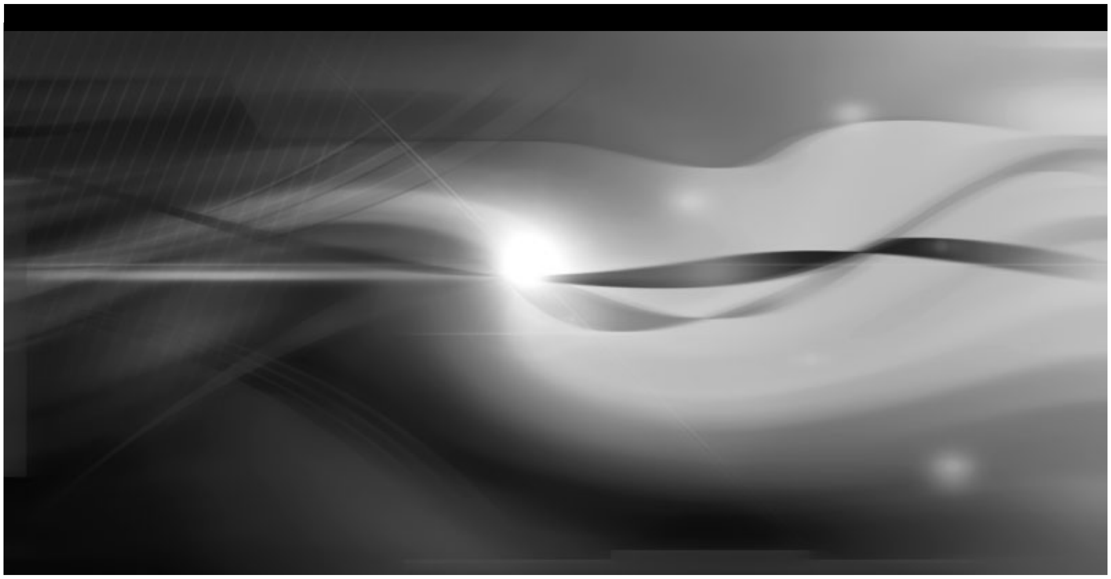
What's New in SYSTEM 2000® V2



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2007. *What's New in SYSTEM 2000® V2*. Cary, NC: SAS Institute Inc.

What's New in SYSTEM 2000® V2

ISBN 978-1-59994-579-8

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

1st printing, September 2007

Table of Contents

What's New in SYSTEM 2000 for Version 2

Get the latest news: To find out what's happening in the Austin SYSTEM 2000 Development and Technical Support group, visit **http://support.sas.com/documentation/onlinedoc/system2000**.

For questions or comments, write us at **s2k@sas.com**.

Overview

Increased File Sizes—Database Files 5 and 6

This change allows up to 2,147,483,639 records in a database and 16 gigabytes of data, or 17,179,854,962 bytes. For more complete information, see "Database File Capacities and CISIZE" in Chapter 3 of the *SYSTEM 2000 Software: Product Support Manual*.

Increased File Sizes—Index Files 2 and 4

Database Files 2 and 4 have increased in capacity from 1,073,741,824 bytes to 4,294,962,719 bytes. For more complete information, see "Database File Capacities and CISIZE" in Chapter 3 of the *SYSTEM 2000 Software: Product Support Manual.*

Keepfile Blocksize Is Always 4088

This enhancement enables KEEP and APPLY processing to work in any session that has a POOL7 buffer size of 4096 or greater. The Keepfile blocksize is always set to 4088 (CISIZE 4096 for the pad).

Pad File Can Be Any Valid CISIZE

Any valid CISIZE is allowed for the pad file. Examples are LRECL 505, CISIZE 512, or LRECL 32761, CISIZE 32768. You can also specify any size for POOL7. If you do not have a pad file, SYSTEM 2000 dynamically allocates a temporary pad. The CISIZE is 22528 if you specify that size for POOL7; otherwise it is 26624.

Multiple Local Holds Buffer and User Exit EXIT07

EXIT07 can be used to specify the size of the Multiple Local Holds buffer. The size can range from 512 to 327,683 bytes, which allows for 65,535 entries (local holds). Each entry is five bytes in length, and there is an eight-byte header for the buffer. The default buffer size is 50,000 bytes. For more information about EXIT07, see Chapter 8, "User Exits," in the *SYSTEM 2000 Software: Product Support Manual.*

Two Web Interfaces to SYSTEM 2000

The CGI Web Interface and the Web Interface to CICS are included in Version 2. These interfaces provide easy access to all your Multi-User databases from a Web browser.

CGI Web Interface
The SYSTEM 2000 CGI Web interface provides direct access to SYSTEM 2000 databases via the Internet. CGI (Common Gateway Interface) is a standard, supported by almost all Web servers, that defines how information is

exchanged between a Web server and an external program (CGI program). CGI programs are stored in the CGI-BIN subdirectory, which is an HFS file under z/OS UNIX. In order to use the CGI Web interface, your site must have an z/OS Web server such as the IBM HTTP server. A CGI-BIN subdirectory and userid and security authorizations must be defined by your z/OS UNIX system services administrator.

Web Interface to CICS

The SYSTEM 2000 Web interface to CICS also provides direct access to SYSTEM 2000 databases via the Internet. The Web SCF interface enables you to access SYSTEM 2000 databases by using product-supplied HTML interfaces. All functionality provided in the 3270 interface, with the exception of the Command Editor, is provided in the Web interface. There is no need for paging and PF keys, because all output is sent to the browser, eliminating constant interaction with the mainframe to view the next or the previous page.

With CICS TS 1.3 and above, IBM supplies a built-in Web server that enables client access from the Web to the mainframe. Rather than entering CICS directly from TCP/IP, clients have an option to enter through a Web server such as the WebSphere Application Server on z/OS. Clients on the Internet rather than an intranet might want this option for additional security. Web support includes not only database access, but operator communication and user status display.

Administrator's Guides for Web Interfaces

Administrator's Guides that include installation instructions are delivered with your Version 2 documentation. Both documents are also available in PDF format on the SYSTEM 2000 Web page (http://support.sas.com).

Messages and Codes

Messages and SYSTEM 2000 Error Codes that are new in Version 2, along with seven messages that were omitted in the Version 1 *SYSTEM 2000 Software: Messages and Codes* manual, are documented in the "Messages and Codes" section of this document. See Chapter 1.

XML Support

XML document generation using SYSTEM 2000 database data is now available through the Self-Contained Facility (SCF) PRINT command. PRINT generates structured output for a group of components, a record, or a group of records. The output consists of the component number or name, a system separator, and the component's value.

Because the output from a PRINT command goes to the Report File, you can assign the Report File to a sequential data set and generate your XML document there. This sequential file in turn can be sent to a PC (or another platform) using an FTP command. See Chapter 2.

User Exits and Database Triggers

EXIT45, which allows you to control database access by checking RACF permissions, has been upgraded and has new action codes 16 and 20. It can be used stand-alone or with EXIT48. EXIT48 enables you to monitor and assign database passwords.

EXIT46 and EXIT47 are used to implement database triggers. New CONTROL module commands are available to specify and control triggers. See Chapter 3.

SQL Data Retrieval in SYSTEM 2000

With Version 2, SQL commands and functionality are available in SYSTEM 2000. New syntax that uses SELECT, GROUP BY, HAVING, and ORDER BY commands offers an even wider variety of retrieval capabilities in the Self-Contained Facility (SCF). Use of the new commands is documented in Chapter 4, "SQL Data Retrieval—Command and Reference."

SCF 64-Bit Reload and High-Speed Index Processing

Using above-the-bar storage, a high-performance SCF RELOAD command lets you reload a database, with keys, in a third of the time required for a PLEX load without keys. Coupled with high-speed index processing that uses the system sort, the new SCF reload makes any load process much faster and more efficient. See Chapter 5.

Database Files 5 and 6 in 64-bit Storage

SYSTEM 2000 Version 2 allows database Files 5 and 6 to be processed above the bar, which provides a tremendous improvement in performance.

New Installation Exec S2KIVR

An interactive application called S2KIVR, written in the REXX language, is available to generate all batch JCL and CLISTs. An overview is included here, and complete information is in the *SYSTEM 2000 V2 Basic, Multi-User, Quex, and Interface to CICS: Installation Guide*.

Two REXX execs, S2KIVR and S2KIVRMN, are copied into a CLIST or REXX library that is accessible to the person generating the SYSTEM 2000 JCL members. The application keylist member S2KKEYS is copied into an ISPF table library. Then, after one modification to the S2KIVR exec, the application is ready for execution.

From an ISPF command line, you enter TSO S2KIVR and press the ENTER key. The first time you run the exec, you must supply the source library name. Then the main S2KIVR panel is displayed. This is where you fill in all the fields based on the values provided on the worksheet in the installation guide.

The application contains a comprehensive Help system. You can press the PF1 key at any time to display a Help panel for the application or any of the data entry fields.

Updated Cross-Memory Services (XMS) Communications

Communication between Multi-User and your user address space is facilitated via IBM Cross Memory Services (XMS). The SYSTEM 2000 XMS routine S2KPC has been updated in Version 2 to take advantage of new and improved cross-memory services. Installation remains the same as in previous releases. See Chapter 2 in the *SYSTEM 2000 Software: Product Support Manual* for additional information.

Chapter 1: Messages and Codes

QUEST Processor Messages

`-352- DATABASE ITEMS IN SELECT CLAUSE MUST ALSO BE IN GROUP BY CLAUSE -`
is a *nonfatal, non-destructive* error. All items in the SELECT clause must be single-valued per group, and must identify either a grouping item or a function. Grouping items do not have to be in the SELECT clause, but SELECT items must be in the GROUP BY clause. Functions are excluded from this requirement. This requirement ensures that all SELECT items have only one value per group.

`-366- TITLE AND AS SPECIFICATIONS MUTUALLY EXCLUSIVE -`
is a *nonfatal, nondestructive* error involving the use of TITLE and AS specifications in the same command. If you define any column specifications with the TITLE syntax, you cannot use AS syntax to define the same or any other columns within that command.

`-369- COUNT(*) SPECIFIED MORE THAN ONCE -`
is a *nonfatal, non-destructive* error. COUNT(*), which is a record count for each group, was specified more than once in the HAVING clause.

`-376- FUNCTION ORDINAL NUMBER IN HAVING CLAUSE IS NOT IN SELECT CLAUSE -`
is a *nonfatal, non-destructive* error. The HAVING clause refers to the relative position of a function in the SELECT clause that does not exist.

`-377- LIMIT SPECIFICATIONS IGNORED FOR GROUP BY -`
is a *warning* message. TRUNCATE does not logically fit with GROUP processing. Therefore, if the command is to be processed (for example, not prematurely terminated due to LIMIT), messages -343- and -342- are replaced with -377-, and LIMIT is ignored for subsequent processing of the command.

`-386- THE BY CLAUSE AND GROUP-BY ARE MUTUALLY EXCLUSIVE -`
is a *nonfatal, nondestructive* error. In a LIST or SELECT command, you cannot use both a standard SCF BY clause and an SQL ORDER BY clause.

`-389- INVALID COLUMN SPECIFICATION IN SQL ORDER BY CLAUSE -`
is a *nonfatal, non-destructive* error. An SQL ORDER BY clause specified a column reference, either number or name, that did not identify a valid column from the retrieval clause. If a number was specified, it was larger than the number of columns defined. If a name was specified, it was not associated with a defined column.

CONTROL Processor Messages

`-525- SPECIFIED COMPONENT MUST BE AN ITEM -`
is a *batch fatal, nondestructive* error. The item specified in a CREATE TRIGGER, REMOVE TRIGGER, or MODIFY TRIGGER command is either a record item or a string/function component and must be an item to be valid.

-526- SPECIFIED COMPONENT IS ALREADY A TRIGGER -
is a *batch fatal, nondestructive* error. The item component specified in a CREATE TRIGGER command is already defined as a trigger.

-527- SPECIFIED COMPONENT IS NOT A TRIGGER -
is a *batch fatal, nondestructive* error. The item specified in a MODIFY TRIGGER or REMOVE TRIGGER command is not currently defined as a trigger.

-532- RELCOMP IS NOT IN TRIGGER RECORD OR PARENT -
is a *batch fatal, nondestructive* error. The item specified by a RELCOMP keyword parameter is not in the same record as the trigger item or is not in a parent record of the record that contains the trigger item.

-579- KEEPFILE BLKSIZE NOT 4088, OR POOL7 BLKSIZE IS LESS THAN 4088 -
is a *batch fatal, nondestructive* error. The Keepfile block size is set by SYSTEM 2000 to 4088 when the first KEEP command is invoked after a SAVE or RESTORE. You might have the wrong file assigned to KEEPFILE, or you might have allocated the pad file with a CISIZE less than 4096.   A SAVE of the database resets the Keepfile length to zero; the next KEEP that follows the SAVE command sets the block size to 4088.

-585- XXXXXXXX FILE ALLOCATION FAILED WITH ERROR – *error code* – *info code*
is a *batch fatal error.* SYSTEM 2000 received a non-zero return code from a Dynamic Allocation function. *xxxxxxxx* indicates either SORTIN or SORTOUT. This error is nondestructive for SCF LOAD, CREATE INDEX, RELOAD (non 64-bit), and PLEX LOAD. This error is destructive for 64-bit RELOAD. A SYSTEM 2000 Error Code 51 is issued with this message. Check the Job Log for any messages relating to the problem. The *error code* and *info code* fields are from the DYNALLOC control block (S99ERROR and S99INFO). These codes are explained in the section “Interpreting DYNALLOC Return Codes” of the *z/OS MVS Authorized Assembler Services Guide.*

-589- SYSTEM SORT OF INDEX VALUES FAILED -
is a *batch fatal, nondestructive* error. The call to the system sort routine return a non-zero return code. The cause of this error can be found on the job log (if SYSOUT is not defined in the JCL) or in the sort output on the SYSOUT data set.

-607- INDEX VALUE LONGER THAN SSMXV VALUE - VALUE WAS -
is a *batch fatal, nondestructive* error. This error occurs when you specify a value in the SSMXV parameter that is shorter than the longest key value in the database.

SYSTEM 2000 Interface Messages

-712- GETMAIN FAILED FOR 00000 BITES -
is a *fatal, nondestructive* error. A request for additional storage failed. Increase region size. If the request was for storage above the bar, try a region size of 0 or consult your systems programmer about usage of above-the-bar storage. A GETMAIN failed for the number of bytes indicated. This message is written to the job log. SCF message -712- is written to S2KMSG. EXIT is automatically invoked for the user.   If the request for storage was above the bar, IARV64 replaces the word GETMAIN.

-713- UNABLE TO PLACE FILE *DDname* ABOVE THE BAR –

Subtask CPYSTSK abended while attempting to copy file *DDname* above the bar.

-730- *DDname* PAGE LIMIT REACHED -

is a *batch fatal, destructive* error. An attempt to format a secondary extent is disallowed because the file has reached maximum size. The limits are as follows:

File 1 - 10,000 items
File 2 - Maximum pointer is FFFFEE1F.
MAX POINTER / (CISIZE-20) -2 = maximum control intervals (pages).
Maximum control intervals for CISIZE 4096 is 1,053,717.
File 3 - Maximum pointer is 0FFFFBB7.
MAX POINTER / (CISIZE-20) -1 = maximum control intervals (pages).
Maximum control intervals for CISIZE 4096 is 65,856.
File 4 - Same as for File 2.
File 5 - Maximum pointer is 7FFFFFF7.
(MAX POINTER / entries per page) -1 = maximum control intervals (pages).
Maximum control intervals for CISIZE 4096 is 9,502,138.

File 6 - Maximum pointer is FFFFF21A.
(MAX POINTER / words per page) -4 = maximum control intervals (pages).
Maximum control intervals for CISIZE 4096 is 4,214,876.

System-Wide and Multi-User Messages

-814- VERSION *xx.x* DATABASE IS NOT COMPATIBLE WITH VERSION *xx.x* -

is a *batch fatal, nondestructive* error. You have specified a database that requires conversion before it can be attached and processed by the current version of software that you are running. See the *SYSTEM 2000 Software: Product Support Manual* for information about conversion processes.

-823- REQ PAGE *nnnnnnnn* BEYOND RANGE OF DDN *xxxxxxxx*. MAX RANGE *nnnnnnnn*.

is a *batch fatal, nondestructive* error. This message precedes SYSER 812. *nnnnnnnn* is a hexadecimal number. REQ PAGE is the page number to be retrieved. MAX RANGE is used pages + count of assigned buffers + 4. The value should be slightly higher than that shown with PRINT SIZE.

SYSTEM 2000 Error Codes

3 A GETMAIN for the multiple local holds buffer failed. This error is preceded by WTO 1220, which identifies the database for which the buffer was requested and the number of bytes requested. The default for number of bytes is 50,000. EXIT7 (S2KEX07) can be used to override the default.

4 The insert of a new record caused the size of either File 5 or File 6 to exceed the maximum capacity. This error is preceded by WTO 0220 or WTO 0221, identifying the file in error.

51 Dynamic allocation failed for the data set named in the -585- message that accompanied this error.

52 The system sort routine indicated a failure.

53 A key value was encountered that was longer than the value specified in the SSMXV parameter.

The following SYSTEM 2000 error codes were in the R12.1 *SYSTEM 2000 Software: Messages and Codes* manual but mistakenly removed from the Version 1.0 *SYSTEM 2000 Software: Messages and Codes* manual:

401 SYSTEM 2000 tried to apply a Keepfile, but the file did not contain recognizable Keepfile records.

402 SYSTEM 2000 found a bad entry trailer while applying a Keepfile. (Probably the Space Flags word was just read.)

403 Instead of reading the Space Flags word, SYSTEM 2000 found an $EOF while applying a Keepfile. The Keepfile was bad. Try to do an APPLY through a good cycle number.

404 SYSTEM 2000 found an error in the length of the Keepfile while processing a KEEP command. The database is damaged.

405-444 Reserved.

445-483 REPORT processor errors occurred. See “REPORT Processor Messages” in the *SYSTEM 2000 Software: Messages and Codes* manual on page 24 for an explanation of these errors. Contact SYSTEM 2000 Technical Support.

484 The REPORT processor could not handle the requested DECLARE nesting. The report needs to be simplified.

WTO Message Descriptions

S2K0220/*sid*- FILE 6 SIZE EXCEEDED -

Insert of a new record caused the size of File 6 to exceed the maximum capacity. This message precedes SYSTEM 2000 Error Code 4.

S2K0221/*sid*- FILE 5 SIZE EXCEEDED -

Insert of a new record caused the size of File 5 to exceed the maximum capacity. This message precedes SYSTEM 2000 Error Code 4.

S2K0237/*sid*- nnnn PAGES TRUNCATED FOR <*DDname*>. SPACE IS ALLOCATED BUT IS UNUSABLE -

is an *informative* message. Space formatted for the indicated file is more than can be used by SYSTEM 2000. The space is allocated to the file, but SYSTEM 2000 will not attempt to use the excess space. See SCF message -730- for space limits.

S2K0500/*sid*- YOUR SERVICE AGREEMENT FOR THIS PRODUCT HAS EXPIRED. PLEASE CONTACT YOUR DBA -

is a *warning* message. The current release of SYSTEM 2000 expires on the date specified in the licensing agreement. After that date, SYSTEM 2000 does not accept input.

S2K0300/*sid*- GENERATING XML NAME REPLACEMENT TABLE

This message is displayed during system initialization when an XREPLACE DD card is found in the JCL.

S2K0301/*sid*- INVALID DBN= IN XREPLACE

This message is displayed by the replacement name table build routine when the first non-comment card is read from the XREPLACE file and it does not contain DBN= . It is recommended that you run all name replacement files through the stand-alone utility XMLCHECK. This utility identifies all errors on the file and gives you a detailed report.

S2K0302/*sid*- ERROR IN LINE# *nnnn* FOR DATABASE *xxxxxxxxxxxxxxxx*
This message is displayed during name replacement table build when a name replacement line is read from XREPLACE and there is no comma between the old name and the new name. It is recommended that you run all name replacement files through the stand-alone utility XMLCHECK. This utility identifies all errors on the file and give you a detailed report.

S2K0303/*sid*- ERROR ENCOUNTERED ACCESSING XREPLACE FILE
This message is displayed when an I/O error is encountered on the XREPLACE file.

S2K0304/*sid*- GETMAIN FAILED FOR INTERNAL XML NAME TABLE
This message is displayed when the attempt to get storage to build the name replacement table fails.

S2K0305/*sid*- THE XREPLACE FILE HAS NO RECORDS
This message is displayed when there are no records on the XREPLACE file.

S2K0712/*sid*- GETMAIN FAILED FOR 00000 BYTES -
is a *fatal, nondestructive* error. A request for additional storage failed. Increase region size. If the request was for storage above the bar, try a region size of 0 or consult your systems programmer about usage of above-the-bar storage. A GETMAIN failed for the number of bytes indicated. This message is written to the job log. SCF message -712- is written to S2KMSG. EXIT is automatically invoked for the user. If the request for storage was above the bar, IARV64 replaces the word GETMAIN.

S2K0713/*sid*- UNABLE TO PLACE FILE *DDname* ABOVE THE BAR –
Subtask CPYSTSK abended while attempting to copy file *DDname* above the bar.

S2K1220/*sid*- GETMAIN FAILED FOR *n* BYTES FOR MLH BUFFER FOR DATABASE *database* -
is an *informative* message. A GETMAIN failed for the Multiple Local Holds buffer. This WTO is followed by SYSTEM 2000 Error Code 3.

S2K1427/*sid*- PLEASE RE-ENTER -
This WTO is not used in Version 2. In earlier releases it indicated that a cancel request for a TP job not in a JOBQ could not be processed because the first JOBQ slot was in use. In Version 2, the TP table is marked for cancel and the WTO is not needed.

Changes to Operator Commands

DBNS

Two columns were added for each database row. The 64-bit column identifies the amount of 64-bit storage that is being used by this database. The 31-bit column identifies the number of pages that have been added to Files 5 and 6 of the database since the original 64-bit space request.

```
S2K1902/01- MUSTATS~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~---
S2K1902/01- MUSTATS DATABASE NAME       SIZE    64-BIT  31-BIT           -
S2K1902/01- MUSTATS LIBRARY               -S        9M      10           -
```

DBN=

Two additional lines of output were added. The 64-bit line identifies the amount of 64-bit storage that is in use by this database. The 31-bit line identifies the number of pages that have been added to Files 5 and 6 of the database since the original 64-bit space request.

```
S2K1903/01- MUSTATS THERE ARE      9 MEGABYTES 64-BIT STORAGE IN USE   -
S2K1903/01- MUSTATS THERE ARE      0 ADDITIONAL 31-BIT PAGES IN USE    -
```

Display Status of Open Databases

Two additional columns of information were added. The 64-bit column identifies the amount of 64-bit storage that is in use by this database. The 31-bit column identifies the number of pages that have been added to Files 5 and 6 of the database since the original 64-bit space request.

```
S2K1429/01-       DBN        PLX SCF S   JOB    64-BIT  31-BIT -
S2K1431/01- LIBRARY          000 000 -              9M      10 -
```

Chapter 2: Generating XML Output with the SCF PRINT Command

The SCF PRINT command enables you to generate an XML document. Two format options, XMLON and XMLOFF, cause the output normally generated by a PRINT command to be reformatted into a valid XML document. These format options are valid only with the PRINT command.

Like the other format options, these options are in effect until you turn them off or exit an SCF session.

The generated XML document has XML element tags made up of either the component number (C101) or the component name (POSITION_TITLE). You control the tags by using the existing NAME format option. The default is component number.

A default XML version specification is supplied, as follows:

```
<?xml version=’1.0’ standalone=’yes’?>
```

This specification means that SYSTEM 2000 conforms to the 1.0 version of XML and that everything needed for this document is contained in the document.

Another part of any valid XML document is the root element, which looks like this:

```
<S2K_xxxxxxxxxxxxxxxx>
```

where *xxxxxxxxxxxxxxxx* is the current database name.

XML tag names do not allow the following characters to be imbedded in the tag name:

blank	( )
ampersand	(&)
less-than	(<)
greater-than	(>)
apostrophe	(‘)
quotation mark	(“)

If you are using the NAME format option and a component name contains one of the preceeding characters, that character is replaced with an underscore character. For example, the EMPLOYEE NUMBER component name would show up as the XML tag <EMPLOYEE_NUMBER>.

To generate an XML document that contains all of the data values in your database, you issue the following command:

```
PRINT /XMLON/ C0:
```

To limit the output of your XML document to the components in the C100 record, you issue the following command:

```
PRINT /XMLON/ C100:
```

To limit your XML document to C100 records but only those C100 records where the C101 component is equal to the value “ABCDEF,” you issue the following command:

```
PRINT /XMLON/ C100 WH C101 EQ ABCDEF:
```

Examples

Output from the PRINT command goes to the Report File. The following examples were generated from the standard EMPLOYEE database:

Example 1:

```
PRINT /XMLON/ C132 WH C1 EQ 1043:

<?xml version=’1.0’ standalone=’yes’?>
<S2K_EMPLOYEE>
       <C130>
       <C132>HUSBAND EMPLOYED AS INSTRUCTOR</C132>
       <C132>IN THE MARKETING DEPARTMENT </C132>
       <C132>NAME: GEORGE J. GIBSON</C132>
       <C132> TEST TEXT  </C132>
</C130>
</S2K_EMPLOYEE>
```

Example 2:

```
PRINT /NAME,XMLON,REPEATSUPPRESS/ C1,C201 WH C1 EQ 1043:

<?xml version=’1.0’ standalone=’yes’?>
<S2K_EMPLOYEE>
<ENTRY>
<EMPLOYEE_NUMBER>1043</EMPLOYEE_NUMBER>
       <SKILL_TYPE>GRAPHICS</SKILL_TYPE>
       <SKILL_TYPE>CARTOON ART </SKILL_TYPE>
</ENTRY>
</S2K_EMPLOYEE>
```

Element Name Replacement and XML Overrides

In cases where the database component names are not sufficient or not recognizable enough, you can specify a name replacement table in your SYSTEM 2000 JCL using the XREPLACE ddname. The names specified in this table are replaced as the XML element tags are being built. The name replacement table must be a sequential file or a member of a PDS that is defined with a logical record length of 80. A normal entry in the table looks like the following:

```
DBN=EMPLOYEE
EMPLOYEE NUMBER,EMPNO
SOCIAL_SECURITY_NUMBER,SSN
BIRTHDAY,DATE OF BIRTH
C1,EMPNO
C7,SSN
C6,DATE OF BIRTH
```

You must specify the database name to which the replacements pertain. Each *NAME,REPLACEMENT* pair must be on a separate record. A *NAME,REPLACEMENT* pair cannot occupy more than one record (80 bytes) on the file. For the *NAME* part of the pair, you can specify either the component number (C1), the component name (EMPLOYEE

NUMBER), or the XML tag name that would have been built (EMPLOYEE_NUMBER). Do not put spacing after the *NAME* or before the *REPLACEMENT* or these will be included in trying to match the names in between.
You can validate a name replacement table before putting it into production by running the XMLCHECK utility against it. This utility will identify any error conditions in the table. Sample JCL for running this utility is as follows:

```
//XMLCHECK JOB
//STEPLIB DD DSN=S2K.V2.LOAD,DISP=SHR
//XREPLACE DD DSN=MYDSN.REPLACE,DISP=SHR
//PRINT DD SYSOUT=*
```

You can also override the default XML version specification by specifying one prior to any DBN= records. You can add comments to the name replacement table that will be placed in all generated XML documents and you can add XML directives for specifying style sheets. You can also override the default root element label in the name replacement table. SYSTEM 2000 automatically saves your specified root element name and generates a closing tag for it at the end of the XML document. The previous replacement table with all of the optional specifications would look like the following:

```
/* COMMENTS DELIMITED LIKE THESE WILL NOT BE MOVED TO   */
/* GENERATED XML DOCUMENTS. THESE COMMENTS ARE          */
/* HERE SIMPLY TO EXPLAIN THE SOURCE OF THIS TABLE.     */

<!-- THIS COMMENT WILL BE PLACED INTO EACH GENERATED
       XML DOCUMENT -->

/* THE FOLLOWING DIRECTIVE WILL BE ADDED TO EACH XML    */
/* DOCUMENT GENERATED.  IT WILL REPLACE THE DEFAULT.     */

<?xml version=’1.1’?>

/* THE FOLLOWING DIRECTIVE WILL SPECIFY A STYLE SHEET   */
/* THAT WILL BE APPLIED TO THE XML DOCUMENT WHEN IT IS  */
/* OPENED.  IT WILL BE ADDED TO EACH GENERATED XML      */
/* DOCUMENT.                                            */

<?xml-stylesheet type=”text/xsl” href=”emp2salh.xsl”?>

/* THE FOLLOWING ROOT ELEMENT TAG WILL OVERRIDE THE     */
/* DEFAULT ROOT ELEMENT TAG IN GENERATED DOCUMENTS.      */

<emproot type=”s2k” date=”02/05/2004”>

DBN=EMPLOYEE
EMPLOYEE NUMBER,EMPNO
SOCIAL_SECURITY_NUMBER,SSN
BIRTHDAY,DATE OF BIRTH
C1,EMPNO
C7,SSN
C6,DATE OF BIRTH
```

Further explanation of XML document generation is available on the SYSTEM 2000 external Web site, including more samples of XML documents and XML style sheets.

Chapter 3: User Exits 45, 46, 47, and 48

EXIT45

EXIT45 has two additional action codes:

16 (update) Master password updates are allowed, but master password authorities are disabled. Do not save TEMPSTR2 in the URA.

20 (update) Master password updates are allowed, but master password authorities are disabled. Save TEMPSTR2 in the URA.

EXIT46

EXIT46 occurs each time the value for a trigger item is modified, inserted, or deleted.

When Coordinated Recovery is enabled for this database, a 32K buffer is provided to store information about each trigger occurrence until all updates are committed. Once committed, EXIT47 occurs and is passed this 32K buffer. If the updates are rolled back, then the 32K buffer is deleted.

Without Coordinated Recovery, EXIT46 must process the trigger information completely, because EXIT47 will not occur.

CONTROL module commands are available to control trigger items. See the following for the new command syntax. Here is an example:

```
USER,DEMO:
DBN IS EMPLOYEE:
CONTROL:
CREATE TRIGGER C3:
CREATE TRIGGER C3 RELCOMP=C1:
REMOVE TRIGGER C3:
LIST TRIGGERS:
ACCESS:
...
```

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information (Multi-User only)
ACOMBLK	COMMBLOCK (PLEX only)
ASCHEMA	subschema record (PLEX only)
AS2KDUM	S2KDUM (PLEX only)
AERMSGLN	user message length
AERMSGBF	user message buffer
ACOMPTYP	trigger component type

Parameter Label	Field Description
ACOMPLN	trigger component length
ACOMPKY	trigger component key indicator
ACOMPFLT	trigger component decimal specification
ACOMPVAL	trigger component value
ACOMPNO	trigger component number
ACMPLOLD	old trigger value length
ACMPVOLD	old trigger value
ARELCTYP	related component type
ARELCFLT	related component decimal specification
ARELCLN	related component value length
ARELCVAL	related component value
ARELCNO	related component number
ATRIGBUF	trigger buffer (Coordinated Recovery only)
ATRIGBFL	trigger buffer length (Coordinated Recovery only)

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

EXIT47

EXIT47 occurs when updates are committed to database(s) and EXIT46 has processed trigger information and done a GETMAIN for a 32K buffer. Coordinated Recovery must be enabled when EXIT46 occurs. If no trigger buffer is available, EXIT47 is not taken at COMMIT time. EXIT47 is not taken if updates are backed out using the ROLLBACK command.

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
ACCTINFO	ACCT information (Multi-User only)
AS2KDUM	S2KDUM (PLEX only)
AERMSGLN	user message length
AERMSGBF	user message buffer
ATRIGBUF	trigger buffer (Coordinated Recovery only)
ATRIGBFL	trigger buffer length (Coordinated Recovery only)

Allowable action codes are: 4, 12, 40, 44

EXIT48

EXIT48 occurs before a database is opened. This exit can alter the password by storing a valid password in APASSWRD.

For PLEX programs with multiple databases, EXIT48 needs to modify the APASSWRD field each time the program switches databases, because the password specified in the COMMBLOCK is not altered.

Parameter Label	Field Description
AUSRCTL	user exit control block
ATMPSTOR	temporary storage
AEXTWT	exit wait routine
APASSWRD	password
ADBNAME	database name
ACCTINFO	ACCT information (Multi-User only)
ACOMBLK	COMMBLOCK (PLEX only)
ASCHEMA	subschema record (PLEX only)
AS2KDUM	S2KDUM (PLEX only)
AERMSGLN	user message length
AERMSGBF	user message buffer
ATMPSTRZ	2nd 400 byte temporary storage

Allowable action codes are: 0, 4, 8, 40

Database Triggers

You can specify a trigger item that will cause user exits to be called anytime the trigger item is updated. The process is driven by EXIT46 and EXIT47. Designated triggers fire when an update command is issued against the trigger item.

The new CREATE, REMOVE, and MODIFY TRIGGER commands are CONTROL processor commands and must be issued under the master password holder with exclusive use of the database.

Command Syntax:

```
CREATE TRIGGER Cnnnn     [RELCOMP=Cnnnn]:
REMOVE TRIGGER Cnnnn:
MODIFY TRIGGER Cnnnn     [RELCOMP=Cnnnn]:
LIST TRIGGERS:
```

The CREATE TRIGGER command is used to identify an item as a trigger. If the specified item is involved in any update process, designated user exits are called.

The REMOVE TRIGGER command can be used to remove a previously defined trigger. The REMOVE TRIGGER command also clears any RELCOMP fields associated with the named trigger item.

The MODIFY TRIGGER command can be used to change the specifications of a previous CREATE TRIGGER command. If no RELCOMP keyword is specified on the MODIFY TRIGGER command, any existing RELCOMP information is cleared for the named trigger item. If a RELCOMP parameter is specified, information is added, or replaced, for an existing trigger item.

You can specify any item in a database to be your trigger item. The item specified cannot be a record component. A trigger fires when any update is done that involves that item.

The RELCOMP= keyword can be used to define a related item whose value is to be supplied at the time the trigger fires. The related item can be either in the same record as the trigger item or in a parent of the record that contains the trigger item. If the related item is not in the same record as the trigger item or is not in a parent record of the trigger item, you receive an error message to that effect and the command fails. The value of this related item is presented to the user exit when the trigger fires.

Considerations:

- In an INSERT, the related item is not valued when the related item is after the trigger item in DESCRIBE order.

- For REMOVE, the related item is not valued when the related item has been removed and precedes the trigger item in DESCRIBE order.

- When the length of trigger item, related item, or old item is 0, then the values for these items do not exist.

Chapter 4: SQL Data Retrieval—Command and Reference

SELECT

The SELECT statement retrieves data. The general format of the SELECT statement is as follows:

```
SELECT [ALL/DISTINCT] SELECT-LIST [AS <column-name>]
GROUP BY expression
HAVING condition
ORDER BY expression
WHERE condition(s)
```

The order of clauses in a SELECT (or LIST) command must be as follows:

```
SELECT ..., GROUP BY ... HAVING ..., ORDER BY ... WHERE ...:
```

The SQL ORDER BY must be the last command before the WHERE clause. The GROUP BY, HAVING, and ORDER BY clauses are optional, but if specified, they must be in the indicated order.

The ALL option retains all rows of the final result table and does not eliminate duplicate results. ALL is the default; it can't be specified in the command.

The DISTINCT option eliminates all but one of each set of duplicate rows of the final result table. Two rows are duplicates of one another if the value of the distinct item in the first row is equal to the corresponding value in the second row.

SELECT-LIST is the items, or columns, of information requested.

AS <column-name> allows for user-specified column names.

AS Clause

The AS syntax specifies an alias column heading name to replace the default output from a LIST or SELECT command. The item alias is specified with the following syntax:

```
item AS [L(width)] column-heading
        [R(width)] column-heading
```

See the rules for width and column-heading under the TITLE description in Chapter 11 of the Version 12, *SYSTEM 2000 Data Management Software: QUEST Language and System-Wide Command* manual.

GROUP BY

The GROUP BY clause enables you to create groups of selected rows and to use group functions to return a single row of summary information. GROUP BY logically rearranges the table represented by qualified records of the WHERE clause into groups, so that within any one group, all rows have the same value for the GROUP BY field.

Without a GROUP BY clause, the application of SQL column functions returns one row. When GROUP BY is used, the function is applied to each group, returning as many rows as there are groups.

You can specify one or more database items in the GROUP BY clause to group the rows. Functions cannot be used in the GROUP BY clause because the result has not yet been derived when the rows are first grouped. Also, because each expression in the SELECT clause must be single-valued per group, each database item (non-function) in the SELECT clause must also be specified in the GROUP BY clause. However, you can specify items in the GROUP BY clause that are not in the SELECT clause.

HAVING

The HAVING clause is a conditional expression used to eliminate groups, just as WHERE is used to eliminate rows. You can use the HAVING clause to specify a search condition for the groups selected based on a GROUP BY clause. The HAVING clause restricts groups to those that satisfy the condition in that clause. This means that your search condition in the HAVING clause must test *properties of each group* rather than *properties of individual rows in the group.*

The HAVING clause refers to an ordinal of a function in the SELECT statement, or to a row count within a group. For example:

```
SELECT MIN(C1), MAX(C1), GROUP BY C1 HAVING (1) GT '10',(2) GT (1), COUNT(*) GT '1'
WH C1 EXISTS:
```

In the HAVING clause, (1) refers to the first function in the SELECT clause. (2) refers to the second SELECT function. COUNT(*) refers to the number of rows in a group. With the exception of COUNT(*), functions specified in the HAVING clause must also be in the SELECT clause. Also, the HAVING clause can refer only to a function or to a literal value (such as 10 in the example).

HAVING conditions are assumed to be connected with the AND operator. If any condition fails, the group is rejected.

Comparisons can be `(function)` to `(function)`, or `(function)` to `'literal'`. Also, COUNT(*) is a row count by group, whose results can be compared to a function or a literal. For example:

```
SELECT MAX(C1), AVG C2, SUM(C3), COUNT C4, MIN C5,
GROUP BY C1
HAVING (5) EQ (1), (3) GT '100.0000', (4) LT '999',COUNT(*) GE '2':

A) MIN C5 EQ MAX(C1)
B) SUM(C3) GT '100.0000'
C) COUNT C4 LT 999
```

`D) Groups with LT 2 records are rejected`

The comparison of HAVING conditions must deal with like values. If (5) is 3 decimal places, then (1) must also be 3 decimal places when the comparison is performed. Also, (3) must be 4 decimal places, or the literal 100.0000 must be changed to be like (3). Decimal digits are automatically added or deleted for comparison purposes only.

The field left of the relational operator determines the type and number of decimal places of the right field. The left field must always be a function. The right field can be a function or a literal. A literal must be numeric or a date. If decimal places are truncated, rounding up occurs based on the left-most truncated digit. When the left-most truncated digit is GE 5, the preceding digit is incremented by 1.

You can specify a system MIN or MAX function to select the minimum or maximum value of a text or character field. You can also refer to that system function in the HAVING clause, but that part of the HAVING statement is ignored. Real, double, and undefined item types in GROUP BY and HAVING statements are processed the same as text and character; you can use those item types in a GROUP BY statement, but if they are referred to in a HAVING statement, that part of the HAVING statement is ignored.

Functions in SQL vs. S2K

There are five aggregate functions in SQL. They are SUM, AVG, MAX, MIN, and COUNT. They are aggregate because they summarize a query's results, rather than listing all of the rows. SYSTEM 2000 has the same functions, plus SIGMA. The difference between functions in SYSTEM 2000 and SQL is the way they can be used. For example, SYSTEM 2000 does not allow a function in the WHERE clause; SQL does.

Ad Hoc Functions

DATE: MAX or MIN—date format. If integer format is included in the computation, the result is elapsed days in integer format. If decimal format is included, the result is three decimal places.

SUM—integer format. There is some variance, depending on whether GROUP BY is used. Without GROUP BY and with only one selected database value, the result is date format; otherwise, the result is integer format. With GROUP BY, the result is always integer format.

AVG—three decimal places.

INTEGER: MIN, MAX, SUM**—**integer format. If decimal format is included in the function, in any order, the result is three decimal places.

Integer + money = money format, two decimal places
Money + integer = decimal format, three decimal places

AVG—three decimal places, unless money is the last field in the computation.

DECIMAL: Three decimal places, even if the database has more or less. Database value 1.0005 is rounded up to 1.001. Zeroes are added if the database value is less than three decimal places.

Decimal + money = money format, two decimal places
Money + decimal = decimal format, three decimal places

The decimal point is not assumed on user input.

MONEY: Two decimal places, even if the database has more.

Integer + money = money format, two decimal places
Money + integer = decimal format, three decimal places

Decimal + money = money format, two decimal places
Money + decimal = decimal format, three decimal places

Money + money = money format, two decimal places

The decimal point is not assumed on user input.

System Functions

DATE: MAX or MIN—date format.

SUM—integer format. There is some variance depending on whether GROUP BY is used. Without GROUP BY and with only one selected database value, the result is date format; otherwise, the result is integer format. With GROUP BY, the result is always integer format.

AVG—three decimal places.

INTEGER: Integer format.

DECIMAL: MAX or MIN—same as database value.

SUM or AVG—three decimal places.

MONEY: MAX or MIN—same as database value.

SUM or AVG—two decimal places.

TEXT or **CHARACTER**: MAX or MIN—same as database value.

ORDER BY

The ORDER BY clause is used to produce a result table in a specific order, such as ascending order of some column(s). When GROUP BY is in effect, ORDER BY applies to the arrangement of output rows. It has no effect on order of processing, as it does when there is no GROUP BY.

The syntax for the SQL ORDER BY clause is as follows:

```
,ORDER BY          ASC           column-number
,ORDERED BY        LOW           column-name
,OB                DESC
                   HIGH
```

The *column-number* represents the position of a retrieval operand in the retrieval clause. The *column-name* is the name assigned to an output column either by a TITLE specification, an AS specification, or by default. The default column name for a database component is the component name from the database. For functions, the default column names are determined by the type of function. Ad hoc functions do not get assigned a column name by default. System functions are, by default, assigned the name *function component-name.* Here are some examples:

Ad Hoc Functions	Default Column Name
`MAX(C1)`	`No default column name`
`(C124-C114)`	`No default column name`

System Functions	Default Column Name
`MAX C1`	`MAX EMPLOYEE NUMBER`
`MIN C2`	`MIN LAST NAME`

If no name is designated for a column that does not have a default name, then that column can only be referenced in an SQL ORDER BY clause using *column-number*.

Ascending (ASC) and descending (DESC) represent the ordering rules for column values. ASC is the default. LOW and HIGH can be substituted for ASC and DESC. The SQL ORDER BY clause must be placed in the command between the GROUP BY clause and the WHERE clause. If there is a HAVING clause associated with a GROUP BY clause, the ORDER BY clause must be placed between the HAVING clause and the WHERE clause. The ORDER BY clause must be preceded by a comma (,). The columns specified in an ORDER BY clause are limited to those defined in the retrieval clause of a LIST / SELECT command (actual columns of output).

LIMIT

The limit command is used to establish a minimum number of records selected by the WHERE processor, and to specify the maximum number of records for which the action part of the command (SELECT, CHANGE, etc.) is to be applied. The minimum specification must always be met, or the command is terminated. The options CANCEL and TRUNCATE identify which action to take if the maximum specification is exceeded.

CANCEL directs that the command be terminated with no further processing.

TRUNCATE directs that all records, in LOC6 order, exceeding the maximum limit be ignored.

The following messages are issued when TRUNCATE or CANCEL is invoked (the number of selected records for CANCEL is 0):

```
 -340-     329 TOTAL SELECTED RECORD(S) -
 -343-     309 RECORD(S) CANCELLED OR TRUNCATED -
 -342-      20 SELECTED RECORD(S) -
```

TRUNCATE does not logically fit with GROUP processing. Therefore, if GROUP BY is in the command and the command is to be processed (e.g., not prematurely terminated due to LIMIT), messages -343- and -342- are replaced with message -377-, and LIMIT is ignored for subsequent processing of the command.

```
-340-     329 TOTAL SELECTED RECORD(S) -
-377- LIMIT SPECIFICATIONS IGNORED FOR GROUP BY -
```

LIMIT applies for the WHERE processor, and messages -340- , -343-, and -342- are issued when the command is prematurely terminated.

LIKE

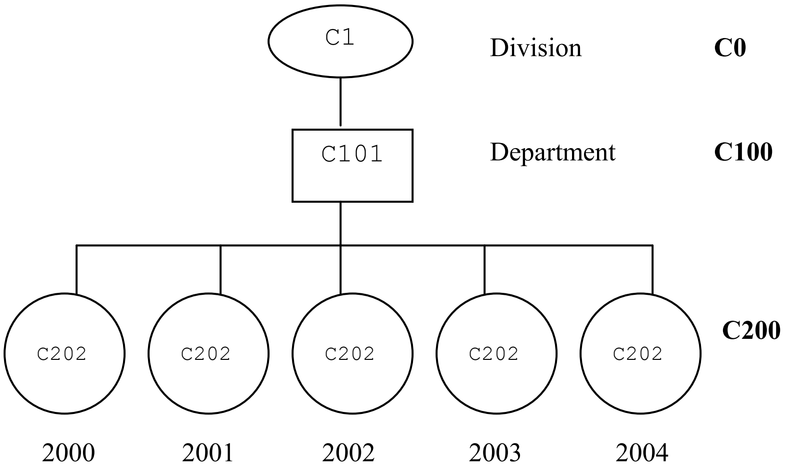
LIKE is a WHERE clause text search specification. It is similar in function to CONTAINS, but has stricter limits. Assume a C101 value of ABC.

`WH C101 LIKE '%C'` 0 to any number of leading characters. Ends with the character C.
`WH C101 LIKE 'A%'` begins with the character A. 0 to any number of trailing characters.
`WH C101 LIKE '%A%'` contains the character A anywhere in the value.
`WH C101 LIKE A` same as '%A%'.

The value must be enclosed in single quotation marks.

SELECT Command Examples

The following examples of the SELECT command use a database that contains sales and expense data for a department store. The database contains data for the Automotive, Electronics, and Housewares divisions. Each division contains several departments. Data is recorded for each department by year. This diagram shows the partial layout of a Division/Department C0 record:



Here is the database definition:

```
1*   DIVISION NAME (CHAR X(20)):
100* DEPARTMENT (RECORD):
101* DEPARTMENT NAME    (CHAR X(20) IN 100):
200* SALES EXPENSE      (RECORD IN 100):
202* REPORTING YEAR     (INTEGER 9999 IN 200):
203* QTR1 EXPENSES      (NON-KEY MONEY NUMBER 9(07).99 IN 200):
204* QTR2 EXPENSES      (NON-KEY MONEY NUMBER 9(07).99 IN 200):
205* QTR3 EXPENSES      (NON-KEY MONEY NUMBER 9(07).99 IN 200):
206* QTR4 EXPENSES      (NON-KEY MONEY NUMBER 9(07).99 IN 200):
207* YTD EXPENSES       (MONEY FUNCTION
                         $ (C203 + C204 + C205 + C206) $ ):
208* PROJECTED EXPENSES (NON-KEY MONEY NUMBER 9(07).99 IN 200):
```

```
209* QTR1 SALES         (NON-KEY MONEY NUMBER 9(07).99 IN 200):
210* QTR2 SALES         (NON-KEY MONEY NUMBER 9(07).99 IN 200):
211* QTR3 SALES         (NON-KEY MONEY NUMBER 9(07).99 IN 200):
212* QTR4 SALES         (NON-KEY MONEY NUMBER 9(07).99 IN 200):
213* YTD SALES          (MONEY FUNCTION
                         $ (C209 + C210 + C211 + C212) $ ):
214* PROJECTED SALES    (NON-KEY MONEY NUMBER 9(07).99 IN 200):
```

There are five years of data for each department.

The SELECT command can use the same options and parameters that the LIST command uses. Most of these examples use the TITLE specification to provide a heading for the output of the SELECT command.
Example 1:

We want to list the first-quarter sales and expenses for all divisions and departments for the year 2002, and we want the output ordered by department within division. The ORDER BY clause in this example is a regular SCF ORDER BY clause and not an SQL ORDER BY clause, because the GROUP BY clause is not used.

```
SELECT /TITLE(79),
D(20)FIRST QUARTER SALES AND EXPENSES FOR 2002,
L(12)DIVISION, L(15)DEPARTMENT, R(15)QTR1+SALES, R(15)QTR1+EXPENSES/
C1, C101, C209, C203,
ORDER BY C1, C101
WH C202 EQ 2002:
```

```
1                   FIRST QUARTER SALES AND EXPENSES FOR 2002
                                   07/20/2004

 * DIVISION       DEPARTMENT                    QTR1              QTR1
                                               SALES          EXPENSES
 ***
 * AUTOMOTIVE     SERVICE                $600,000.00       $520,000.00
 * AUTOMOTIVE     TIRES                   $77,000.00        $50,000.00
 * AUTOMOTIVE     TOOLS                   $33,000.00        $32,000.00
 * ELECTRONICS    CAMERAS                $115,000.00       $111,000.00
 * ELECTRONICS    COMPUTERS              $600,000.00       $620,000.00
 * HOUSEWARES     APPLIANCES             $490,000.00       $470,000.00
 * HOUSEWARES     FURNITURE            $1,250,000.00     $1,150,000.00
```

Example 2:

We want to list the average dollar amount of annual sales and expenses for the Electronics Division. The AVG function is used to get the average of first through fourth quarter sales and expenses and the GROUP BY clause groups the results by department. The ORDER BY (OB) clause rearranges the output in order of year, then department.

```
SELECT /TITLE(79),
D(05)AVERAGE ANNUAL SALES AND EXPENSES FOR THE ELECTRONICS DIVISION,
L(12)DIVISION,L(12)DEPARTMENT,
R(4)YEAR,R(13)AVERAGE+ANNUAL+SALES,R(13)AVERAGE+ANNUAL+EXPENSES/
C1,C101,C202,AVG(*C213*),AVG(*C207*),
GROUP BY C1,C101,C202,
OB 3,2
WH C1 EQ ELECTRONICS:
```

```
1    AVERAGE ANNUAL SALES AND EXPENSES FOR THE ELECTRONICS DIVISION
                               07/20/2004
```

```
 * DIVISION       DEPARTMENT     YEAR         AVERAGE         AVERAGE
                                               ANNUAL          ANNUAL
                                                SALES        EXPENSES
 ***
 * ELECTRONICS    CAMERAS        2000     $487,000.00     $444,000.00
 * ELECTRONICS    COMPUTERS      2000   $2,815,000.00   $2,625,000.00
 * ELECTRONICS    CAMERAS        2001     $600,000.00     $431,000.00
 * ELECTRONICS    COMPUTERS      2001   $3,262,500.00   $2,375,500.00
 * ELECTRONICS    CAMERAS        2002     $663,000.00     $463,000.00
 * ELECTRONICS    COMPUTERS      2002   $2,928,000.00   $2,783,000.00
 * ELECTRONICS    CAMERAS        2003     $747,000.00     $553,000.00
 * ELECTRONICS    COMPUTERS      2003   $3,282,000.00   $2,941,000.00
 * ELECTRONICS    CAMERAS        2004     $533,000.00     $539,000.00
 * ELECTRONICS    COMPUTERS      2004   $2,448,000.00   $2,279,000.00
```

Example 3:

We want to list all departments with annual sales greater than $2,000,000.00. The SELECT statement retrieves data for all departments and divisions, and the HAVING clause eliminates from the output results for any departments with $2,000,000.00 or less in annual sales.

```
SELECT /TITLE(79),
D(05)DIVISIONS AND DEPARTMENTS WITH MORE THAN $2000000 ANNUAL SALES,
R(4)YEAR, L(12)DIVISION, L(12)DEPARTMENT, R(13)ANNUAL+SALES/
C202, C1, C101, SUM(*C207*),
GROUP BY C202, C1, C101
HAVING (1) GT ‘2000000.00’
WH C1 EXISTS:
```

```
1    DIVISIONS AND DEPARTMENTS WITH MORE THAN $2000000 ANNUAL SALES
                              07/20/2004

 * YEAR   DIVISION       DEPARTMENT              ANNUAL
                                                  SALES
 ***
 * 2000   ELECTRONICS    COMPUTERS      $2,625,000.00
 * 2000   HOUSEWARES     FURNITURE      $5,000,000.00
 * 2001   ELECTRONICS    COMPUTERS      $2,375,500.00
 * 2001   HOUSEWARES     FURNITURE      $5,420,000.00
 * 2002   ELECTRONICS    COMPUTERS      $2,783,000.00
 * 2002   HOUSEWARES     APPLIANCES     $2,070,000.00
 * 2002   HOUSEWARES     FURNITURE      $5,350,000.00
 * 2003   AUTOMOTIVE     SERVICE        $2,210,000.00
 * 2003   ELECTRONICS    COMPUTERS      $2,941,000.00
 * 2003   HOUSEWARES     FURNITURE      $8,950,000.00
 * 2004   AUTOMOTIVE     SERVICE        $3,110,000.00
 * 2004   ELECTRONICS    COMPUTERS      $2,279,000.00
 * 2004   HOUSEWARES     APPLIANCES     $2,110,000.00
 * 2004   HOUSEWARES     FURNITURE      $8,460,000.00
```

The following examples use the EMPLOYEE database that is shipped with SYSTEM 2000. You can run these examples against your copy of the database to see how they work.

Example 4:

We want to list the year-to-date salaries of all employees for the year 1991. This example uses the AS specification instead of TITLE. This example limits the query to employees with an employee number of less than 1015 just to limit the amount of output for the example. You can remove that part of the WHERE clause to see the output for all employees.

```
SELECT C2        AS L(15)NAME,
       C101      AS L(33)JOB TITLE,
       SUM(C124) AS R(13)YTD+PAY,
GROUP BY C2,C101
WH C121 SPANS 01/01/1991*12/31/1991 AND C124 EXISTS AND C1 LT 1015:
```

```
* NAME              JOB TITLE                                     YTD
                                                                  PAY
***
* BOWMAN            EXECUTIVE VICE-PRESIDENT               $27,000.00
* BROWN             MANAGER WESTERN REGION MARKETING       $21,750.00
* GARRETT           VICE-PRESIDENT MARKETING               $24,750.00
* HERNANDEZ         MANAGER DATA BASE ADMINISTRATION       $19,500.00
* JONES             SR SYSTEMS ANALYST                     $11,250.00
* KNAPP             VICE-PRESIDENT ADMINISTRATION & F      $17,250.00
                    INANCE
* KNIGHT            SECRETARY                               $6,000.00
* QUINTERO          OPERATIONS SUPERVISOR                  $15,000.00
* SALAZAR           ADMINISTRATIVE ASSISTANT                $7,500.00
* SMITH             SR SYSTEMS PROGRAMMER                  $11,250.00
* VAN HOTTEN        MANAGER PUBLIC RELATIONS & EDUCAT      $15,000.00
                    ION
* WATERHOUSE        PRESIDENT                              $39,800.00
```

To see the effect of the GROUP BY clause, run the SELECT statement again without GROUP BY, like this:

```
SELECT C2        AS L(15)NAME,
       C101      AS L(42)JOB TITLE,
       SUM(C124) AS R(10)YTD+PAY,
WH C121 SPANS 01/01/1991*12/31/1991 AND C124 EXISTS AND C1 LT 1015:
```

Example 5:

In this example, we want to list the year-to-date salaries for each job title, not for each employee. This query uses SUM to add the monthly gross pay for each employee, then groups these sums by job title.

```
SELECT /TITLE(79)
D(10)YTD SALARIES BY JOB TITLE,
L(42) JOB TITLE,R(13)YTD+SALARY/
C101,SUM(C124),
GROUP BY C101
WH C121 SPANS 01/01/1991*12/31/1991 AND C124 EXISTS:
```

```
1         YTD SALARIES BY JOB TITLE
                 07/20/2004

 *  JOB TITLE                                             YTD
                                                       SALARY
 ***
 * ADMINISTRATIVE ASSISTANT                         $7,500.00
 * BOOKKEEPER                                       $6,937.50
 * COMPUTER LIBRARIAN                               $5,812.50
 * COMPUTER OPERATOR                               $13,500.00
 * EXECUTIVE VICE-PRESIDENT                        $27,000.00
 * GENERAL MAINTENANCE                                $934.50
 * INSTRUCTOR                                      $23,250.00
 * JR PROGRAMMER/ANALYST                            $9,000.00
 * JR SALES REPRESENTATIVE                         $27,300.00
 * JR SYSTEMS PROGRAMMER                            $9,000.00
 * MAIL CLERK                                       $4,578.00
 * MANAGER ACCOUNTING                               $7,500.00
 * MANAGER DATA BASE ADMINISTRATION                $19,500.00
 * MANAGER EASTERN REGION MARKETING                $21,750.00
 * MANAGER PERSONNEL                               $15,000.00
 * MANAGER PUBLIC RELATIONS & EDUCATION            $15,000.00
 * MANAGER SYSTEMS DEVELOPMENT & MAINTENANCE       $20,250.00
 * MANAGER SYSTEMS OPERATIONS & RESOURCES          $19,500.00
 * MANAGER WESTERN REGION MARKETING                $21,750.00
 * OFFICE SUPERVISOR                               $11,625.00
 * OPERATIONS SUPERVISOR                           $15,000.00
 * PR & ADVERTISING SPECALIST                      $15,000.00
 * PRESIDENT                                       $39,800.00
 * PROGRAMMER                                       $9,450.00
 * SECRETARY                                       $35,778.92
 * SR SALES REPRESENTATIVE                        $157,725.00
 * SR SYSTEMS ANALYST                              $11,250.00
 * SR SYSTEMS PROGRAMMER                           $24,375.00
 * STANDARDS & PROCEDURES ANALYST                  $16,500.00
 * SUPPLY CLERK                                     $4,578.00
 * SYSTEMS ANALYST                                 $13,500.00
 * TECHNICAL WRITER                                $10,500.00
 * VICE-PRESIDENT ADMINISTRATION & FINANCE         $17,250.00
 * VICE-PRESIDENT INFORMATION SYSTEMS              $26,250.00
 * VICE-PRESIDENT MARKETING                        $24,750.00
```

Example 6:

In this last example, we roll the year-to-date salary information up to the department level. The monthly salaries for each employee are added together and grouped by department.

```
SELECT /TITLE(79)
D(10) YEAR TO DATE SALARIES BY DEPARTMENT,
L(24)DEPARTMENT,R(13)YTD SALARY/
C102, SUM(C124),
GROUP BY C102
WH C121 SPANS 01/01/1991*12/31/1991 AND C124 EXISTS:
```

```
1          YEAR TO DATE SALARIES BY DEPARTMENT
                       07/20/2004

 * DEPARTMENT                    YTD SALARY
 ***
 * ADMINISTRATION & FINANCE       $56,254.20
 * EXECUTIVE                     $166,470.36
 * INFORMATION SYSTEMS           $197,137.50
 * MARKETING                     $288,532.36
```

Chapter 5: SCF 64-Bit RELOAD and High-Speed Index Processing

SCF 64-Bit RELOAD

Version 2 provides a new high-performance SCF single-user reload capability. This feature enables you to reload a database, with keys, in about a third of the time it takes for a PLEX load without keys. After an SCF reload, your database, including indexes, is in optimal order. To use the new feature, you open a database using single-user SCF and issue the command RELOAD.

Rebuilding a large database with earlier SYSTEM 2000 releases usually requires a PLEX unload/load program. Because of sort and pad file disk space requirements, you have to load the database without keys and then use SCF CREATE INDEX to establish keys for selected items. With Version 2 you can achieve the same results, and in much less time, using only the SCF RELOAD command. This performance feature is available only in single-user mode. If you issue the RELOAD command in Multi-User, the old process is invoked.

RELOAD and System Sort Used Together

There are two distinct parts of the new reload. One is the 64-bit reload, which copies database Files 1, 3, 5, and 6 above the bar and uses above-the-bar data to reload the data base. Each file is arranged sequentially, which allows direct and rapid addressability to records without concern for paging. This addressing technique is unaffected by database skewness or records spanning page boundaries and contributes significantly to improved performance.

The second part of the reload process described in the next section, "High-Speed Index Processing," uses the system sort (such as DFSORT) for the indexes. Maximum performance is achieved when using both parts, and we recommend that both be enabled for a reload.

MAP-Directed RELOAD

SCF 64-bit reload also applies to MAP-directed reloads. Now you can use DEFINE to add or delete records and items and to change item attributes without the inconvenience of first unloading and then loading the database after the definition change. Above-the-bar storage and use of system sort enable you to make definition changes without concern for pad file and SYSTEM 2000 sort file space allocations. A MAP-directed reload automatically invokes the SCF 64-bit reload process unless you have disabled the process with the new execution parameter MEM64.

Operating System Considerations

SYSTEM 2000 use of the 64-bit memory requires operating system z/OS V1R3 or later. Above-the-bar storage, introduced in z/OS V1R2, is 16 exabytes, which is far more than required for even the largest database. Although there is seemingly no practical limit to virtual storage above the bar, limits do exist for the real storage frames and auxiliary storage that back the area. This means that your site might set limits on the amount of above-the-bar storage your job can use. Even if your job is allocated to above-the-bar storage, inadequate auxiliary storage paging space can adversely affect your entire system. The JES log might indicate “auxiliary storage shortage,” and logon, mount, and start commands might be rejected until the shortage is relieved.

If memory is not available, message -712- is issued and the job terminates; the database is not damaged. You should also ensure that the new sort technique for keys is enabled before invoking reload. Otherwise, you might get a B37 ABEND on the pad or SYSTEM 2000 sort files and damage your database.

Disabling SCF 64-Bit Reload

A new parameter, MEM64, is used to disable 64-bit reload. The default is MEM64=YES. Therefore, you do not have to do anything to enable the feature. You can disable the new feature by specifying MEM64=NO. In Multi-User, only for the SCF command reload, MEM64=NO regardless of what you specify.

High-Speed Index Processing

High-speed index processing reduces elapsed and CPU times. It also eliminates usage of pad and sort files (DDN SF01 – SF06) for the index process. This feature is for a single-user environment, and it is available when running the following commands:

```
SCF LOAD
SCF RELOAD
SCF CREATE INDEX
SCF MAP-Directed Reload
PLEX Optimized Load
```

Key value sort records are written to a dynamically allocated QSAM file (DDN SORTIN/SORTOUT). Your system sort (such a DFSORT) sorts the data. SYSTEM 2000 execution parameters determine whether your system sort is to be invoked, and the unit device and space needed for the files. SYSTEM 2000 dynamically allocates SORTIN and SORTOUT files unless you specify those DD names in JCL. The execution parameters are as follows:

SSFLAG=YES | NO

This parameter indicates whether high-speed index processing is to be used. The default is NO.

SSUNIT=xxxxxxxx | SYSDA

This parameter specifies what unit device is to be used when dynamically allocating the SORTIN/SORTOUT file. For large volume processing, we recommend a cartridge unit. The default is SYSDA.

DISK:

SYSTEM 2000 dynamically allocates one temporary data set for use by both SORTIN and SORTOUT. Alternatively, you can allocate the data set and bypass dynamic allocation. The following are example JCL statements for a disk data set:

```
//SORTIN     DD DSN=my.dataset.name,DISP=(NEW,DELETE,DELETE),
//                             UNIT=SYSDA,SPACE=(CYL,(10,10))
//SORTOUT DD DSN=*.SORTIN.DISP=SHR,VOL=REF=*.SORTIN
```

TAPE/CARTRIDGE:

Due to restrictions in dynamic allocation, two data sets and two tape or cartridge drives are necessary if dynamic allocation is used. SYSTEM 2000 dynamically allocates a temporary data set for SORTIN and a second temporary data set for SORTOUT. The data set retention period is one day. You can allocate the data set using JCL, which will bypass dynamic allocation and use only one data set and one tape/cartridge unit. The following are example JCL statements for a cartridge data set:

```
//SORTIN     DD DSN=my.dataset.name,DISP=(NEW,DELETE,DELETE),
//                          UNIT=CART
//SORTOUT DD DSN=*.SORTIN,DISP=(NEW,DELETE,DELETE),
//                          UNIT=AFF=SORTIN,VOL=REF=*.SORTIN
```

SSPRI=nnnn | 500

Output of the EXAMINE utility shows the correct value for this parameter. SSPRI is ignored when SSUNIT indicates tape or cartridge.

This parameter specifies the primary allocation, in cylinders, for dynamic allocation of the SORTIN/SORTOUT file. It only applies if SSUNIT specifies a disk device. The default is 500 cylinders.

SSSEC=nnnn | 125

Output of the EXAMINE utility shows the correct value for this parameter. SSSEC is ignored when SSUNIT indicates tape or cartridge.

This parameter specifies the secondary allocation, in cylinders, for dynamic allocation of the SORTIN/SORTOUT file. It only applies if SSUNIT specifies a disk device. The default is 125 cylinders.

SSMXV=nnn | 250

This parameter specifies, in bytes, the longest key value in the database being processed. The default is 250 bytes.

SSMXV is used to set the record length of SORTIN and should, therefore, identify length of the longest key value. SSMXV is used for SCF LOAD, CREATE INDEX, and PLEX LOAD; it is ignored for RELOAD and MAP-directed (define changes) reload. Output of the EXAMINE utility shows the longest key value and number of key value occurrences.

Space calculation for disk are derived by the following formula:

(Longest value =20) * total occurrences of key values = bytes needed/SORTIN Blksize = blks needed. The blksize for disk is 27978.

In order to help you determine the values for the SSMXV, SSPRI, and SSSEC parameters, you can run the EXAMINE utility against your database(s). The output from the EXAMINE utility contains three lines that show recommended values for these parameters.

High-speed index processing is intended primarily for use with large volumes of data. You can use the process with small databases or databases with few indexes, but there might not be a performance improvement.

Databases above the Bar

In the 31-bit address space, a virtual line marks the 16-megabyte address. The 64-bit address space also includes a second virtual line called the *bar*. The bar separates storage below the 2-gigabyte address, called *below the bar*, from storage above the 2-gigabyte address, called *above the bar*. The area above the bar is a data area. Programs do not run above the bar.

As of z/OS R2, an address space begins at address 0 and ends at 16 exabytes, which is a 64-bit address. The space is slightly more than one billion gigabytes. It is 8 billion times the size of the former 2-gigabyte address space that logically has 2^{31} addresses. The decimal number is 16,000,000,000,000,000,000 bytes, or 16 exabytes (that is the number 16 followed by 18 zeroes). The address space structure below 2-gigabytes has not changed; all programs in AMODE24 and AMODE31 continue to run without change.

SYSTEM 2000 Version 2 allows database Files 5 and 6 to be processed above the bar, which provides a tremendous improvement in performance. Data is stored in memory and on secondary storage devices that are managed by system paging. This allows access nearly as rapid as if all of the data was retained in real memory.

With 64-bit addressing, the files do not use POOL buffers. Altered data is written to VSAM files significantly less often, and is done only when necessary to preserve the data, specifically at physical close of the database, reset of the rollback log, and save of the database. Data access is unaffected by file skewness or records that span page boundaries. The reduction of buffer pool usage can have a ripple effect. More buffers are available for other files and there are fewer buffer steals, which reduces I/O.

If criteria for above the bar usage is met, all pages “in use” for Files 5 and 6 are loaded above the bar at open time. Above-the-bar memory is never expanded after open. INSERT TREE and APPEND TREE use reusable space above the bar; if there is no reusable space, data is processed by the buffer manager (just as in previous releases). If the last page in 64-bit memory is only partially filled, newly inserted data uses the remaining space. If the record does not fit, it is continued in a 31-bit buffer and processed by the buffer manager.

Newly inserted trees use 64-bit memory only when using reusable space, or when being added to the last and only partially filled page that is above the bar. All changes to, and retrievals of, data that existed when the database was opened use 64-bit memory.

Restore and SCF Reload with 64-bit Addressing

Restore, both single user and Multi-User, does not open a database for 64-bit processing. Only SCF DBN and PLEX OPEN commands cause Files 5 and 6 to be loaded above the bar. The first DBN or OPEN invoked after the restore move the files above the bar.

Reload does not open a database for 64-bit processing. Even in single user, when the database is loaded above the bar and high performance reload is invoked, the 64-bit memory is released. DBN or OPEN reloads the files above the bar.

Above-the-Bar Usage Restrictions

SYSTEM 2000 usage of above-the-bar memory is regulated in two ways. The first is system wide with a new execution parameter, MEM64, which must be set to MEM64=YES in order for SYSTEM 2000 to use above-the-bar

memory. The second is by database with an entry in the S2KDBCNT table. Three new fields were added to S2KDBCNT for V2; a Y in a new field indicates that the option is to be invoked. Anything other than Y is accepted as NO and does not invoke the option.

Field	Beginning Column	Length	Description
CNTABVBR	67	1	Database files above the bar. Y qualifies the database for above-the-bar memory, but only if the MEM64 parameter allows above-the-bar usage.
CNTOPNMU	68	1	Open this database during Multi-User initialization. This option is not related to 64-bit memory.
CNTOPEN	69	1	Open 31-bit processing if the request for above-the-bar memory fails.

At open time, Files 5 and 6 are copied above the bar if execution parameter MEM64=YES and S2KDBCNT field CNTABVBR=Y. If the SYSTEM 2000 request for above-the-bar memory fails, or a GETMAIN for map of above-the-bar memory fails, the database is opened for 31-bit processing if CNTOPEN=Y. Otherwise, the database is flagged as offline in S2KDBCNT. You can vary the database online and specify MEM64=N to cause S2KDBCNT field CNTABVBR to be changed to N (e.g., VARY EMPLOYEE ONLINE, MEM64=N).

VARY <db> ONLINE

The VARY ONLINE command was modified to enable you to specify MEM64=YES/NO. The new operand sets CNTABVBR in S2KDBCNT to Y or N as specified in operand MEM64. MEM64 is an operand used with the VARY command. The operand is used to specify a Y or N value for field CANABVBR in file S2KDBCNT. The operand has no effect on execution parameter MEM64.

VARY EMPLOYEE ONLINE, MEM64=N	flags the database as online and not qualified for 64-bit memory.
VARY EMPLOYEE ONLINE, MEM64=Y	flags the database as online and qualified for 64-bit memory.
VARY EMPLOYEE OFFLINE, MEM64=N	flags the database as offline and not qualified for 64-bit memory.
VARY EMPLOYEE OFFLINE, MEM64=Y	flags the database as offline and qualified for 64-bit memory.

Operating System Requirements

In many cases, storage for the vast address range available above the bar is a mixture of real and secondary storage and is less in capacity than the address range. System programmers allocate storage volume for system paging based on anticipated usage. Consequently, there is a serious negative effect on the entire computer system when usage of 64-bit addresses exceeds intended limits of the paging system. See “Operating System Considerations” in Chapter 5 of this document.

Chapter 6: Migration to Version 2

To migrate to Version 2, you must do a complete installation using the new document *SYSTEM 2000 V2 Basic, Multi-User, QueX, and Interface to CICS: Installation Guide* shipped with your Version 2 software. After you have installed Version 2 and executed the validation programs, you can use the following information as a guide for migrating your current databases and applications. In all cases, be sure to contact SYSTEM 2000 Technical Support if you have questions or want to discuss your upgrade to Version 2.

Review User Exits and Special Zaps

Be sure that the following changes have been made:

- All your user exits have been copied into the V2 LOAD library.
- The V2 version of S2KEXIN has been modified to include all the user exits that you want.
- S2KEXIN has been reassembled and relinked into your V2 LOAD library.
- Any SYSTEM 2000 special zaps that your site requires have been applied to your V2 system. Special zaps for earlier releases of SYSTEM 2000 might no longer apply in V2. Whatever function the zap performed in earlier releases might now be a permanent change, or it might be invoked by using an OPT*nnn* execution parameter. Check with the SYSTEM 2000 Technical Support staff to find out if you still need special zaps.

Modify Multi-User and Single-User Parms Files

Review the SYSTEM 2000 parms files and give special attention to new parms and to existing parms that might have changed.

For example, the POOL parameter specifications have the following changes:

- POOL0 through POOL6 are for database usage, and POOL7 is for work files.
- The S2KUSERS file has a required CISIZE of 12288. Be sure to specify one POOL parameter that has this
  CISIZE.
- The PADUNIT parameter should be removed if it is in your current parms file(s) because it is no longer used.

See Chapter 5 in the *SYSTEM 2000 Software: Product Support Manual* for detailed information about buffer pools. See Chapter 15 for details about execution parameters.

Reassemble XBUF Macros

If you are using the XBUF caching feature, you need to make some changes and reassemble the macros.

- The BLKSIZE parameter of the XBMEMORY macro has been changed to CISIZE.
- You must change the BLKSIZE values to the new V2 CISIZE values.
- After you make these changes, reassemble and relink XBUFTBL into your V2 LOAD library.

See Chapter 13 in the *SYSTEM 2000 Software: Product Support Manual* for details about implementing XBUF.

S2KDBCNT Table

If you have changed the data set names of your SYSTEM 2000 database files and you are using the S2KDBCNT table, be sure to modify the table to reflect the new names. Additionally, you might want to add Y or N to Columns 67, 68, and 69. See “Above-the-Bar Usage Restrictions” in Chapter 5 of this document.

67 - Open this database for 64-bit processing.

68 - Open this database during Multi-User initialization.

69 - Open for 31-bit processing (as in previous versions) if acquisition of above-the-bar memory fails.

Allocate Database Files

Use job JCLALLOC to create new V2 database files for each database to be used in the production environment. Pay attention to the CISIZE and RECORDSIZE parameters of the file definitions. The RECORDSIZE value must be 7 bytes less than the CISIZE value. See the table below for valid CISIZE and RECORDSIZE values.

Valid Values for CISIZE and RECORDSIZE

CISIZE	**RECORDSIZE**
4096	4089
7168	7161
12288	12281
18432	18425
22528	22521
26624	26617

If you are using a value other than one of those shown in the table as your current database page size, you can determine how many pages will be used for each database file when they are converted to V2. For example, if your current database blocksize is 6216 and your V2 CISIZE is 7168, and File 6 of your database contains 1000 pages, calculate as follows:

1. Divide 6216 by 7168 (to get the percentage difference in size).

   6216 / 7168 = .8671875 (round to .87)

2. Now multiply 1000 (the number of pages used currently by File 6) by .87.

   1000 · .87 = 870

The V2 version of File 6 will use 870 pages. This same formula can be used for each database file. See Chapter 3 in the *SYSTEM 2000 Software: Product Support Manual* for details about database file capacities and CISIZE.

Convert Existing Databases to Version 2

Create Savefiles of current production databases, and execute program CVRTV2 to convert the production Savefiles to V2 databases. See Chapter 15 in the *SYSTEM 2000 Software: Product Support Manual* for details about executing CVRTV2. Make sure to change V1 to V2 in the JCLCNVRT job. You can also use the PLEX or SCF UNLOAD and LOAD process to convert your databases.

Your Turn

We want your feedback.

- If you have comments about this book, please send them to **`yourturn@sas.com`**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **`suggest@sas.com`**.