit is enabled

Syntax

STAX=YES|NO

Details

The STAX parameter allows you to specify whether an attention interrupt exit is enabled. The exit is enabled in both the SCF and SAS/FSEDIT environments. This exit also allows SYSTEM 2000 logic to get control when an attention interrupt is issued during SYSTEM 2000 processing. The default is YES.

SYSOUT=*x*

specifies the class when dynamically allocating S2KMSG to a SYSOUT data set

Syntax

SYSOUT=*x*

Details

The SYSOUT parameter specifies the class to use when dynamically allocating the S2KMSG file to a SYSOUT data set. *x* is a 1-character value that represents a valid SYSOUT class. This parameter represents the value you would place in a DD card, for example, //S2KMSG DD SYSOUT=A. The default is A.

S2KMSGL=*nnn*

specifies the LRECL for a dynamically allocated S2KMSG file

Syntax

S2KMSGL=*nnn*

Details

The S2KMSGL parameter allows you to specify the logical record length for an S2KMSG file that is dynamically allocated by SYSTEM 2000 in a single-user session. *nnn* specifies a 2- or 3-digit value in a range from 44 to 256. In a submitted job, if S2KMSGL is omitted, the logical record length for a dynamically allocated S2KMSG file will be 133. In a TSO session, the logical record length for a dynamically allocated S2KMSG file will be equal to the logical line size of the terminal. S2KMSGL is ignored in a Multi-User session.

THREADS=*n*

specifies the number of threads that can be used simultaneously

Syntax

THREADS=*n*

Details

The THREADS parameter specifies the number of threads that can be used simultaneously in a Multi-User session. 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 of each Thread Work Area (RWTCOM01) is 61K with the REPORT processor (RW=YES). One set of SF files (Sort files) is required for each thread. The range for *n* is 1 to 63. The default is 1. For a single-user environment, THREADS must equal 1.

TPSCRUN=*n*

specifies the maximum number of active and inactive SCF TP users allowed

Syntax

TPSCRUN=*n*

Details

The TPSCRUN parameter specifies the maximum number of active and inactive SCF TP users. The inactive users are signed on but are not currently active. The S2KUSERS file must be allocated with enough disk space to handle the number of SCF TP users specified. The CISIZE for the S2KUSERS file is 12288 bytes. The range is 1 to 10000. The default is 180.

TPSEG=*hsec*/*nio*[/DELAY]

controls what SCF TP job data, if any, is written to the Accounting Log

Syntax

TPSEG=NO|*CPU-time*/NO|*I/O-count* [/DELAY]

Details

The TPSEG parameter controls how much accounting data is generated for Multi-User segments. Multi-User produces accounting data for segments that meet the *I/O* criterion and whose *CPU-time* equals or exceeds the specified time.

CPU-time is the threshold time in 100^{ths} of a second, specified as an integer. The range is 0 to 999999. *I/O-count* is the threshold number of database I/O's, specified as an integer. The range is 0 to 999999. Multi-User produces accounting data for segments meeting the *CPU-time* criterion and whose I/O count equals or exceeds the specified number. 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. NO means that the *CPU-time* or the *I/O-count* is not used as a criterion. The default is NO, NO.

You can also issue an operator console command to control segment data generation. The syntax for that command is:

```
|MODIFY      jobname,    |START    STAT   TP|ALL
|F                       |STOP     STAT   TP|ALL
```

jobname refers to the SYSTEM 2000 job name. STOP tells Multi-User to stop writing accounting data in the Accounting Log file. STOP applies if the writing started because DELAY was not specified or because of a previous START command. TP controls writing of accounting data for SCF TP segments, and ALL means NL, PL, and TP.

TPTHREADS=*n*

specifies the number of SCF TP users that can be queued

Syntax

TPTHREADS=*n*

Details

The TPTHREADS parameter specifies the maximum number of active SCF TP users that can be queued for SYSTEM 2000 processing. Queued means that 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 a "busy" message appears. The range for *n* is 0 to 230. The default is 5.

For XMS Multi-User software, the TPTHREADS 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. 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 regenerating the SVC, set a high number in the SVC TPUSERS parameter so that it will accommodate any increase in the number specified in TPTHREADS. The TPTHREADS parameter value can exceed the THREADS parameter value to provide for command queuing. This action risks saturating SYSTEM 2000 with TP activity at times of heavy TP message traffic. However, it can reduce command re-tries in the TP transaction caused when SYSTEM 2000 signals a "busy" condition (due to the absence of an available queue slot).

TPVOL=*volume*

specifies the volume for a dynamically defined S2KUSERS file

Syntax

TPVOL=*xxxxxx*

Details

The TPVOL parameter specifies the volume where the S2KUSERS file resides when a volume is dynamically defined. *xxxxxx* is a 6-character volume serial number. There is no default value.

USERS=*n*

specifies the maximum number of users that can execute jobs concurrently

Syntax

USERS=*n*

Details

The USERS parameter specifies the total number of jobs that can be executing SYSTEM 2000 concurrently, that is, the total number of batch SCF jobs, batch PLEX jobs, and PLEX TP jobs. The range is 1 to 230. The default is 16.

For XMS Multi-User software, the USERS 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 a high number in the SVC USERS parameter so that it will accommodate any increase in the number specified in the USERS parameter.

NOTE: 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.

XBUF=*option*

enables the XBUF feature

Syntax

XBUF=YES|NO

Details

The XBUF parameter enables or disables the XBUF data space feature in a single-user job or in a Multi-User environment under OS/390. XBUF=YES activates XBUF software and data spaces are created according to XBUFTBL. The default is NO, which means that XBUF software is not activated.

XBUFSUF=*suffix*

identifies a specific XBUF load module if several modules exist in the load library

Syntax

XBUFSUF=*suffix*

Details

The XBUFSUF parameter identifies (by a suffix) a specific XBUF load module if more than one exists in the load library. *suffix* is any single national character that is valid for a PDS member name and is appended to the XBUFTBL load module name. You do not need *suffix* if you generate only one XBUF load module. 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 using the XBUFSUF execution parameter. *suffix* indicates which configuration to activate. For example, to activate the load module named XBUFTBLA, specify XBUFSUF=A in the S2KPARMS file. There is no default value.

YEARCUTOFF=*year*

defines a century window for date expansion

Syntax

YEARCUTOFF=*nnnn*

Details

The YEARCUTOFF parameter defines a century window for date expansion. *nnnn* is a 4-digit year value that ranges from 1500 through 9900. When a date that has a 2-digit year comes into the system, the YEARCUTOFF parameter determines what century the date is in. The default is 1920. Using the default, the date 01/01/18 is interpreted as 01/01/2018 and the date 01/01/40 is interpreted as 01/01/1940.

Chapter 15: Utilities

Converting Databases to Version 1 Format

Databases created before Version 1 must be rebuilt in the new Version 1 format. To convert your existing databases, use the CVRTV1 conversion program. CVRTV1 reads an existing Savefile and produces a Version 1 database. After you have converted a database to Version 1 format, the database files are not compatible with previous releases of SYSTEM 2000. Version 1 software cannot access database tables created with previous releases.

CVRTV1 Conversion Program

The CVRTV1 program

- runs in stand-alone batch mode, which allows you to convert individual databases offline as needed.
- reads a Savefile as input and writes VSAM database files.
- allows you to change the block size of a saved database by using the REBLOCK option.
- rebuilds the entire database structure, including indexes.

Note: The CVRTV1 program cannot convert Savefiles that were written prior to Release 10.1 of SYSTEM 2000.

To execute the CVRTV1 program, you need a set of Version 1 database files and a valid Savefile. If the database files contain an existing database, error message 56 is issued (which indicates that the database already exists) and processing stops.

When your datasets are VIO data sets, the CVRTV1 execution takes a little more time than a RESTORE command. Testing has shown that the use of RIO data sets is less efficient. The number of key date fields can affect the total time required for conversion.

The Input Parameter File

The DDname of the input parameter file must be CNVPARM. You must supply this file to the CVRTV1 program. CNVPARM contains the name of the database to be converted. For databases created before Version 1, you can specify the CONVERT option, the REBLOCK option, or both. If you use both options, they can be specified in any order. For Version 1 databases, you can specify only the REBLOCK option.

The syntax for a parameter file is

database [(*action* [,*action*])]

database is the name of the database in the Savefile. It is assumed that the CISIZE of the new database files will be different from the old block sizes indicated in the Savefile. All of the Version 1 block sizes are different from block sizes in previous releases. *action* specifies conversion or re-blocking of the database.

action CONVERT|REBLOCK

CONVERT indicates that you want to convert a database that was created by Release 10.1, 11.0, 11.5, 11.6, 11.6-3, 12.0, 12.1, or 12.B. When you use CONVERT with Version 1 files, those files are automatically re-blocked. This is the default. REBLOCK indicates that you want the database to be re-blocked. Lack of the REBLOCK option will not cause the conversion to fail.

Note: You can re-block a Version 1 database by specifying the REBLOCK option only, for example,

V1EMPLOY(REBLOCK)

Here are sample parameter files for four databases:

EMPLOYEE (CONVERT) converts pre-Version 1 EMPLOYEE database.

CARS(CONVERT) converts pre-Version 1 CARS database.

SALES(CONVERT,REBLOCK) converts and re-blocks pre-Version 1 SALES database.

V1SALES(REBLOCK) re-blocks the Version 1 SALES database.

Sample DDnames

You must define all database files in the JCL. Each database file DDname must use the standard SYSTEM 2000 naming conventions. That is, each of the six DDnames, which represent database Files 1 through 6, consists of the first seven characters of the database name (with blanks replaced by X's) followed by an integer from 1 to 6. The Savefile DDname has an 'S' as the last character. For example, if you specify EMPLOYEE(CONVERT), DDnames in the JCL for the new database files would be

```
//EMPLOYE1
       .
       .
       .
//EMPLOYE6
//EMPLOYES
```

Temporary Data Sets

The CVRTV1 program uses four temporary data sets. The best way to create these files is to allow the conversion process to dynamically allocate them as needed. You can also specify these files in a batch job. The temporary files are

TEMPFIL2 preserves database File 2 data.

TEMPFIL4 preserves database File 4 data.

SORTIN is the input file for the sort process.

SORTOUT is the output file for the sort process.

Note: The LRECL of TEMPFIL2 and TEMPFIL4 must match the page size of database File 2 and File 4, respectively, that are in the Savefile. The SORTIN and SORTOUT files are used to sort File 4 entries and are required to have an LRECL of 4.

By default, the temporary files are allocated to VIO devices. If your site has limitations on the size of VIO data sets and you are receiving dynamic allocation failures, change the default unit for dynamic allocation. You do this by using an execution parameter as follows:

```
//CONVERT  EXEC PGM=CVRTV1,PARM='DAUNIT=unit'
```

unit is your installation setting for DASD devices (3380, SYSDA, RIO, and so on).

SYSOUT Data Set

The SYSOUT data set must be defined for use by the system Sort facility. This data set is used to hold the messages from the Sort facility. SYSOUT should be defined as follows:

```
//SYSOUT   DD  SYSOUT=A
```

Executing the CVRTV1 Program

Follow these steps to convert or re-block a database with the CVRTV1 program:

1. Allocate Version 1 database Files 1 through 6 in your JCL. If the CISIZE of your database files is different from the block sizes of the database on the Savefile, the conversion process automatically re-blocks the database even if you do not specify the REBLOCK option in the conversion parameter card.

2. Establish the parameter file for the program. Place the parameter card in a data set that has the DDname CNVPARM. Each card contains the name of a database to be converted and/or re-blocked, followed by the CONVERT or REBLOCK options where appropriate. You can convert only one database in each job execution.

3. Execute the CVRTV1 program.

Sample JCL

This sample JCL executes CVRTV1 to convert and re-block the EMPLOYEE database. It includes the REBLOCK option, and it assumes that the new block sizes were specified when the database files were cataloged.

This job is in your Version 1 CNTL library as member JCLCNVRT.

```
//JCLCNVRT JOB (ACCTINFO),
//    S2K,REGION=0M
//*
//*****************************************************
//*  CONVERT DATABASE TO VERSION 1
//*
//*    1.  CHANGE DBNAMEX TO YOUR DATABASE NAME
//*    2.  FILL BLANKS WITH X'S IN DDNAME FOR 7 CHARS
//*    3.  EXTERNAL SORT MAY BE INVOLVED
//*        RUN WITH MAXIMUM REGION SIZE AVAILABLE
//*    4.  TEMPORARY FILES ARE DYNAMICALLY ALLOCATED FOR
//*        FILES 2 AND 4. YOU MAY SPECIFY DD STATEMENTS
//*        FOR TMPFILE2 AND TMPFILE4 IN THE JCL.
//*
//*  NOTE: FOR BEST ELAPSED TIME PERFORMANCE,
//*  ENSURE THAT TMPFILE4 IS ALLOCATED TO UNIT=VIO.
//*  CHECK WITH YOUR SYSTEMS PROGRAMMER TO SEE IF THERE
//*  IS ANY SIZE LIMITATION AT YOUR SITE.
//*
//*    5.  CNVPARM INPUT:
//*    <DATABASE NAME> (<OPTIONS>)
//*    WHERE <OPTIONS> CAN BE:
//*        A.  CONVERT - CONVERT DATABASE
//*        B.  REBLOCK - MUST SPECIFY IF YOU INTEND
//*                      TO REBLOCK ANY DATABASE FILE
//*        C.  NOINDEX - CONVERT, BUT DO NOT RE-CREATE
//*                      INDEX TABLES
//*
//*********************************************************
//*
//CONVERT   EXEC PGM=CVRTV1,REGION=0M
//STEPLIB   DD DISP=SHR,DSN=S2K.V1.LOAD
//SYSOUT    DD SYSOUT=A
//SYSPRINT  DD SYSOUT=A
//TIMEIO    DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//DBNAMEX1 DD DISP=OLD,DSN=S2K.V1.DBNAMEX1
//DBNAMEX2 DD DISP=OLD,DSN=S2K.V1.DBNAMEX2
//DBNAMEX3 DD DISP=OLD,DSN=S2K.V1.DBNAMEX3
//DBNAMEX4 DD DISP=OLD,DSN=S2K.V1.DBNAMEX4
//DBNAMEX5 DD DISP=OLD,DSN=S2K.V1.DBNAMEX5
//DBNAMEX6 DD DISP=OLD,DSN=S2K.V1.DBNAMEX6
//DBNAMEXS DD DISP=OLD,DSN=S2K.V1.DBNAMEXS
//CNVPARM  DD *
DBNAMEX  (CONVERT,REBLOCK)
//*
```

In this example, the CNVPARM is specified inline. Alternatively, you could specify an 80-byte sequential data set that has 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:

```
//CNVPARM  DD DISP=SHR,DSN=USER.PDS(PARMCARD)
```

Note: The system Sort facility can only do a limited number of occurrences in core. After that limit is exceeded, the Sort process requires work files to proceed. If you are getting messages from sort during a conversion that indicates that the in-core sort limit has been exceeded, you will need to add work files to your JCL. The Sort manuals describe how to define these work files, but as a general guideline, the following definitions should work:

```
//SORTWK01  DD  SPACE=(CYL,(15,5)),UNIT=3390
//SORTWK02  DD  SPACE=(CYL,(15,5)),UNIT=3390
//SORTWK03  DD  SPACE=(CYL,(15,5)),UNIT=3390
```

Conversion Report

After a successful conversion, a report is generated that contains the database name, the original SYSTEM 2000 release number, date the original database was defined, date the new database was converted, and information about page counts for each file in the old database (Savefile) and the new database. This report is written to the SYSPRINT file and requires an LRECL of 132.

The format of the conversion report is controlled by the version of SYSTEM 2000 that is identified in the Savefile. If the Savefile is prior to Version 1, the 'old' sizes and counts are by page size. If the Savefile is Version 1, the 'old' sizes and counts are by CISIZE.

PLEX Program Generators: S2KGLOAD and S2KGUNLD

In the daily process of updating your databases, records are added and deleted and, eventually, your database gets "skewed." This means that the records in your database change from being in load order to being scattered across many pages of the database. This can cause increased overhead and increased I/O, which can slow down your processing time. Occasionally, it is a good practice to unload your data from the database and re-load it in a new database order. In SCF, this process can be slower than you want and can require a lot of resources for a large database. The PLEX optimized load process is much faster than SCF and requires considerably fewer resources. In order to make this process easier for you and to eliminate the necessity for you to write PLEX programs to do this, two program generators are supplied for your use. These program generators, S2KGLOAD and S2KGUNLD, are written in PL/I and generate COBOL PLEX programs that are custom-built for unloading and loading a specific database. The unload program can be run against an older database to generate data that the load program puts into a current database.

The S2KGLOAD and the S2KGUNLD program can also be used for converting databases. The CVRTV1 conversion program converts a database to the current format but does not affect any skewness that exists in the original database. Both of these programs are delivered in source and executable form. If you want to change the way these programs work or if you have a different version of PL/I (the executable is compiled under Release 5.1 of PL/I), you might need to re-compile one or both of the programs. The JCLGPL1 job (generated at installation time) is a sample for compiling and linking the program generators. It is recommended that you run the programs using the executable provided.

Input to the Program Generators

In order to create the UNLOAD and LOAD programs, you must provide the input file DESC, which contains the output from a SYSTEM 2000 DESCRIBE command, for the database you want to process. This DESCRIBE output is used to construct PLEX schemas in the resulting COBOL PLEX programs. Member JCLDESC shows how to generate DESCRIBE output in a sequential file. The DESCRIBE output contains the original defined lengths for components. For character and text components, this length might not be satisfactory for unloading or loading your database. Because character and text components are allowed to overflow in the database, the values for these two component types can be a maximum of 250 characters in length, regardless of what the definition says. To determine if you have character or text components that overflow, you can run the EXAMINE utility against the database that will be processed. The output from the EXAMINE utility indicates the longest value for a specific component. You should update your DESCRIBE file to match the longest length of data or supply a "change" file as input to the generating process to make sure you get all the data for a component.

If you are providing a 'change' input file, the DDname for the file must be SYSIN. Using this input process, you can cause the length of character and text fields to be changed to fit the data that is in your database. This input file contains two kinds of records. The first record in the file must be a BLKSIZE / LRECL record formatted as follows:

bbbb rrrr

bbbb is the requested block size (BLKSIZE), *rrrr* is the requested record length (LRECL). This block size and record length are used to build the COBOL file definition for the OUTFL data set in the generated COBOL PLEX unload program. The UNLOAD program writes the unloaded data to the OUTFL data set. If you specify 0's for the block size, a default of 6216 is assigned. The default record length will be the length of the longest record in the database. If the SYSIN file is omitted, the default block size and record length are used. All of the other records in this file must have the following format:

cccc llll

cccc is a database component number. *llll* is the requested length.

If you want your database component 12 (C12) to have a length of 44 and your UNLOAD file to have the characteristics BLKSIZE=23464 and LRECL=64, your SYSIN data set would be set up as follows:

```
23464 64
12 44
```

Output from the Generator Programs

The output from the execution of the generator programs is COBOL PLEX source code that either unloads or loads data. The PROGRAM DD statement defines where the resulting COBOL program source will be placed. The PROGRAM DD statement is required for execution of the generator programs. You can use this statement to point to a PDS or to an 80-byte sequential file.

Executing the Generated Programs

The generated COBOL PLEX programs must be compiled and linked by using JCLGCOB (generated during installation). The input for this process is the PROGRAM data set that is generated by the execution of one of the program generators. Sample job JCLGCBGO shows how to run the generated programs after they have been compiled and linked.

When using the programs for conversion purposes, make sure that you have specified a load library that has a Release number that is the same as your existing database. For example, if you are unloading a Release 12.0 database, use the Release 12.0 load library in the JCLGCBGO JCL.

Execution of the generated programs requires the database files for the database that is going to be unloaded or loaded. The UNLOAD program requires the OUTFL DD statement. The OUTFL data set contains the data that is unloaded from a SYSTEM 2000 database. The UNLOAD program requires either the master password or a secondary password that has retrieval authority to all components. The password to be used is supplied as an execution parameter in the EXEC statement. The format of the parameter must be PARM='ABCD' where ABCD is the password to be used to access the database. If this parameter is not supplied, a SYSTEM 2000 return code 24 is received at execution time. DCB parameters are not required in the OUTFL DD statement because the UNLOAD program has defined them by using a file definition. The block size for the OUTFL data set cannot exceed 32760. The default block size is 6216. The record length for the OUTFL data set must be at least 4 less than the block size. The default record length is the length of the longest record in the database.

The LOAD program requires the INFL DD statement that points to the data set that was generated by the UNLOAD program (data set OUTFL). The LOAD program also requires the master password for loading the new database. The password is supplied as an execution parameter. DCB parameters are not required in the INFL DD statement because the LOAD program has defined them by using a file definition.

Before you execute the generated LOAD program, you must define the database that will be loaded. If you already have a process in place to define a database, use it. Otherwise, you can follow member JCLGDEF (generated at installation time), which is a sample job for defining a database.

Installing the Program Generators

The processes to generate and execute the COBOL UNLOAD and LOAD programs are controlled by the JCL members described in this section. This JCL is generated at installation time. When using the jobs for conversion purposes, the STEPLIB, SYSLIB, and SYSPARM data sets must match the release of the database that is being processed. The JCLGPL1 member is generated to allow you to compile and link the S2KGUNLD and S2KGLOAD programs in your PL/I environment. The distributed executable versions of these programs are compiled and linked in Release 5.1 of PL/I. This is the sample JCL:

JCLGCBGO executes the generated COBOL programs.

JCLGCOB pre-compiles, compiles, and links the generated COBOL programs.

JCLGDEF defines a new database.

JCLDESC creates a DESCRIBE file and a database definition file.

JCLGENCB executes PL/I programs to generate the COBOL programs.

JCLGPLI compiles and links PL/I programs.

The following is an example scenario of what might be involved when using the new release of SYSTEM 2000 with existing databases.

You have an existing EMPLOYEE database created under Release 12.0 of SYSTEM 2000 that is not compatible with Version 1, and you need to convert the database to the current release. You also want to clean up some database skewness at the same time.

Step 1: To execute the following command sequence, run a standard SCF job in the existing database environment.

```
USER,DEMO:DBN IS EMPLOYEE:
REPORT FILE IS MYREPT:
DESCRIBE:
EXIT:
```

This creates the file DESCRIBE from the current EMPLOYEE database in the MYREPT data set. This data set should be an 80-byte sequential file or a member of a PDS.

Step 2: Using the EXAMINE utility, get the longest value for each character and text component.

Step 3: Build a SYSIN data set for the S2KGUNLD execution that changes the component lengths for your character and text components that have values longer than those shown in the file DESCRIBE. For example, the output from DESCRIBE shows that C2 has a defined length of 10 characters. The EXAMINE utility shows that the longest value for C2 is 22 characters. Therefore, you need to build a record in the SYSIN data set that specifies 2 22.

Step 4: Run the JCLGENCB job to generate the COBOL UNLOAD program. Use the output from Step 1 as the DESCIBE data set, and use the result of Step 3 as the SYSIN data set. The result is the source code for a COBOL PLEX program that unloads your existing database with all values intact.

Step 5: Run the JCLGCOB job to pre-compile, compile, and link your generated COBOL UNLOAD program.

Step 6: Run the JCLGCBGO job to unload your existing database. The result of this run is the OUTFL data set. This will be the input for your COBOL LOAD program.

Step 7: Use the JCLGDEF job or a standard SCF job to create a new database in the current SYSTEM 2000 environment.

Step 8: Run the JCLGENCB job to create a COBOL LOAD program. Use the same input for this as you used for Step 4. The result is the source code for a COBOL PLEX program that will load your new database.

Step 9: Run the JCLGCOB job to pre-compile, compile, and link your generated COBOL LOAD program.

Step 10: Run the JCLGCBGO job to load your new database with the data that was unloaded in Step 6. Use the OUTFL from Step 6 as the INFL for this step. The result will be the current version of your EMPLOYEE database, which is not skewed.

Step 11: Validate that all of the data that was on the old database is on the new database. You can use the EXAMINE utility to do this, or you can use a series of PRINT COUNT commands for a lower level of validation. Run the EXAMINE program against your new database and compare the results from the EXAMINE program that was run against your old database. The results should be the same. You can do the same kind of comparison using the output from the PRINT COUNT commands.

Timing Statistics: QAEXIT AND QASTAT

QAEXIT (SCF) and QASTAT (PLEX) allow you to gather CPU timing for commands and I/O counts by file. In SCF, you issue the system-wide QAEXIT command. In PLEX, you issue a call to QASTAT. The format of information produced by QAEXIT and QASTAT is identical. When gathering these timing statistics, consider the following guidelines:

- You can use QAEXIT and QASTAT in single-user or Multi-User mode.
- You can monitor up to 256 files for I/O counts. If this maximum is exceeded, a message is issued during the first call for timing and during all subsequent calls that produce I/O counts for the first 256 files (DD statements).
- The execution time for producing timing statistics is not included in any of the CPU totals.

QAEXIT and QASTAT Output

The first request for timing initializes a time and displays a set of comments that explain the content of fields in the output. These comments do not appear for subsequent 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. CPU time is reported to the nearest 1000^{th} of a second.

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" later in this chapter. If you want several consecutive calls totalled, 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 re-set. If you do not specify any sequence numbers in your timing requests, the 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.

The subsequent lines of output show the I/O counts for the first 256 files specified in the JCL DD statements. Two counts appear to the left of each filename (DDname). If blanks precede the filename, the file has not been accessed.

- The first count is the number of I/O's since the last QAEXIT command or PLEX call to QASTAT.
- The second count is the total number of I/O's 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 displayed filenames depends on the order of 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.

Output 15.1 shows a sample display of timing statistics produced by issuing QAEXIT commands. The timing and I/O counts are mixed with the echoes of SCF commands in 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 written to the TIMEIO file (or the FT09F001 file).

QAEXIT and 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 1, First Edition.*

Output 15.1 Format of QAEXIT and QASTAT

```
05/03/99  16:14:47  BEGIN SYSTEM 2000 - VERSION 1
---
USER,ABC: QAEXIT:
---TIMING AND I/O STATISTICS BEGIN---00.214---16:14:4829----------------------
    I/O IS LISTED BY DDNAME ACROSS PAGE WITH COUNTS THEN DDNAME
    1ST FIGURE IS EXCPS BECAUSE 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
---
MAP: QAEXIT:
--------------------CALL # 001/  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
           221   221 TESTXXX4      211   211 TESTXXX5     211   211       TESTXX6
---
IT C0 * 0 EQ 1*A*2* 1*10*11*AX1*12* 2*20*21*AY1*22* 3*END*:
 -342-       1 SELECTED RECORD(S) -
 ---
QXEXIT:
```

Output 15.1 Format of QAEXIT and QASTAT (*continued*)

```
--------------------CALL # 002/  C   -------- 16:15:520-----------------------
 TIME -   CUR=00000.055    SEQ=00004.587    TOT=00004.587    OVD=0000.010
 I/0  -           185 STEPLIB                  STEPLIB       SYS00866
                      SF01                     SF02   SF03
                      SF04                     SF05          SF06
                      S2KSYS01                 S2KSYS02                  S2KSYS03
                      S2KSYS04                 S2KSYS05             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  05/03/99   END  SYSTEM 2000 - VERSION 1.
```

Timing for the Self-Contained Facility: QAEXIT Command

To gather the timing statistics and the I/O counts during an SCF job, issue 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/O's accumulated for each file since the beginning of the session. However, because the first call serves to initialize the timer, all timing information is 0 for this call. Subsequent QAEXIT calls provide timing information from this initial point. For details about the content and format for displaying statistics, see "QAEXIT and QASTAT Output." Each time you give the QAEXIT command, SYSTEM 2000 software writes the statistics to the Message File for your individual SCF job.

When using QAEXIT,

- in a Multi-User environment, SYSTEM 2000 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 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.

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 program written in COBOL, FORTRAN, PL/I, or Assembler.

Use the following formats:

PLEX Language	CALL Syntax
FORTRAN	CALL QASTAT [(*var*)]
COBOL	CALL 'QASTAT' [(*var*)] .
PL/I	CALL QASTAT [(*var*)] ;
Assembler	CALL QASTAT [, (*var*)]

var is a fullword binary number. This number is optional. It affects the CPU time displayed for SEQ (see "QAEXIT and QASTAT Output"). If you want the CPU time and the I/O counts totalled 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 re-set. If you do not specify any sequence numbers in your timing requests, the SEQ equals TOT, which 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 FT09T001 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.

Output 15.2 The TIMEIO File

```
//TIMEIO DD SYSOUT=A
//FT09F001 DD SYSOUT=A
```

The format and content of statistics written in this file are the same as for QAEXIT. The output file is re-started 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 environment to close it at the end of the step.

Linking QASTAT into a PLEX program You can link QASTAT with any COBOL, FORTRAN, PL/I, or Assembler language program. However, you must include QASTAT with your program by using 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, the two references S2KCVT and COMENT are unresolved, but this does not cause any problem during execution of your program.

Note: The QASTAT module requires an additional 6244 bytes of memory for the PLEX job.

EXAMINE

EXAMINE is a database analysis tool. It uses Files 1, 3, 5, and 6 and generates up to three reports, depending on error conditions. A report written to the DD statement PRINTER is always generated. The DD statements PRNTF5 and PRNTF6 are written to only if errors are found in those files.

The EXAMINE databases analysis tool

- verifies that pointers are within the range of the database records in use. A pointer that is too large in File 5 or File 6 (overflow) will be detected by EXAMINE.
- performs elementary validation of File 6 data. Validates the sign bit and the digits in numeric fields. The format of the numeric field is validated; the value is not validated.
- provides a count of every record and item in the database. The count shows the number of times an item has a value, and the number of times the item does not have a value.
- counts the number of times a character or text item overflows (actual length is greater than defined length).
- identifies the length of the longest value for each item.
- shows the amount of PAD and SORT file space needed for an SCF reload.
- counts the number of skewed and re-usable records in the database.

A record is skewed if the record that points to it is on a different page. Some skewed records are likely to exist, even in a freshly loaded database, however, an excessive number of skewed records can affect performance. A general guideline is that if skewness is 20% or more of the total record count, your performance is reduced due to increased I/O.

Sample JCL

This job is in your Version 1 CNTL library as member JCLEXAM.

```
//JCLEXAM JOB (ACCTINFO),
//    S2K,REGION=0M
//*********************************************
//*
//*  THIS JOB READS DATABASE FILES 1, 3, 5, AND 6 DIRECTLY
//*  AND PRODUCES A COUNT OF EACH DATABASE COMPONENT.
//*  FOR CHARACTER AND TEXT COMPONENTS, IT COUNTS THE CHARACTERS
     THAT OVERFLOW THE DEFINITION AND
//*  IDENTIFIES THE LENGTH OF THE LONGEST VALUE.
//*
//*********************************************
//*
//EXAMINE  EXEC PGM=EXAMINE
//STEPLIB  DD  DISP=SHR,DSN=S2K.V1.LOAD
//SYSPRINT DD  SYSOUT=A
//PRINTER  DD  SYSOUT=A
//PRNTF5   DD  SYSOUT=A
//PRNTF6   DD  SYSOUT=A
//FILE1    DD  DSN=S2K.V1.EMPLOYE1,DISP=SHR
//FILE3    DD  DSN=S2K.V1.EMPLOYE3,DISP=SHR
//FILE5    DD  DSN=S2K.V1.EMPLOYE5,DISP=SHR
//FILE6    DD  DSN=S2K.V1.EMPLOYE6,DISP=SHR
```

RECHAIN

The RECHAIN program provides a quick-and-easy way to count all records in a database by record type. It also counts File 5 re-usable space entries and counts any re-usable entries that are in error. If an error is found, you can use RECHAIN to correct the error, which likely will result in omitting the erroneous entry.

One of two possible parameters is required in the EXEC statement. One parameter is COUNT, which counts records and re-usable entries. The other is RECHAIN, which rechains all re-usable space. Use RECHAIN only to correct errors.

Sample JCL

This job is in your Version 1 CNTL library as member JCLF5CNT.

```
//JCLF5CNT JOB (ACCTINFO),
//    S2K,REGION=0M
//*******************************************************
//*
//*  THIS JOB READS DATABASE FILES 1 AND 5 DIRECTLY
//*  AND PRODUCES A COUNT OF EACH RECORD TYPE,
//*  BOTH USED AND RE-USABLE ARE COUNTED.
//*
//*******************************************************
//*
//F5COUNT  EXEC PGM=RECHAIN,PARM='COUNT'
//STEPLIB  DD  DISP=SHR,DSN=S2K.V1.LOAD
//SYSPRINT DD  SYSOUT=A
//FILE1    DD  DSN=S2K.V1.EMPLOYE1,DISP=SHR
//FILE5    DD  DSN=S2K.V1.EMPLOYE5,DISP=SHR
```

INDEXRPT

The INDEXRPT utility is a Database administrator tool to assist in maintaining the database indexes. The utility is designed to analyze database indexes in Files 2 and 4 and to produce a report that breaks down the indexes by component showing File 2 pages and File 4 blocks in use.

Note: When you run the INDEXRPT utility, if any records are written to the ERRORS file, you should contact SYSTEM 2000 Technical Support.

Sample JCL

This job is in your Version 1 CNTL library as member JCLINDX.

```
//JCLINDX JOB (ACCTINFO),
//    S2K,REGION=0M
//*******************************************************
//*
//*  THIS JOB READS DATABASE FILES 1, 2, 3, AND 4 DIRECTLY
//*  AND PRODUCES A REPORT ON EACH KEYED DATABASE COMPONENT.
//*
//*******************************************************
//*
```

```
//INDEXRPT EXEC PGM=INDEXRPT
//STEPLIB  DD  DISP=SHR,DSN=S2K.V1.LOAD
//SYSPRINT DD  SYSOUT=A
//FILE1    DD  DSN=S2K.V1.EMPLOYE1,DISP=SHR
//FILE2    DD  DSN=S2K.V1.EMPLOYE2,DISP=SHR
//FILE3    DD  DSN=S2K.V1.EMPLOYE3,DISP=SHR
//FILE4    DD  DSN=S2K.V1.EMPLOYE4,DISP=SHR
//REPORT   DD  SYSOUT=A
//ERRORS   DD  SYSOUT=A
//DUMPFILE DD  SYSOUT=A
```

The REPORT file contains a summary of indexes that are contained in Files 2 and 4 by component number. The File 2 summary shows the pages in use, by level, and the number of distinct values that are unique or that occur multiple times. The File 4 summary lists the CENTS and HLI blocks that contain the indexes by block sizes. File 4 shows how many of each size are in use. Output 15.3 shows a sample of the REPORT file.

Output 15.3 REPORT File

```
DBNAME = EMPLOYEE
  ITEM  FILE 2 PAGES       DISTINCT VALUES            FILE 4 CENTS             FILE 4 HLI
NUMBER LEVEL    IN USE      UNIQUE    MULTIPLE      BLOCK SIZE      IN USE  BLOCK SIZE   IN USE
------ ----- --------- ----------- -----------     ---------- ----------- ---------- ----------

    C1   3           1
         2           6
         1          38
         0         260       1,300           0
    TOTAL OCCURRENCES:                   1,300

    C2   1           1                                63 -  126          26   31 -   62
         0           6           0          27      PAGE-SIZE           2
    TOTAL OCCURRENCES:                   1,300

    C3 FILE 2 ERRORS DETECTED --- SEE ERRORS
    C4                                            FILE 4 ERRORS DETECTED --- SEE ERRORS
```

In the sample REPORT file , C1 has 1,300 distinct values that occupy 305 pages in File 2. The highest level page in File 2 for C1 is at level 3, there are 6 level-2 pages, 38 level-1 pages and 260 level-0 pages. All the values for C1 are unique, so there are no File 4 CENTS blocks for C1.

C2 has 27 distinct values that occupy 7 pages in File 2. The highest level page in File 2 for C2 is at level 1, and there are 6 level-0 pages. All 27 distinct values for C2 occur multiple times, and these multiple occurrences are stored in File 4. The REPORT files shows there are 2 page-size blocks and 27 non-page-size blocks in use. One HLI non-page-size block is in use.

C3 has errors in File 2. The message SEE ERRORS in the REPORT file refers to the ERRORS file.

C4 has errors in File 4. The message SEE ERRORS in the REPORT file refers to the ERRORS file.

Output 15.4 is a sample of the ERRORS file.

Output 15.4 ERRORS File

```
DBNAME = EMPLOYEE
       FILE 2  FILE 4  FILE 4
  ITEM PAGE IN PAGE IN OFFSET
NUMBER DECIMAL DECIMAL IN HEX ERROR DESCRIPTION                                  ITEM VALUE
------ ------- ------- ------ ---------------------------------------------- ----------------
    C3     313                MISMATCH WITH INDEX VALUE ON PAGE      314        B310
    C4                 0  0104  LOC6 LESS THAN OR EQUAL TO PREVIOUS LOC6          D400
```

In the sample ERRORS file, C3 shows an error in File 2. The description states that there is a mismatch with the index value. Page 313 in File 2 contains 5 distinct values for C3 and the last value on page 313 is C310, but the index on page 314 shows that the last value on page 313 is B310.

C4 shows an error in File 4. The description indicates that the LOC6 values are out of order. File 4 contains all of the File 5 pointers for each distinct value that occurs multiple times. The File 5 pointers are referred to as LOC6 values. The block that contains the error is on the first page in File 4 at hex offset 0104. The distinct C4 value in question is D400.

Whenever an error is detected in File 2 or 4 a snap dump of the pages in question is produced. The snap dump is written to DUMPFILE. A sample of DUMPFILE is shown in Output 15.5.

Output 15.5 DUMPFILE: Page 1 of File 2

```
JOB RPRT121    STEP STEP1     TIME 122319   DATE 99034       CPUID = 5D2055999672   PAGE 00000001

  PSW AT ENTRY TO SNAP    078D1000  0000AC4A  ILC  02  INTC  0033

  REGISTERS AT ENTRY TO SNAP

  FLOATING POINT REGISTER VALUES
      0-6  00000000 00000000    00000000 00000000    00000000 00000000    00000000 00000000

  GPR VALUES
      0-3  00000002  A000AC14  00009291  000168DF
      4-7  00000139  00016100  00009D95  0000B79C
      8-11 000007E0  00009FE8  00008FE8  00007FE8
     12-15 0000ABC0  0000AC74  0000AC74  0000ABC0

  ACCESS REGISTER VALUES
      0-3  0C0FC288  00000000  00000000  00000000
      4-7  00000000  00000000  00000000  00000000
      8-11 00000000  00000000  00000000  00000000
     12-15 00000000  00000000  00000000  00000000

-STORAGE
00016100 00000000 0009A5DE 80000E17 C3F3F0F6    00000000 00000000 00000000 00000000 *....C306..*
00016120 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000 *.........*
      LINES 00016140-000161E0  SAME AS ABOVE
00016200 00000000 00008000 0E18C3F3 F0F70000    00000000 00000000 00000000 00000000 *....C307..*
00016220 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000 *.........*
      LINES 00016240-000162E0  SAME AS ABOVE
00016300 00000000 80000E19 C3F3F0F8 00000000    00000000 00000000 00000000 00000000 *....C308..*
00016320 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000 *.........*
      LINES 00016340-000163E0  SAME AS ABOVE
00016400 00008000 0E1AC3F3 F0F90000 00000000    00000000 00000000 00000000 00000000 *....C309..*
00016420 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000 *.........*
      LINES 00016440-000164E0  SAME AS ABOVE
00016500 80000E1B C3F3F1F0 00000000 00000000    00000000 00000000 00000000 00000000 *....C310..*
00016520 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000 *.........*
      LINES 00016540-000165C0  SAME AS ABOVE
```

The DUMPFILE contains the pages of Files 2 and 4 that contained errors. At the top of each page is the header GPR VALUES followed by the values in Registers 0 through 15. The value in GPR 0 is 00000002, which indicates that this is a snap dump from File 2. GPR 4 = 00000139, which is the page number in hex, and GPR 8 = 000007E0, which is the page size. Under the heading -STORAGE is the actual data from the database file. So, this first page in the DUMPFILE is from File 2, page number hex 139 (decimal 313). The actual data is listed in hex in groups of 4 bytes each. To the right of the data is the same data displayed in character format. Without detailing the format of File 2 entries, we see the values contained on this page are C306 through C310. The next page in the DUMPFILE is shown in Output 15.6.

Output 15.6 DUMPFILE: Page 13A from File 2

```
JOB RPRT121     STEP STEP1     TIME 122319   DATE 99034      CPUID = 5D2055999672   PAGE 00000001

  PSW AT ENTRY TO SNAP    078D1000  0000AC4A  ILC  02  INTC  0033

  REGISTERS AT ENTRY TO SNAP

  FLOATING POINT REGISTER VALUES
      0-6  00000000 00000000    00000000 00000000    00000000 00000000    00000000 00000000

  GPR VALUES
      0-3  00000002  A000AC14  00009291  000160FF
      4-7  0000013A  00015920  00009D95  0000B79C
      8-11 000007E0  00009FE8  00008FE8  00007FE8
     12-15 0000ABC0  0000AC74  0000AC74  0000ABC0

  ACCESS REGISTER VALUES
      0-3  0C0FC288  00000000  00000000  00000000
      4-7  00000000  00000000  00000000  00000000
      8-11 00000000  00000000  00000000  00000000
     12-15 00000000  00000000  00000000  00000000

-STORAGE
00015920 00000001 0009AAC4 00099908 C3F3F0F5    00000000 00000000 00000000 00000000 *....C305...*
00015940 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000 *............*
      LINES 00015960-00015A00  SAME AS ABOVE
00015A20 00000000 00000009 A0E8C2F3 F1F00000    00000000 00000000 00000000 00000000 *....B310...*
00015A40 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000 *............*
      LINES 00015A60-00015B00  SAME AS ABOVE
```

GPR 0 = 00000002 indicates that this is a snap dump from File 2. GPR 4 = 0000013A is the page number in hex and GPR 8 = 000007E0 is the page size. This is a snap dump of File 2, page 13A (decimal 314), which contains a level-1 index page for C3. According to the level-1 index page, the last value for C3 in File 2 on page 139 should be B310. The previous snap dump of File 2 shows that the last value on page 139 is C310.

The last page in the DUMPFILE is a snap dump of a File 4 page, which is identified as having an error.

Output 15.7 DUMPFILE: Page 0 from File 4

```
JOB RPRT121     STEP STEP1      TIME 122319   DATE 99034     CPUID = 5D1055999672   PAGE 00000001

  PSW AT ENTRY TO SNAP    078D1000  0000AC4A  ILC  02  INTC  0033

  REGISTERS AT ENTRY TO SNAP

  FLOATING POINT REGISTER VALUES
      0-6  00000000 00000000    00000000 00000000    00000000 00000000    00000000 00000000

  GPR VALUES
      0-3  00000004  A000AC14  00009291  0001B81F
      4-7  00000000  0001B040  0000000F  0000B79C
      8-11 000007E0  00009FE8  00008FE8  00007FE8
     12-15 0000ABC0  0000AC74  0000AC74  0000ABC0

  ACCESS REGISTER VALUES
      0-3  0C0FC288  00000000  00000000  00000000
      4-7  00000000  00000000  00000000  00000000
      8-11 00000000  00000000  00000000  00000000
     12-15 00000000  00000000  00000000  00000000

-STORAGE
0001B040 00000000 003F0032 00000515 0000052F    00000549 00000563 0000057D 00000597 *...........*
0001B060 000005B1 000005CB 000005E5 000005FF    00000619 00000633 0000064D 00000667 *...........*
0001B080 00000681 0000069B 000006B5 000006CF    000006E9 00000703 0000071D 00000737 *...........*
0001B0A0 00000751 0000076B 00000785 0000079F    000007B9 000007D3 000007ED 00000807 *...........*
0001B0C0 00000821 0000083B 00000855 0000086F    00000889 000008A3 000008BD 000008D7 *...........*
0001B0E0 000008F1 0000090B 00000925 0000093F    00000959 00000973 0000098D 000009A7 *...........*
0001B100 000009C1 000009DB 000009F5 00000A0F    00000000 00000000 00000000 00000000 *...........*
0001B120 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000 *...........*
0001B140 00000000 003F0032 00000518 00000532    0000054C 00000566 00000580 0000059A *...........*
0001B160 000005B4 000005CE 000005E8 00000602    0000061C 00000636 00000650 0000066A *...........*
0001B180 FFFFFFF1 0000069E 000006B8 000006D2    000006EC 00000706 00000720 0000073A *...........*
0001B1A0 00000754 0000076E 00000788 000007A2    000007BC 000007D6 000007F0 0000080A *...........*
0001B1C0 00000824 0000083E 00000858 00000872    0000088C 000008A6 000008C0 000008DA *...........*
0001B1E0 000008F4 0000090E 00000928 00000942    0000095C 00000976 00000990 000009AA *...........*
0001B200 000009C4 000009DE 000009F8 00000A12    00000000 00000000 00000000 00000000 *...........*
```

GPR 0 = 00000004 indicates that this is a snap dump from File 4. GPR 4 = 00000000 is the page number and GPR 8 = 000007E0 is the page size. This is a snap dump of File 4, page 0 (database pages are numbered relative to 0), which contains a block of LOC6 values for C4 for the value D400. The ERRORS file message (Output 15.4) states that, at offset hex 0104 in File 4, on page 0, there was an incorrect LOC6 value. At offset hex 0104 in page 0 is a block capable of holding hex 3F LOC6 values, but it actually contains hex 32 LOC6 values. The LOC6 value at storage address 0001B180 is invalid.

DUMPXX

You can use the program DUMPXX (see the following sample JCL) to snap selected pages of a BDAM, a BSAM, a QSAM, or a VSAM file. You specify the beginning block number and the number of blocks to snap in a parameter in the EXEC statement. For example, PARM='A,5,10' specifies snap blocks from the file identified with the DDname A; begin the snap with block 5 and snap 10 blocks. A snap is taken after each selected page is read into memory. Consequently, each snapped block has the same beginning and ending address. General purpose registers snapped with each block show the block size and number. Register 0 shows the block size; Registers 5 through 9, 11, 14, and 15 show the block number.

Sample JCL

This job is in your Version 1 CNTL library as member JCLDUMP.

```
//JCLDUMP JOB (ACCTINFO),
//    S2K,REGION=0M
//*******************************************************
//*
//* THIS JOB WILL PRODUCE A SNAP DUMP OF A PAGE OR PAGES
//* OF A DISK FILE. IT IS ALL PARM DRIVEN WITH 3
//* PARAMETERS: PARM='DDNAME,FIRST PAGE,NUMBER OF PAGES'
//*
//* EXAMPLE TO DUMP FILE WITH THE DDNAME MYFILE
//* BEGINNING AT PAGE 25, DUMP 1 PAGE:
//*      PARM='MYFILE,25,1'
//*
//*******************************************************
//*
//DUMPFILE EXEC PGM=DUMPXX,PARM='MYFILE,25,1'
//STEPLIB  DD  DISP=SHR,DSN=S2K.V1.LOAD
//SYSPRINT DD  SYSOUT=A
//MYFILE   DD  DSN=S2K.V1.EMPLOYE6,DISP=SHR
```

F2BUILD Utility

The F2BUILD utility is designed to rebuild an existing database File 2. When your existing File 2 contains a lot of empty space due to page splitting, you can run F2BUILD to condense the key values in File 2 into full pages of data. You can also specify a padding factor to use when rebuilding the File 2 data. The padding factor represents the percentage of a page that is left empty for future key-value additions.

Empty space due to page splitting can be caused by many insertions of unique key values. The system recognizes when a File 2 page is full and splits that one page into two pages that are only half full. With many insertions of many key items, it doesn't take long to get a lot of usable space in File 2. The F2BUILD utility allows you to clean up this space without having to do a REMOVE INDEX/CREATE INDEX.

F2BUILD is a stand-alone utility that expects one parameter card. The parameter card defines the database you are working on and an optional padding factor. The format of the parameter card is

xxxxxxxxxxxxxxx [PAD=*nn*]

xxxxxxxxxxxxxxx is the database name to process against, and *nn* is the percent of padding to be left on each page.

If you want to rebuild File 2 of the EMPLOYEE database and allow 10% padding (90% of the page filled) on the new File 2 pages, use the following parameter card

EMPLOYEE PAD=10

If the PAD= option is omitted, 25% padding is assumed.

The input for the F2BUILD utility consists of four files: the parameter file, identified by the F2BPARM DDname; the database File 1, identified by the XXXXXXX1 DDname; the old (existing) database File 2, identified by the TEMPFIL2 DDname; and the new (to be built) database File 2, identified by the XXXXXXX2 DDname. The data set pointed to by the XXXXXXX2 DD statement must be an empty data set. If it is not empty, a return code 68 from the F2BUILD execution is issued.

The new File 2 data set does not have to be the same CISIZE as the old File 2 data set. F2BUILD totally rebuilds File 2 and, in that process, adjusts to whatever CISIZE the new File 2 indicates. After the F2BUILD utility has run successfully (return code 0), the old File 2 is no longer compatible with File 1 of the database. A one-page report is output from the successful execution of the F2BUILD utility. Here is a sample report.

```
F2BUILD REPORT FOR THE LIBRARY          DATABASE

TIME OF RUN: 14:35:00   DATE OF RUN: 99043

OLD FILE 2:   V1.LIBRARY2

NEW FILE 2:  V1NEW.LIBRARY2

OLD FILE 2 PAGE COUNTS: 000026   000149

NEW FILE 2 PAGE COUNTS: 000026   000149
```

Here are the return codes issued from the execution of the F2BUILD utility:

00	Successful completion of the F2BUILD utility
04	Parameter file OPEN error
08	Parameter file retrieval error
12	Database File 1 OPEN error
16	Database File 1 RPL generation error
20	Database File 1 GET error
24	Temp File 2 OPEN error
28	Temp File 2 RPL generation error
32	New File 2 OPEN error
36	New File 2 RPL generation error
40	Database File 1 CLOSE error
44	Database File 1 OPEN for output error
48	Database File 1 PUT error
52	Temp File 2 GET error
56	New File 2 PUT error
60	GETMAIN error
64	Index pointer problem in old File 2
68	The new File 2 is not an empty data set

Sample JCL

This job is in your Version 1 CNTL library as member JCLF2BLD.

```
//JCLF2BLD JOB (ACCTINFO),
//    S2K,REGION=0M
//*******************************************************
//*
//*  THIS JOB READS DATABASE FILES 1 AND 2 DIRECTLY
//*  AND PRODUCES OPTIMIZED FILE 2.
//*
//*******************************************************
//*
//F2BUILD  EXEC PGM=F2BUILD
//STEPLIB  DD  DISP=SHR,DSN=S2K.V1.LOAD
//SYSPRINT DD  SYSOUT=A
//FILE1    DD  DSN=S2K.V1.EMPLOYE1,DISP=SHR
//FILE2    DD  DSN=S2K.V1.EMPLOYE2,DISP=SHR
```

CFIND TABLE VALIDATION PROGRAM

Introduction

The main function of the CFIND utility program is to inspect a database Hierarchical Table (HT) for structural errors. The HT consists of CFIND entries, which contain the logical record structure of the database. In addition to inspection, this program can produce formatted listings of either a specific CFIND entry, a range of CFIND entries, or entries matching specified criteria. Also, an available space report and execution statistics are available. The program executes as a stand-alone utility program.

The CFIND program can detect several types of errors in the Hierarchical Table. It is not intended to fix extensive errors. If it detects structural errors, it reports only on five and writes the entry being investigated to the MESSAGES output file.

The CFIND validation program runs in any of these four modes:

- CHAIN mode: used to detect HT error
- PRINT mode: prints one HT entry or a range of HT entries
- SCAN mode: prints CFIND entries that match specified criteria
- PATCH mode: repairs a CFIND table entry.

Usually, you execute the CFIND validation program modes in the following order:

1. Determine if any damage exists by using CHAIN mode.
2. Diagnose the scope of the damage by using PRINT mode or SCAN mode.
3. Correct the damage by using PATCH mode.
4. Run CHAIN mode again to verify the success of the correction.

The process of checking each entry can be very time-consuming and often unnecessary. For example, if damage is detected and corrected in an entry in the middle of the CFIND table, you should (in subsequent runs) skip validation of entries that have already been checked. CHAIN, SCAN, and PRINT modes can specify a START parameter to begin checking at a specific HT entry. You can use the START parameter to bypass portions of the CFIND table that have been previously validated, as long as you perform one final chain check of the entire CFIND table after all other runs.

Sample CFIND Jobs

Sample CFIND JCL is specified here. Included is a brief synopsis of each output file and its purpose. The SYSIN statement in this example specifies CHAIN mode. Each CFIND mode concept topic contains a discussion of SYSIN parameters and DD statements that are necessary for a specific mode, report, or function.

Sample JCL

This job is in your Version 1 CNTL library as member JCLCFIND.

```
//JCLCFIND JOB (ACCTINFO),
//    S2K,REGION=0M
//*********************************************************
//*
//*  THIS JOB EXECUTES THE DATABASE FILE 5 VALIDATION
//*  PROGRAM.  IT HAS 4 MODES AND IS CONTROLLED WITH
//*  SYSIN INPUT.  THE 4 MODES ARE
//*        CHAIN
//*        PATCH
//*        PRINT
//*        SCAN
//*
//*  DEPENDING ON THE SIZE OF FILE 5 AND THE MODE
//*  YOU CHOOSE, THIS PROCESS CAN TAKE A LONG TIME TO RUN AND
//*  PRODUCE A LOT OF OUTPUT.
//*
//*  CONTACT YOUR TECHNICAL SUPPORT REPRESENTATIVE
//*  BEFORE YOU RUN THIS JOB.
//*
//*********************************************************
//*
//CFINDPR  PROC SYSOUT=A
//CFINDIT  EXEC PGM=CFIND,
//             PARM='LINE=80,BUFNO=500'
//STEPLIB  DD  DISP=SHR,DSN=S2K.V1.LOAD
//MESSAGES DD  SYSOUT=&SYSOUT
//FT09F001 DD  DUMMY <== QASTAT
//SYSPRINT DD  DUMMY <== PRINT/SCAN MODE POINTERS
//SYSUDUMP DD  DUMMY
//DISPLAY  DD  DUMMY <== "FILE#, PAGE#" : LOTS OF OUTPUT
//TOTALS   DD  DUMMY <== PRINT/CHAIN MODE
//*                     AVAILABLE SPACE REPORT
//         PEND
//*
//CFINDRUN EXEC CFINDPR
//*
//* DB FILES
//*
//CFINDIT.FILE1 DD DSN=S2K.V1.EMPLOYE1,DISP=SHR
//CFINDIT.FILE5 DD DSN=S2K.V1.EMPLOYE5,DISP=SHR
//CFINDIT.FILE6 DD DSN=S2K.V1.EMPLOYE6,DISP=SHR
```

```
//*
//* MODES
//*
//* ********** *
//* CHAIN MODE * DETECT HT ERRORS
//* ********** *
//* ENTRY, START AND STOP ARE OPTIONAL
//* FIELDS (UP/DOWN) NOT USED
//* MODE=CHAIN(22)
//* MODE=CHAIN,START=000000,STOP=FFFFFF
//*
//* ********** *
//* PATCH MODE * PATCH HT WITH ERRORS
//*            * DETECTED IN CHAIN MODE
//* ********** *
//* *** PATCH CANNOT USE START/STOP
//* MODE=PATCH,ENTRY=XXXXXXXX,DATA=0
//* MODE=PATCH,ENTRY=XXXXXXXX,DOWN=0
//* MODE=PATCH,ENTRY=XXXXXXXX,RIGHT=0
//* MODE=PATCH,ENTRY=XXXXXXXX,UP=0
//* MODE=PATCH,ENTRY=XXXXXXXX,COMP=0
//*
//* ********** *
//* PRINT MODE * PRINT HT ENTRY
//* ********** *
//* *** ENTRY, START AND STOP ARE OPTIONAL:
//* ***        FIELDS (UP/DOWN/RIGHT) NOT USED
//* MODE=PRINT,START=XXXXXXXX,STOP=XXXXXXXX
//* MODE=PRINT,START=XXXXXXXX,STOP=NEVER
//* MODE=PRINT,ENTRY=XXXXXXXX,START=XXXXXXXX,STOP=NEVER
//* MODE=PRINT,ENTRY=XXXXXXXX
//*
//* ********** *
//* SCAN MODE * SCAN FOR SPECIFIED POINTER VALUES
//* ********** *
//* *** ONLY ONE FIELD CAN BE SPECIFIED:
//* *** FIELDS: DATA OR DOWN OR RIGHT OR UP OR COMP
//* *** START AND STOP ARE OPTIONAL
//* MODE=SCAN,START=XXXXXXXX,STOP=XXXXXXXX
//* MODE=SCAN,DATA=XXXXXXXX
//* MODE=SCAN,DOWN=XXXXXXXX
//* MODE=SCAN,RIGHT=XXXXXXXX
//* MODE=SCAN,UP=XXXXXXXX
//* MODE=SCAN,COMP=XXXXXXXX
//*
//* ************************** *
//* MODE PARAMETERS FOLLOW SYSIN *
//* ************************** *
//*
//CFINDIT.SYSIN DD *
//* MODE=CHAIN(22)
/*
```

CHAIN Mode Concepts

Use CHAIN mode to detect errors in a database Hierarchical Table. Options allow you to display entries as they are scanned, and to validate either re-usable or used space, or both. Output is written to the DISPLAY file. The page number for each page that is real (physical I/O) is also written to the DISPLAY file. Consequently, output might be large. Unless you are using START and STOP parameters and are interested in what is written to DISPLAY file, you should omit the DISPLAY JCL statement.

Available options are specified by using two numeric characters:

MODE=CHAIN(*mn*)
m=0, 1, or 2.
n =0, 1, 2, or 3.

The meanings of *m* and *n* for CHAIN mode are

First Numeric Character (*m*)	Second Numeric Character (*n*)
0 - no displays (default)	0 - validate used entries only (default)
1 - display current entries	1 - validate re-usable space only
2 - display current and related entries.	2 - validate used and re-usable space
	3 – validate level-0 used entries only, starting at the last entry and chaining left to the first entry.

The first numeric character (*m*) controls what is written to the DISPLAY output file, and determines the level of detail that is displayed. Each entry can be displayed as it is validated. Also, all of its immediate relatives can be displayed because they are read one-by-one for inspection of back-pointers to the current entry. CUR is printed below the current entry if the first option character is 2, and a blank line is printed below its relatives. This option is primarily intended for debugging the program itself because it tends to generate impractical volumes of output. The second numeric character (*n*) controls the type of entries to be validated and determines whether the CFIND program should validate used space, re-usable space, or both.

Note: When validating re-usable space, no message is issued indicating whether any re-usable space exists. To determine if there are any records in the re-usable space chain, you must include the TOTALS file. For details see "Statistical Space Reports" later in this chapter.

Several possible forms for specifying CHAIN mode options follow. MODE=CHAIN defaults to MODE=CHAIN(00), which means chain-checking is performed on used space only and no debug trace print is provided even if a DISPLAY file is specified.

The first digit controls the level of debug trace output written to the DISPLAY file if specified in the JCL. It is optional.

CHAIN(0X) - No entries are displayed because they are chained.
CHAIN(1X) - Only the current entry is displayed.
CHAIN(2X) - Both the current and related entries are displayed.

The second digit controls the type of processing performed.

CHAIN(X0) - Chain-check used space only.
CHAIN(X1) - Chain-check re-usable space only.
CHAIN(X2) - Process both used and re-usable space.
CHAIN(X3) - Chain-check level 0 up (left) pointers.

If the program detects an error, it issues a user abend and writes a message to the MESSAGES file that includes the last two entries that are being investigated. (See "User Abend Codes" later in this chapter) If no error is detected, the following message is written to the MESSAGES file:

** NO HT ERRORS DETECTED

The program validates each entry to ensure that there are no internal errors, for example, that no entry points to itself. In addition, the program ensures that no record is encountered an inappropriate number of times. Also, various other tests check for loops or breaks in the tree structure. After validating used entries, the program validates re-usable space if requested.

If an error is detected and the error maximum (set to 5, by default) has not been reached, the program sends a user abend completion code to the MESSAGES file. Then, the program displays the last two entries it was checking when the abend occurred. You can usually determine the cause of the error by reading the description of the user abend code in the section "User Abend Codes" and inspecting the two displayed entries. Usually, a storage dump is not needed, but one is produced if you include a SYSUDUMP output file.

The program resumes checking by searching for the next RIGHT-pointer or UP-pointer, whichever is appropriate; it continues this re-start process after each abend until the error maximum is reached. The error maximum is checked to avoid loops that can be introduced by continually re-starting at the same place. When the error maximum is reached, User Abend Code 49 is issued, and a storage dump is produced. By studying the dump, you can usually limit further investigation to a small part of the CFIND table. Also, the dump shows the actual contents of the bad CFIND page, which is a prerequisite to patching it. (See "Areas in a CFIND Memory Dump" later in this chapter for data offsets in a CFIND dump.) User Abend Codes 256 and higher cause immediate termination with a dump, and no re-start is attempted.

PRINT Mode Concepts

PRINT mode ignores the tree structure and prints records in sequential order. No validity check is performed. You can limit the report to a specific portion of the CFIND table by specifying START and STOP entry numbers in the SYSIN input or in the PARM field in the EXEC statement. Here are examples of PRINT mode SYSIN specifications.

MODE=PRINT
MODE=PRINT,START=123
MODE=PRINT,START=123,STOP=NEVER
MODE=PRINT,START=123,STOP=456

For PRINT mode, you must define the SYSPRINT file. If you do not specify the START or STOP parameters, the entire CFIND table is formatted and displayed, beginning at entry 0 and ending with the entry indicated in ENDTAB(6). ENDTAB(6) is a record counter in the ID table (database File 1), which indicates the end of the CFIND table.

The START and STOP parameters cause the program to start with the specified START entry and print a report for entries through and including the specified STOP entry. The START and STOP fields must be hexadecimal numeric characters; otherwise, an error message appears on the report and the run is terminated. If the STOP field is lower than the START field, the CFIND table is wrapped around in the order indicated. That is, it is formatted and displayed from the specified START field to the end, then, from the beginning to the specified STOP field. The START and STOP fields are handled as optional keyword parameters. Either or both of them can be omitted. However, if the STOP field is coded as STOP=NEVER, the program processes to the end of database File 5 (extent boundary for File 5).

Include the TOTALS output file to create a report of the number of entries in the CFIND table, subtotaled by record type. Sample details are specified in "Statistical Space Reports."

SCAN Mode Concepts

SCAN mode reads the CFIND table sequentially and writes records to the SYSPRINT file. SCAN mode recognizes only one field parameter, and you must specify a field parameter (DATA, DOWN, RIGHT, UP, COMP) to select records. For example, MODE=SCAN,UP=1 prints those records whose UP pointer equals 1. The entire CFIND table is scanned, but only records that meet the selection criteria are displayed. If you do not want to scan the whole file, you can use the START and STOP parameters to limit the area of the file that is scanned. If you specify more than one parameter, only the last one is recognized.

SCAN mode can be most useful when isolating damage in the Hierarchical Table. For example, To print every child of entry 273, the SYSIN parameter would be

MODE=SCAN,UP=273

This produces a SYSPRINT display that contains every entry that has an UP pointer (parent) of 273.

PATCH Mode Concepts

The CFIND program allows you to patch fields in a CFIND table entry. If an error is detected when using CHAIN mode and the error is possibly isolated by using the PRINT and/or SCAN mode, you can use PATCH mode to correct it. Using parameters, you can correct one field in one entry at a time. For example, if you determine that entry 245 should have a RIGHT pointer of 246, you can specify

MODE=PATCH,ENTRY=245,RIGHT=246

Note: Specifying RIGHT=0 would cause a break in the sibling chain because 0 indicates the end of a chain. (Using RIGHT=0 can be useful in pruning damaged sibling chains.)

PATCH mode requires that you specify the following

- MODE=PATCH (to indicate that a patch should be made).
- ENTRY=*entry-number*, (where *entry-number* is a 1- to 8-digit hexadecimal number that represents the entry record to be patched. This entry must be within the used space; otherwise, an error message appears).
- One of the following field parameters DATA=, DOWN=, RIGHT=, UP=, or COMP=. The value supplied is a 1- to 8-digit hexadecimal number. The parameter that you specify indicates the CFIND field to be patched and the new value for it. If you specify more than one of these five parameters in a patch operation, the last parameter that was entered is processed. If several fields must be patched in the same entry, a patch operation must be performed for each field. The MESSAGES file lists the contents of the record before and after the patch.

Input Parameters and Output Files

The following describes the filenames shown in sample JCL.

Table 15.1 Filenames in Sample JCL

Filenames	Descriptions
STEPLIB	This statement identifies the library that contains the CFIND validation program.
EXECUTION PARM	The execution parameter 'LINK=80,BUFNO=500' is sufficient for most needs. For a detailed discussion of the JCL EXEC parameters, see the section "Specifying Parameters" later in this chapter.
MESSAGES	This report file contains the start and stop times for the CFIND jobs, an echo of the EXEC parameters, and the mode parameter, which is read from SYSIN. If any errors are detected by using CHAIN mode, a user abend is issued to explain the detected error and the last two entries are printed.
FT09F001	Including this file executes QASTAT and gives timing statistics and the I/O count for the CFIND run. This report is useful if you are attempting to increase the execution speed of the CFIND program. The QASTAT utility (documented earlier in this chapter) is called by the CFIND program if you include FT09F001. CFIND invokes QASTAT at the beginning and at the end of processing. By subtracting the earlier CPU and I/O figures from later ones, you can determine actual CPU time and I/O counts. QASTAT itself uses a trivial amount of CPU time to handle the calls from the program.
SYSPRINT	A table of selected PRINT mode and SCAN mode entries is printed in this report. It includes the entry number, DOWN pointer, RIGHT pointer, UP pointer, entry level, internal component number, and a DATA RECORD pointer.
SYSUDUMP	By including this file, a dump is printed when a structural error is encountered. Dump fields are described in "Areas in a CFIND Memory Dump".
TOTALS	This file reports record counts by record type.
DISPLAY	This report prints the file number and the page number of every READ. It can create a lengthy display. CHAIN mode can control the output to this file by using an additional CHAIN mode parameter.

Database File Specifications

The filename FILE1 must allocate File 1 of the database that is being checked, and FILE5 must allocate the CFIND table. Failure to allocate these files correctly causes unpredictable results. If either File 5 or File 1 cannot be opened, the program abends with User Abend Code 256 or 257, respectively. This is probably a user error.

- **FILE1**: File 1 of the database that is being checked.
- **FILE5**: File 5 of the database that is being checked.
- **FILE6**: File 6 of the database that is being checked.

To dump the pertinent CDATA pages when a CHAIN mode verification error occurs, include a FILE6 and the SYSUDUMP filename.

SYSPRINT Output

PRINT mode displays the seven structural parts of each CFIND table entry. An internal component number of (MAXCOM + 1) means the entry is part of a re-usable space chain. The following example shows part of a typical report.

PRINT Mode SYSPRINT Display

ENTRY NO	273	274	275	276	277
DOWN	274	275	276	277	278
RIGHT	302	288	0	282	0
UP	272	273	274	275	276
LEVEL	1	2	3	4	4
INT COMP #	1D	1F	21	23	2F
DATA RECORD	7D2F	7D3F	7D4F	7D5F	7D6F

The number of lines in the report is about two-thirds the number of entries displayed, with a minimum of seven lines. Eighty entries per full listing page are displayed for LINE=132. The number of pages in the report is about three times the number of CFIND pages in use for 4096-byte pages.

TOTALS Output

The PRINT and SCAN Modes write a record count, by record C number (such as, C0), to the TOTALS file. (SYSTEM 2000 does not report re-usable space information.) For example,

Print Mode TOTALS Output

***** SEQUENTIAL PASS TOTAL RECORD REPORT *****

DATABASE NAME IS PROBLEM

RECORD NUMBER	TOTAL
C0	2218
C20	4364
C30	12382
C40	1865
C50	4737
C60	0

TOTAL AVAILABLE SPACE ENTRIES: 2

The SYSPRINT report can be dummied if you want only the TOTALS output. The program detects the dummy specification and bypasses all of the print-line formatting for the SYSPRINT report.

Because PRINT mode is a sequential process, the buffers obtained by using BUFNO are bypassed. A separate buffer is used for PRINT and SCAN modes.

Specifying Parameters

You can specify parameters for the CFIND program in the SYSIN file, as program parameters (in the JCL EXEC statement), or both. SYSIN is a physical sequential file that has a logical record length of 80 and a block size of 80. It does not matter where the parameters are specified, the format is the same. The following general rules apply:

- All parameters have defaults.
- The SYSIN file can be used to control multiple executions of the programs. Each 80-character SYSIN record represents a set of parameters for one execution. Parameters cannot exceed one line.
- The parameters specified in the JCL EXEC statement are saved in the program and merged with the SYSIN parameters each time a SYSIN record is read. For example, the number of buffers can be specified in the EXEC statement, and, if you specify the MODE, it will change the default to print to whatever you specify. The program default is MODE=CHAIN.
- You specify parameters by keywords unless otherwise indicated. Decimal numeric values contain from 1- to 5- decimal digits. Hexadecimal values contain from 1- to 8-hexadecimal digits. Use a comma or one or more blanks to separate the parameters in SYSIN records or in the EXEC statement.
- LINE and BUFNO are executed only when specified as program parameters or in the first SYSIN record. This is because obtaining buffers with BUFNO and opening the SYSPRINT file (MODE=PRINT) are performed only one time in the program, at initialization. Reading the EXEC statement and reading the first SYSIN statement constitute the first pass. If BUFNO and LINE are not specified as program parameters in the first pass, or if any of these parameters are in error, the program uses the appropriate defaults. After the first parameter pass, specifying LINE and BUFNO does not cause an error, but the parameters are ignored.
- If a SYSIN record does not contain a MODE specification, the program uses the mode specified in the EXEC statement or the default mode CHAIN.
- If there are errors after any pass through the parameters, the values for all parameters (except LINE and BUFNO in the first pass) are ignored.
- While executing the CFIND program at a terminal, pressing the carriage return without entering input causes batch-fatal message 017.

Parameter Examples

JCL EXEC Statement

PARM='BUFNO=150, MODE=PRINT, START=A'
PARM='BUFNO=0009, MODE=CHAIN, START=12, STOP=002A'
PARM='MODE=PRINT, START=60AB, STOP=006FFF, LINE=132'

SYSIN Statement

BUFNO='150, LINE=80, MODE=PRINT, START=A, STOP=0000C
ENTRY=554, MODE=PRINT
ENTRY=B1, MODE=CHAIN

BUFNO Parameter

The BUFNO= parameter indicates the number of CFIND buffers to GETMAIN.

BUFNO=*nnnnn*

nnnnn is a decimal number in the range of 1 to 65,535 The default is 150.

The buffers are above 16 megabytes. Specifying too few buffers increases I/O CPU time and elapsed time. Specifying too many buffers is a waste of memory, but has little effect on job performance. You might want 500 to 1000 buffers (or more) when using CHAIN mode. Memory requirements depend on the BUFNO parameter and is specified in bytes by the formula BUFNO*(12+PAGESIZE).

A buffer manager allows File 5 pages to reside in the allocated buffers. When a different page is needed and there are no unused buffers, the "oldest" (the one with the earliest time stamp) buffer is re-allocated.

Each buffer has a 12-byte directory entry. The directory entry contains a one-word page number (-1 if unused), a one-word "clock" value that indicates how recently the buffer was referenced, and a one-word address of the buffer. The buffer directory appears at the beginning of the memory acquired for buffers.

START Parameter

The START= parameter specifies the entry number at which processing should begin.

START=*hhhhhhhh*

hhhhhhhh is a 1- to 8-digit hexadecimal CFIND entry number.

This parameter can be specified with the CHAIN, SCAN, and PRINT modes. Use the START parameter with or without the STOP parameter. START is specified outside of the used space, an error message appears, and the program terminates. If the entry that is specified is a re-usable entry, the program terminates with an error.

In PRINT mode and SCAN mode, use the START parameter to limit printing. You can specify an entry that occurs anywhere within the physical limits of the file. If the record entry number that is specified is for unused space, a warning message appears. If the entry is beyond the end of the file, an appropriate message appears and the request is ignored.

STOP Parameter

The STOP= parameter identifies the entry number at which processing should stop.

STOP=*hhhhhhhh*

hhhhhhhh is a 1- to 8-digit hexadecimal CFIND entry number. You can use the STOP parameter with or without the START parameter.

In PRINT mode and SCAN mode, STOP must be specified within the physical limits of the file. If the STOP value is less than the START value, wrap-around occurs. For PRINT mode only, STOP=NEVER requests printing to the end of the physical file.

In CHAIN mode, STOP indicates which used entry stops the chain-checking. If the STOP parameter is not found, processing ends with the last logical entry in the file. The STOP parameter has no effect on re-usable space chain-checking. For example, if you specify,

START=5, STOP=A, MODE=CHAIN(02)

the program starts chain-checking at entry 5 and continues until entry A is found. This process assumes it is a used entry that occurs logically, after entry 5. All re-usable space entries of all record types are chain-checked. Remember that it is possible in CHAIN mode to have a STOP value that is lower than the START value, but the STOP value is placed logically, later in the file.

ENTRY Parameter

Specifying the ENTRY= parameter is equivalent to specifying both the START and STOP parameters, that is, ENTRY=6BA is equivalent to START=6BA, STOP=6BA.

ENTRY=*hhhhhhhh*

hhhhhhhh is the 1- to 8-digit hexadecimal number of the entry to be processed. The ENTRY= parameter can be specified with all modes, however, ENTRY is *required* for MODE=PATCH to specify which entry is to be patched.

LINE Parameter

The LINE= parameter value is the record length written to the SYSPRINT file, which is the output created when MODE=PRINT is specified.

LINE=*nnnnn*

nnnnn is a 1- to 5-digit decimal number that has three valid values: 80, 120, and 132. The default is 80.

Field Parameters

Field parameters are optional for some modes, but one field parameter is *required* for the PATCH and SCAN modes.

DATA=*hhhhhhhh*
DOWN=*hhhhhhhh*
RIGHT=*hhhhhhhh*
UP=*hhhhhhhh*
COMP=*hhhhhhhh*

hhhhhhhh is a 1- to 8-digit hexadecimal number that specifies a value to be patched or scanned. In PATCH mode, one field parameter must be specified when you specify the ENTRY parameter to indicate which entry is to be patched. In SCAN mode, one field parameter is *required* to indicate the scan selection criteria.

END Parameter

The END parameter terminates the program.

```
END
```

The END parameter followed by a space forces an end-of-file in SYSIN and terminates the program.

Summary of Parameters

Table 15.2 contains a summary of parameters.

Table 15.2 Summary of Parameters

Modes	Option Spec	ENTRY	Field Parameter	START	STOP	BUFNO	LINE
CHAIN	optional	optional	n/a*	optional	optional	optional	optional
PRINT	invalid**	optional	n/a*	optional	optional	optional	optional
SCAN	invalid**	n/a*	required	optional	optional	optional	optional
PATCH	invalid**	required	required	n/a*	n/a*	optional	optional
END	n/a*	n/a*	n/a*	n/a*	n/a*	n/a*	n/a*

* Does not operate for this mode.
** Invalid for this mode; causes an error if specified.

Areas in a CFIND Memory Dump

Table 15.3 Hex Displacement into Program (CFINDCH)

V1	R12.1	Contents
1C9C	1818	7-word table that contains (1) current entry being checked (2) entry DOWN pointer (3) entry RIGHT pointer (4) entry UP pointer (5) entry LEVEL (Note: If this is a removed record, this will always be 0) (6) entry component numbers in DESCRIBE order, all in hexadecimal (Note: If this is a removed record, the component number will be equal to the maximum component number) (7) entry byte displacement of CDATA record (CURRENT in routine CFINDCH).
1CB8	1834	7-word table of same structure as preceding one. This table is used to hold immediate relatives of the current entry while their relationship is being validated (UPDENT in routine CFINDCH).
1A58	1528	Fullword that contains the address of the buffer directory (ABFDIR in routine CFINDCH).
1CFC	1878	Fullword that contains the address of the CDATA page for the current entry; 0 if unused (CUR6BUFA in routine CFINDCH).
1CF8	1874	Fullword that contains the address of the CDATA page for the relative of the current entry; 0 if unused (UPD6BUFA in routine CFINDCH).
1A48	1518	Fullword that contains the address of the ID table (IDBUF in routine CFINDCH).
1A4C	151C	Fullword that contains the address of CDEFIN (KDEFIN in routine CFINDCH).
1840	12F8	Fullword that contains the address of the CFIND parameter control block (V(CFPARMS) in CFINDCH).

Statistical Space Reports

You can produce useful reports on used space, as previously shown, and on re-usable space. Statistical space reports can be more efficiently obtained by using the EXAMINE and RCHAIN utilities. However, these reports can also be obtained by the CFIND utility when you

- Modify the sample CFIND JCL to run MODE=CHAIN(01) and include a TOTALS file to produce a report for re-usable entries. The total number of re-usable space entries for each record type is summed and displayed in the TOTALS file.

- Modify the sample CFIND JCL to run MODE=PRINT and include a TOTALS file to inexpensively produce a report for used entries (as opposed to re-usable space entries).

- Combine the CHAIN and PRINT modes statistical displays to provide an inexpensive analysis of CFIND space utilization and a general overview of the distribution of data records in the database. This is a way to discover used and re-usable space information on your database.

Re-Usable Space Report

***** CHAIN-CHECK AVAILABLE SPACE REPORT *****

DATA BASE NAME IS EMPLOYEE

C0	1
C100	12
C110	36
C120	55
C130	5
C200	0
C300	7
C400	5
C410	6
C420	0

User Abend Codes

User abend codes can occur when executing the CFIND program. The descriptions in this section provide a convenient reference for technical support personnel. Completion codes correspond to the following errors:

Code	Description
1	Entry 0 does not have CPDOWN=0.
2	Entry 0 does not have CPLOC=0.
3	Entry 0 does not have CRGNUM=0.
4	Entry 0 does not have CLITER=0.
5	The entry has CPUP=0, but it is not the first logical entry as indicated by entry 0.
6	The first logical entry does not have LEVEL equal to 0.
7	The current level 0 entry has a CPUP pointer to an entry not at level 0.
8	The current level 0 entry has a CPUP pointer to an entry at level 0 that does not have a CPRITE pointer back to the current entry.
9	The current entry at level N, not at level 0, has a CPUP pointer to an entry that is not at level N-1.
10	The current entry at level N, not at level 0, has a CPUP pointer to an entry at level N-1, but the downward chain from the higher level was followed to its end without encountering the current entry.
11	The current entry at level N has a CPDOWN pointer to an entry not at level N+1.
12	The current entry at level N has a CPRITE pointer to an entry not at level N.

Code	Description
13	The current entry at level N, not at level 0, has a CPRITE pointer to an entry at level N, but the two entries have different CPUP pointers.
14	Chaining left, the last logical entry pointed to by entry 0 has LEVEL not 0. This can also be issued for first entry checked if START is specified.
15	The current entry at level N, not at level 0, has CPUP=0.
16	The current entry has a CPDOWN pointer to an entry that does not have a CPUP pointer back to the current entry.
17	Logical entry 0 has CPUP=0.
18	Logical entry 0 has CPRITE=0.
19	The current entry has CRGNUM pointing to a CDEFIN entry that is not for a schema record.
20	The current entry has LEVEL=0 and CPRITE=0, but it is not the last logical entry as indicated by entry 0.
21	The level contained in the CDEFIN entry for this schema record is greater than 32. Register 9 points to the bad CDEFIN entry.
22	The current entry has CPUP=CPDOWN. This would be correct if it were the first level 0 CFIND entry and had no descendants so that CPUP=CPDOWN=0, but this is not the case.
23	"Siamese Twins": The current entry has a sibling somewhere to its right that has a CPRITE pointer back to the current entry.
24	The number of records scanned exceeds database record count (ENDTAB6). This indicates a loop.
25	The current entry has RGNUM greater than indicated as the total in CDEFIN(1).
26	Chaining left, the last logical entry pointed to by entry 0 has CPRITE not equal to 0.
27	The current level 0 entry has a CPRITE pointer to a level 0 entry whose CPUP pointer does not point back to the current entry.
28	The current entry erroneously has CPUP=CPRITE; LEVEL is not equal to 0.
29	Same as 28, but LEVEL=0.
30	The current entry has a CPDOWN pointer to itself.
31	The current entry has a CPRITE pointer to itself.
32	The current entry has a CPUP pointer to itself.
33	The current entry has a CPDOWN not equal to 0.
34	The number of CDEFIN entries indicated by CDEFIN(1) exceeds MAXCOMP in ENDTAB(8).

Code	Description
35	The current entry number exceeds ENDTAB(6).
36	The last logical entry (level 0 CFIND entry) was reached while chaining up from a descendant without first reaching it from the left. This indicates probable errors in the last two entries.
37	The last logical entry (level 0 CFIND entry) was reached twice from the left. The error is at level 0.
38	The current entry in a re-usable space chain points to an entry at a different level.
39	"Siamese Twins" pointers are in a re-usable space chain.
40	The current entry in a re-usable space chain has CRGNUM not equal to (ENDTAB(6)+1). In this situation, SYSTEM 2000 software issues SYSTEM 2000 Error Code 235.
41	The total number of re-usable and used entries does not equal the count of all entries contained in ENDTAB(6).
42	The mode bit is set in the CDEFIN table, indicating that the record is a CALC record, but the SYSTEM 2000 release is prior to Release 10.0.
43	CAREC (L0C0, internal component number) in the CALC definition is not consistent with L0C0 in the Definition Table.
44	The number of CALC records detected in the CDEFIN table by checking the mode bit is not the same as CALCNO.
45	CALOC (byte displacement address) to get the first page in a LOC space bucket is not on a page boundary.
46	BKTOVP pointer (byte displacement address) to an overflow page is not on a page boundary.
47	CALC block has CALOC(1)=0.
48	BKTOVP points to the beginning of a page, but it is an overflow page. It should never point to the beginning.
49	The error maximum has been reached.
50	Chaining left, the right pointer of the current entry does not point back to the latest entry.
51	Chaining left, a record was encountered that was not at level 0.
52	A BKTOVP pointer on a primary page is not on a page boundary.
53	The CALC pointer to the next CALC record points to a first-level overflow page, but it is not the same page indicated by the BKTOVP pointer on the primary page.
54	The File 6 data address of the CALC record is not the same as the File 6 data address (CPLOC) contained in the CFIND record.
55	The CFIND record chained to by a CALC record contains an internal component number (CRGNUM) not equal to the internal component number (CAREC) in the CALC block.

Code	Description
56	The internal component number (CRGNUM) in the CFIND record chain to a CALC record is the same as the internal component number (CAREC) associated with the CALC record, but it is not a valid component number. That is, it is either greater than MAXCOMP or equal to MAXCOMP and, therefore, is flagged as removed.
57	A primary page has a level number (BKTLEV) that indicates it is not a primary page.
58	There is an invalid level (BKTLEV) number.
59	Chaining left, the last logical entry is reached (CPUP=0), but it is not the first as indicated by entry 0.
60	A data pointer (CPLOC) is encountered, but it is past the last used data record in the CDATA table. This probably shows up as SYSTEM 2000 Error Code 247.
256	The FILE5 open failed.
257	The FILE1 open failed.
258	The FILE6 open failed.
259	FILE1, FILE5, or FILE6 has an invalid SYSTEM 2000 page size.
260	An I/O error was detected reading File 1, 5, or 6 or while writing File 5 in PATCH mode. This abend is accompanied by a WTO that contains SYNAD information about the I/O error.
261	The first option byte in the MODE=CHAIN parameter is invalid.
262	The release number in File 1 indicates that the database was last accessed by a SYSTEM 2000 release incompatible with this version of the CFIND program.
263	The second option byte in the MODE=CHAIN parameter is invalid.
264	The release fields in File 1 are different, which indicates that a Release 10.0 conversion attempt failed.
265	The START value specified by using MODE=CHAIN points to re-usable space.
266	The START value specified by using MODE=CHAIN points past the used space.
267	The STOP value specified by using MODE=CHAIN points to re-usable space.
268	The STOP value specified by using MODE=CHAIN points past the used space.
269	The mode specified is invalid.
270	The first option byte specified by using MODE=CALC is invalid.
271	An invalid patch specification occurred. This is probably a problem in the CFIND program, not a user error. (CFFIELD in CFPARM in routine CFINDIN, the CFIND parameter block, is invalid.) It is issued in routine CFINDFX just after being called by CFINDCH.

Code	Description
272	The second option byte specified by using MODE=CALC is invalid.
273	An attempt was made to write a line longer than 80 characters to the MESSAGES file. (Issued in routine CFINDIN and not a user error.) An R2 before the abend contains the beginning address of the line to be written.
274	An entry specification is invalid; probably not caused by the user. CFSTART and CFSTOP are not equal to the current entry. (Check mode in routine CFINDFX).
275	GETMAIN failure

CFIND Messages

This section describes WTO's written to the system and messages written to the MESSAGES file in response to input in the EXEC statement or the SYSIN file or in response to processing performed by the program. These messages are all written in the same format, as shown here.

CFIND-*nnn message-text*

nnn is the message number. The following return codes are associated with these messages:

00 - is an informative message
04 - is a warning message
08 - is a syntax error. The current input is rejected; additional input will be read.
12 - is a fatal error. The program ends. (The files are closed.)
16 - is a fatal error. The program ends. (No files need to be closed.)
FF - is a physical end-of-file on the SYSIN file.

The Destination code 01 means that the message is sent to the MESSAGES file. The Destination code 02 means that the message is a WTO.

Message Number	Message Text	Destination Code	Return Code
000	**** BEGIN CFIND VALIDATION** WTO issued at beginning of program.	**02**	**00**
001	**** END CFIND VALIDATION** WTO issued at end of program.	**02**	**00**
002	**Line showing current date and time,** issued with both messages 000 and 001.	**01**	**00**
003	***** USER ABEND XXX ************** User abend encountered. See "User Abend Codes" for further explanation.	**01**	**00**
004	**INVALID OPTIONS SPECIFIED** Options for CHAIN= or CALC= are not entered correctly.	**01**	**08**

Message Number	Message Text	Destination Code	Return Code
005	**SPACE INVALID IN PARAMETER LIST** A space was found where it is invalid.	01	08
006	**NUMERIC PARM CONTAINS INVALID CHAR** A numeric parameter, such as BUFNO or LINE, contains a non-numeric character.	01	08
007	**ZERO LENGTH PARAMETER** A keyword was recognized, but nothing following it was valid.	01	08
008	**FIVE DIGITS MAX IN NUMERIC VALUE** Numeric parameters can have only 1-5 digits.	01	08
009	**HEX PARM CONTAINS INVALID CHAR** Hexadecimal parameters can have only 1-8 digits.	01	08
010	**UNEXPECTED END OF PARM LIST** Either the EXEC statement or a SYSIN record contains input but has no recognizable keywords.	01	12
011	**INVALID MODE SPECIFICATION** The mode keyword is invalid.	01	08
012	**INVALID BUFNO** BUFNO must be a 1 to 5 digit number ranging from 1 to 99.	01	08
014	**EXEC CARD PARAMETER LIST FOLLOWS:** Parameters specified in EXEC statement follow this message.	01	00
015	**EIGHT DIGITS MAX IN HEX VALUE** Hexadecimal parameters must be 1 to 8 digits.	01	08
016	**OPEN FAILED ON SYSIN** The SYSIN DD statement is missing.	01	16
017	**UNEXPECTED EOF ON SYSIN** First attempt to read a SYSIN record caused an end-of-file condition. This occurs at a terminal when a carriage return is pressed with nothing entered.	01	255
018	**EOF ON SYSIN** SYSIN is at the end-of-file.	01	255
019	**CARD READ FROM SYSIN FOLLOWS:**	01	00
020	**NO PARMS INPUT. . .DEFAULTS TAKEN** Neither SYSIN nor EXEC parameters were specified.	01	00
021	**HT EMPTY. . .PAGE 0 IS ALL ZEROS** First page of the CFIND table is all zeros, which indicates no data in database.	02	00

Message Number	Message Text	Destination Code	Return Code
022	**THERE ARE NO CALC RECORDS** MODE=CALC was specified, but the ID table indicates no CALC records exist.	02	08
023	**ID TABLE EMPTY. . .APPEARS FORMATTED** First page of File 1 (ID table) is all zeros. Database has probably been released.	02	12
024	**END USED SPACE CHECK** End of the used space check.	02	00
025	**END RE-USABLE SPACE CHECK** End of the re-usable space check.	02	00
026	**END OF HT CHAIN CHECK** MODE=CHAIN processing is complete.	02	00
027	****** ERROR **** CFIND ENDING** Fatal user abend has been issued. Message 003 indicates the specific error.	02	00
028	**FOLLOWING FILE HAS INVALID BLKSIZE:** The DDname of a file with an incorrect block size follows this message.	01	12
029	*** PROCESSING MODE IS PRINT** Beginning of PRINT mode processing.	02	00
030	*** PROCESSING MODE IS CALC** Beginning of CALC mode processing.	02	00
031	*** PROCESSING MODE IS CHAIN** Beginning of CHAIN mode processing.	02	00
032	**BEGIN USED SPACE CHECK** Beginning of the used space check.	02	00
033	**BEGIN REUSABLE SPACE CHECK**	02	00
034	**SYSPRINT FAILED TO OPEN** SYSPRINT DD statement is missing.	01	08
035	**PATCH REQUIRES ENTRY SPECIFICATION** MODE=PATCH was specified, but ENTRY was missing.	01	08
036	**NO SYSIN INPUT ENCOUNTERED** The SYSIN file was not specified; informative message only.	01	255
037	**ENTRY PAST LAST USED ENTRY** The record specified in the ENTRY parameter is past the last used entry; warning.	01	255

Message Number	Message Text	Destination Code	Return Code
038	**ENTRY PAST END OF THE FILE** Record specified in ENTRY parameter is past physical end-of-file.	01	04
039	**START PAST LAST USED ENTRY** START specified is past last used entry as shown by ID table.	01	08
040	**STOP PAST LAST USED ENTRY** STOP specified is past last used entry as shown in the ID table.	01	04
041	**START HIGHER THAN STOP. . .WILL WRAP** If the START parameter is greater than the STOP parameter, PRINT mode starts with the START value, runs through the last used entry, and then continues from the beginning of the file; it stops when the STOP entry is encountered.	01	04
042	**START OR STOP PAST END OF FILE** Either START or STOP is past the last record in the file; the last record just before physical end-of-file is substituted for either.	01	04
043	**DISPLAY (DEBUG TRACE) DD MISSING** The trace option was specified, but no DISPLAY DD statement was available.	02	04
044	**BEGIN CFIND VALIDATION - (VER 3)** This message is sent to the MESSAGES file when the job starts execution.	01	00
045	**END CFIND VALIDATION - (VER 3)** This message is sent to the MESSAGES file at the end of job execution.	01	00
046	**FIELD TO BE PATCHED IS MISSING** A MODE parameter was specified, but the field parameter was invalid or missing.	01	08
047	**LATEST TWO ENTRIES FOLLOW:** When a user abend occurs, this message precedes the display of the two most current entries: E = entry #, D = down pointer, R = right pointer, U = up pointer, L = level of record, C = internal component number, R = data record pointer.	01	08
048	**ENTRY BEFORE THE PATCH:** When patching an entry, this message precedes the display of the entry as it exists before the patch. The entry is shown on the next line in the same format as Message 047.	01	00

Message Number	Message Text	Destination Code	Return Code
049	**OPEN FAILED ON MESSAGES** The MESSAGES DD statement is missing.	02	00
050	**INVALID VALUE IN LINE PARM** LINE parameter must be 80, 120, or 132.	01	08
051	*** PROCESSING MODE IS PATCH** Beginning of PATCH mode processing.	02	00
052	***** ERROR MAXIMUM REACHED** CHAIN mode re-starts after each user abend until reaching the error maximum. Then, this message is issued and the program terminates with User Abend 49.	01	00
053	**** NO HT ERRORS DETECTED** Issued at the end of chain-checking if no errors were detected.	01	00
054	**BEGIN CHAIN LEFT CHECK** Beginning of left chain-check.	02	00
055	**INVALID DATA RECORD POINTER** Data record pointer cannot be higher than 3FFFFFFF.	01	08
056	**THIS ENTRY IS FLAGGED AS REMOVED** Warning message issued if displayed record is flagged as removed.	01	04
057	**INVALID COMPONENT NUMBER** Internal component number must be greater than or equal to 0 but less than maximum component number as established by the NEW DATA BASE command.	01	04
058	**ENTRY AFTER THE PATCH:** Issued during PATCH mode processing before displaying the entry as it looks after patching.	01	08
059	**MODE SPECIFICATION MISSING** The field to be patched was specified, but MODE=PATCH was not.	01	08
060	**. .READY FOR INPUT. . .** This is an interactive prompt, issued just before a record is read from the SYSIN file.	01	00
061	**NO RECORDS SELECTED** This message is issued at the end of SCAN mode processing to indicate that no records were selected or written to the SYSPRINT file.	01	00
062	**INVALID SCAN FIELD** This message is issued for SCAN mode if the field parameter is not valid.	01	08

Appendix A: Supplementary PLEX Information

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. Displays A.1 through A.4 show the expansion of the S2KDUM control block for each PLEX processor.

Table A.1 S2KDUM Control Block Format

Offset Hex	Offset Dec	Name	Length	Type or 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 or 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 or file to be created by COPYS or COPYL, set by expanded PLEX code 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 by S2KPLR at START S2K time to decide which environment to run the software under
				0	single-user - SYS2K
				1	Multi-User - MUPLINT
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.

Table A.1 S2KDUM Control Block Format (*continued*)

Offset Hex	Offset Dec	Name	Length	Type or Value	Field Description
20	32	S2KPGM	8	Alphameric	Name of the program or subroutine (left-aligned and blank-filled) that contains 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/I, this name is specified by the $NAME directive.
28	40	S2KVER	4	Alphameric	Release number of SYSTEM 2000 PLEX processor used; 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 that PLEX processor was run (YYDDD, left-aligned and blank-filled); set by PLEX processor
34	52	S2KTIM	8	Alphameric	Time that PLEX processor was run (HHMMSSS, left-aligned and blank-filled); set by PLEX processor
3C	60	S2KVS2	4	Alphameric	Release number of SYSTEM 2000 DBMS supplied at run time 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
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)

Table A.1 S2KDUM Control Block Format (*continued*)

Offset Hex	Offset Dec	Name	Length	Type or Value	Field Description
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 run-time 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)

Display A.1 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 ' 1.0'.
   02 FILLER     PIC X(8) VALUE '00214   '.
   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.
```

Display A.2 S2KDUM Control Block Expansion in PL/I

```
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 (' 1.0'),
      2 S2K5       CHAR(8)       INIT ('00214   '),
      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);
```

Display A.3 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))
```

Display A.4 S2KDUM Control Block Expansion in Assembler

```
Col 1             Col 10     Col 16

S2KDUM            DS            OCL152
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        '
                  DC          C'  1.0'
                  DC          C'00214'
                  DC          C'2137524
                            '
                              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 COMMBLOCK vary with the programming language used.

The S and P in the "Set by" column in Table A.2 indicate whether values for the field are set by SYSTEM 2000 or by the PLEX processor.

Table A.2 COMMBLOCK Control Block Format (COBOL definition is shown in Table A.4)

Offset Hex	Offset Dec	Name	Length	Type or Value	Set by	Field Description
0	0	FCNAM	16	Alphameric	P	Database name, can be modified by PLEX program before open.
10	16	FCACTV	4	Binary	P	Database active
				0		Inactive
				≠0		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 0 by the PLEX processor and modified by S2KPLR.

Table A.3 Variables Described in the COMMBLOCK Section of the *SYSTEM 2000 PLEX Manual, Version 12*

Offset Hex	Offset Dec	Name	Length	Type or Value	Set by	Field Description
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
40	64	FCUSER	4	Binary		Available to the user; initialized by PLEX processor to binary 0.
44	68	FCLAST	4	Binary	S	Last data record indicator
				0		One or more data records remain
				1		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 a LOCATE command

Table A.3 Variables Described in the COMMBLOCK Section of the *SYSTEM 2000 PLEX Manual, Version 12* (*continued*)

Offset Hex	Offset Dec	Name	Length	Type or Value	Set by	Field Description
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 0 filled
70	112	FCENT	4	Alphameric	S	Entry terminator word left-aligned, blank-filled
74	116	FCSTAT	4	Binary	S	Database status at open time
				0		Undamaged
				1		Damaged (updates not allowed)
78	120	FCCPUP	4	Binary	S	Optional variable that indicates the parent of the last data record retrieved.

Table A.4 COBOL COMMBLOCK Control Block Format

Offset Hex	Offset Dec	Name	Length	Type or Value	Set by	Field Description
00	0	CCNAM	16	Alphameric	P	Database name can be modified by PLEX program before open
10	16	CCACTV	4	Binary	P	Database active
				0		Inactive
				≠0		Value of SYSTEM 2000 internal database identification
14	20	CCLNTH	4	Binary		Length in bytes of COMMBLOCK including the prefix initialized by the PLEX processor to include or exclude CPUP as the last field.

Table A.4 COBOL COMMBLOCK Control Block Format (*continued*)

Offset Hex	Offset Dec	Name	Length	Type or Value	Set by	Field Description
18	24	CCFOR	4	Binary		FOR BLOCK chain Pointer to current FOR BLOCK initialized to binary 0 by the PLEX processor and modified by S2KPLR
1C	28	CCSCHM	30	Alphameric	S	Subschema record name The name of the last action subschema record referenced except when specific non-zero return codes occur. May contain the name of the item that caused the command to be rejected
3A	58	CCRTC	4	Binary	S	Return code
3E	62	CCUSER	5	Binary		Available to the user; initialized by PLEX processor to binary 0
43	67	CCLAST	1	Binary	S	Last data record indicator
				0		One or more data records remain
				1		This is the last data record
44	68	CCPASS	4	Alphameric		Password set by PLEX program; must be left-aligned and blank-filled.
48	72	CCNRG	10	Binary	S	Number of data records selected by a GET1 or a LOCATE command.
52	82	CCPOS	10	Binary	S	Relative position of the last data record accessed from a Locate file
5C	92	CCLEV	2	Binary	S	Hierarchical level of the last data record accessed
5E	94	CCTIME	6	Alphameric	S	Time of last update HH:MM:SS
64	100	CCDATE	6	Alphameric	S	Date of last update MM/DD/YY
6A	106	CCCYC	8	Binary	S	Last database cycle number
72	114	CCSEP	1	Alphameric	S	System separator right-aligned, binary 0 filled
73	115	CCENT	4	Alphameric	S	Entry terminator word left-aligned, blank-filled
77	119	CCSTAT	1	Binary	S	Database status at open time
				0		Undamaged
				1		Damaged (updates not allowed)
78	120	CCPUP	4	Binary	S	Optional variable indicating the parent of the last data record retrieved.

Table A.5 SUBSCHEMA Control Block Format

Offset Hex	Offset Dec	Name	Length	Type or 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 that follows the last ECB
24	36	SCBINT	4	Binary	S	Internal number of the schema record represented by this subschema record; initialized to binary 0
28	40	SCBLEV	4	Alphameric	S	Internal level number of schema record initialized to binary 0
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 0
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; 0 if link is not active; set by S2KPLR
35	53	SCBFIL	3	Binary		Reserved for internal use; initialized to binary 0
0	0	ECBNAM	16	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.

Table A.5 SUBSCHEMA Control Block Format (*continued*)

Offset Hex	Offset Dec	Name	Length	Type or Value	Set by	Field Description
24	36	ECBINT	4	Binary	S	Internal SYSTEM 2000 number of the item; initialized to binary 0
28	40	ECBLEV	4	Binary	S	Internal SYSTEM 2000 number of the schema record level; initialized to binary 0
2C	44	ECBNUM	4	Binary	S	Internal SYSTEM 2000 number of the owner schema record; initialized to binary 0
30	48	ECBDBI	4	Binary	S	Internal value of the last database this ECB was used with; initialized to binary 0
34	52	ECBACT	1	Bit		Activity indicator;
				…. 0…		Item identification; initialized by PLEX processor;
				…. …n		Item is or is not to participate; checked only if corresponding SCBACT bit is 0; set by PLEX program n=1 item is to participate n=0 item is not to be used
35	53	ECBTYP	4	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.

Table A.6 Opcodes for PLEX Commands

		OPCODE
PLEX Command	**Decimal**	**Hex**
GET1 *SSR* FIRST	00	00
GET1 *SSR* LAST	01	01
ET1 *SSR* S2KCOUNT	03	03
ET1 *SSR* NEXT	04	04
ET1 *SSR* WHERE *where-clause*	05	05
ET1 *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

Table A.6 Opcodes for PLEX Commands (*continued*)

PLEX Command	OPCODE Decimal	Hex
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
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

*Welcome * Bienvenue * Willkommen * Yohkoso * Bienvenido*

SAS Publishing Is Easy to Reach

Visit our Web page located at www.sas.com/pubs

You will find product and service details, including

- **sample chapters**
- **tables of contents**
- **author biographies**
- **book reviews**

Learn about

- **regional user-group conferences**
- **trade-show sites and dates**
- **authoring opportunities**
- **custom textbooks**

Explore all the services that SAS Publishing has to offer!

Your Listserv Subscription Automatically Brings the News to You

Do you want to be among the first to learn about the latest books and services available from SAS Publishing? Subscribe to our listserv **newdocnews-l** and, once each month, you will automatically receive a description of the newest books and which environments or operating systems and SAS® release(s) each book addresses.

To subscribe,

1. Send an e-mail message to **listserv@vm.sas.com**.
2. Leave the "Subject" line blank.
3. Use the following text for your message:

   subscribe NEWDOCNEWS-L ***your-first-name your-last-name***

   For example: subscribe NEWDOCNEWS-L John Doe

Create Customized Textbooks Quickly, Easily, and Affordably

SelecText® offers instructors at U.S. colleges and universities a way to create custom textbooks for courses that teach students how to use SAS software.

For more information, see our Web page at **www.sas.com/selectext**, or contact our SelecText coordinators by sending e-mail to **selectext@sas.com**.

You're Invited to Publish with SAS Institute's User Publishing Program

If you enjoy writing about SAS software and how to use it, the User Publishing Program at SAS Institute offers a variety of publishing options. We are actively recruiting authors to publish books, articles, and sample code. Do you find the idea of writing a book or an article by yourself a little intimidating? Consider writing with a co-author. Keep in mind that you will receive complete editorial and publishing support, access to our users, technical advice and assistance, and competitive royalties. Please contact us for an author packet. E-mail us at **sasbbu@sas.com** or call 919-531-7447. See the SAS Publishing Web page at **www.sas.com/pubs** for complete information.

See *Observations*®, Our Online Technical Journal

Feature articles from *Observations*®: *The Technical Journal for SAS® Software Users* are now available online at **www.sas.com/obs**. Take a look at what your fellow SAS software users and SAS Institute experts have to tell you. You may decide that you, too, have information to share. If you are interested in writing for *Observations*, send e-mail to **sasbbu@sas.com** or call 919-531-7447.

Book Discount Offered at SAS Public Training Courses!

When you attend one of our SAS Public Training Courses at any of our regional Training Centers in the U.S., you will receive a 15% discount on book orders that you place during the course. Take advantage of this offer at the next course you attend!

SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513-2414
Fax 919-677-4444

E-mail: sasbook@sas.com
Web page: www.sas.com/pubs
To order books, call Fulfillment Services at 800-727-3228*
For other SAS business, call 919-677-8000*

*** Note:** Customers outside the U.S. should contact their local SAS office.