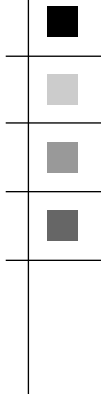




INSTALLATION INSTRUCTIONS

**Installation Instructions for the
SAS/C® 7.50 Software for
z/OS and OS/390**





INSTALLATION INSTRUCTIONS

**Installation Instructions for the
SAS/C[®] 7.50 Software for
z/OS and OS/390**



The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *Installation Instructions for the SAS/C® 7.50 Software for z/OS and OS/390*, Cary, NC: SAS Institute Inc., 2004.

Installation Instructions for the SAS/C® 7.50 Software for z/OS and OS/390

Copyright © 2004 SAS Institute Inc., Cary, NC, USA.

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, by any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

Limited permission is granted to store this copyrighted material in your system and display it on terminals, to print only the number of copies required for use by those persons responsible for installing and supporting the licensed SAS programs for which this material has been provided, and to modify the material to meet specific installation requirements. The SAS Institute copyright notice must appear on all printed versions of this material or extracts thereof, and on the display medium when the material is displayed. Permission is not granted to reproduce or distribute the material except as stated above.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of the software by the government is subject to restrictions as set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

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

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Table of Contents

Chapter 1: Introduction	1
Chapter 2: Installing SAS/C Data Sets	3
Extracting the Install JCL from your Installation Medium	3
Modifying the Install JCL	4
Submitting the Install JCL	5
Continuing the Installation	5
Chapter 3: Customizing and Transferring Cataloged Procedures and CLISTs	7
Tailoring the Customization JCL	7
Submitting the Customized JCL	9
Installation JCL for Cataloged Procedures and CLISTs	10
Chapter 4: Customizing SAS/C Installation	11
Making the Transient Library Available to C Programs	11
Customizing SAS/C Help Information	14
Installing Load Modules into LPA	14
Customizing Temporary File Defaults	15
Customizing Dynamic Allocation Defaults	16
Customizing Data-in-Virtual Defaults	17
sascc.cfg Configuration File	18
Chapter 5: Customizing ISPF Panel Support	21
Installing the SAS/C ISPF Dialogs Using Standard ISPF Libraries	21
Modifying Standard Dependent ISPF Files	21
Modifying SAS/C ISPF Dialogs for HyperText Tools	23
Modifying SAS/C ISPF Dialogs for C++ Tools	24
Chapter 6: Customizing TSO Support	25
TSO Command Processor Support	25
Logic for SAS/C EXEC and C Modules	26
Installing SAS/C Environment Variable Commands	27
Chapter 7: Customizing CICS Library Support	29
Making the CICS Transient Library Available to CICS	29
Defining the CICS Transient Library Programs in the CICS PPT	29
Defining the SAS/C Debugger Front-end Transaction in the CICS PCT	31
Defining CICS Extrapartition Queues in the CICS DCT	31
Allocating the CICS Extrapartition Transient Data Queues	32
Modifying the CICS Start-up JCL	33
Changing Default Names for CICS Modules and TD Queues	33
Chapter 8: Customizing Default Character Representation Tables	35
Modifying the Default Character Set Representation Tables	35
Translating Header Files to an Alternate Character Set	36
Providing Character Translation for C++ Tracebacks	37
Chapter 9: Customizing TCP/IP Defaults	39
Specifying the High-Level Prefix for TCP/IP Data Sets	39
Configuring the SAS/C Library to use the IBM z/OS Name Resolver	40
Chapter 10: Customizing CSL NFS Defaults	41
NFS Security Considerations	43
Chapter 11: Installing UNIX System Services (USS) OS/390 SAS/C Components	45
Chapter 12: Validating Installation	47

SAS/C Compiler Validation	47
SAS/C C++ Development System Validation	47
Validation Notes.	47
Chapter 13: Maintenance of SAS/C Products	49
SAS/C Code Maintenance	49
Product License Maintenance (SETINIT)	49
System Maintenance.	50
Appendix A: Dumping the Limited Distribution Library to Tape	51
Appendix B: Dumping the SAS/C CSL Limited Distribution Library to Tape	53
Appendix C: List of Installed Data Sets.	55
SAS/C Compiler Data Sets.	55
SAS/C Resident Library Data Sets.	57
SAS/C FSSL Data Sets.	60
SAS/C C++ Data Sets.	61
SAS/C CSL Data Sets	62
SAS/C Usage Notes Data Sets	64
Appendix D: Reference Publications	65

Using This Book

Purpose

This document describes the installation and customization of the SAS/C Software. Depending upon which products a site has licensed, the installation media may have any combination of the SAS/C Software (for example, just the SAS/C Compiler, the Compiler plus the FSSL Library, or the SAS/C Compiler and the SAS/C C++ Development System). The installation process has the flexibility to load to disk any combination of products on the media. Since "Introduction" on page 1, serves as an introduction and as an outline of the install process, we ask installers to please read all of the chapter before beginning the installation.

Additionally, this guide contains valuable information on product maintenance and other topics that will be useful in the future. Please keep this document for later reference.

Note: For the purpose of this documentation, z/OS, OS/390, and MVS are synonymous terms.

Products

The products that may be installed with these instructions are:

Release	Products
7.50	SAS/C Compiler SAS/C Resident Library SAS/C FSSL SAS/C C++ Development System SAS/C CSL

Conventions

You will see several type styles in this book. Style conventions are summarized here:

Convention	Description
roman	is the basic type style used for most text in this book.
UPPERCASE ROMAN	is used for OS/390 commands and data set names.
monospace	is used for C and C++ language elements in the text and in programming code. It is also used for workstation commands and filenames.
UPPERCASE MONOSPACE	is used for C and C++ language elements that you must enter in uppercase.
<i>italic</i>	indicates one of two things. In regular text an italicized word means that it is important, or is being defined. In examples italics means to substitute your own value for an item.

Federal Government Rights Notice

If your installation is a Federal Government site or a Federal Government Prime Contractor site, you are responsible for the information contained in a usage rights notice that has been included on the installation media. Please review the Government Rights Notice information contained in the GOVNOTIC member of the *prefix*.CNTL data set as soon as the installation is completed.

1 Introduction

These instructions cover a variety of SAS/C Software product media. Depending on what products a site has licensed, the product medium may contain any combination of the SAS/C Software (for example: just the SAS/C Compiler, the Compiler plus the FSSL Library, or the SAS/C Compiler and the SAS/C C++ Development System). Although all SAS/C Software product media do not contain the same products, the basic install process will be the same.

Every installation consists of the following steps:

1. Creating and submitting the JCL to extract the install JCL jobstream. See “Extracting the Install JCL from your Installation Medium” on page 3.
2. Tailoring and submitting the install jobstream to unload the products from your media. See “Modifying the Install JCL” on page 4.
3. Customizing and transferring catalogued procedures, CLISTS, and REXX execs. See “Customizing and Transferring Cataloged Procedures and CLISTS” on page 7.
4. Optionally, customizing SAS/C installation. See “Customizing SAS/C Installation” on page 11.
5. Optionally, installing the UNIX System Services (USS) OS/390 SAS/C components into the hierarchical file system (HFS). See “Installing UNIX System Services (USS) OS/390 SAS/C Components” on page 45.
6. Validating the installation. See “Validating Installation” on page 47.
7. Optionally, applying maintenance. See “Maintenance of SAS/C Products” on page 49. SAS/C products are shipped without maintenance applied.

The installation process will install a unique set of data sets for each SAS/C Software product. See Appendix C on page 55 for a list of all SAS/C Software products and the corresponding installed data sets.

Note: To ensure all licensed products are included on this medium, refer to the Transmittal Letter for OS/390 sent to the C Compiler Software Consultant at your site.

2

Installing SAS/C Data Sets

Installing the SAS/C data sets consists of the following steps:

1. Using the IEBUPDTE utility to extract the install JCL from your installation medium.
2. Modifying the install JCL.
3. Submitting the install JCL.

Extracting the Install JCL from your Installation Medium

The job CUPDATE has been provided for allocating and installing all of the data sets needed for the SAS/C Software products. Extract the CUPDATE job and the other members used as input to CUPDATE directly from the first file on your installation medium using the IEBUPDTE utility program below.

```
//IEBUPDTE JOB account-code , 'programmer-name' ,           <===MODIFY
//                MSGCLASS=A,MSGLEVEL=(1,1) ,
//                TIME=(,9)
//*
//UPDTE          EXEC PGM=IEBUPDTE, PARM='NEW'
//SYSIN          DD DSN=SASC.UPDT,DISP=(OLD,PASS) ,UNIT=uuuu , <===VERIFY
//                DCB=BLKSIZE=32720,LABEL=(1,SL) ,
//                VOL=SER=volser                               <===VERIFY
//SYSPRINT       DD SYSOUT=*
//SYSUT2         DD DSN=your.dataset.name ,                 <===VERIFY
//                DISP=OLD
//*
//
```

The first three lines of the IEBUPDTE job contain the job cards to be used for executing the job. You will need to:

1. Modify the *account-code* and *programmer-name* values to reflect those used by your site. Additional values that must be changed are denoted by <=== VERIFY to the right of the JCL line.
2. Replace *uuuu* with the particular media type on which you received the SAS/C Software products. If your installation medium is a 4mm tape, replace *uuuu* with the device address of your 4mm tape drive.
3. Replace the *volser* value with the volume serial number of the SAS/C Software product medium to be installed.
4. Make sure that the partitioned data set *your.dataset.name* exists and has at least 60 blocks of available space. The data set *your.dataset.name* on the SYSUT2 DD is used to install the CUPDATE job and other members containing IEBCOPY control information. Suggested DCB characteristics include RECFM=FB, BLKSIZE=3120, and LRECL=80.

After creating and tailoring the IEBUPDTE job for your site requirements, submit the job to extract the install JCL from the installation medium.

Modifying the Install JCL

The first three lines of the CUPDATE job, member CUPDATE, extracted from the installation medium in “Extracting the Install JCL from your Installation Medium” on page 3, contain the job card to be used for executing the job. Modify the *account-code* and *programmer-name* values to reflect those used by your site. Additional job cards have been provided for you to update with the appropriate /*JOBPARM, /*ROUTE, /*MAIN, or /*FORMAT values for your site, if needed.

The CUPDATE job consists of an in-stream procedure with symbolic parameters to help minimize the amount of editing required. The symbolic values to be modified before executing the job are contained at the beginning of the in-stream procedure and are denoted by <=== VERIFY to the right of the symbolic values. The symbolic values in the CUPDATE job are as follows:

```
//CUPDATE PROC PREFIX='SASC.C750',           <===VERIFY
//          UPREFIX='SASC.C750',             <===VERIFY
//          DISKVOL=XXXXXX,                 <===VERIFY
//          UNIT=DISK,                       <===VERIFY
//          EXPDATE=2009/365,                <===VERIFY
//          INSTALL='your.dataset.name',     <===VERIFY
//          TAPEUNT=TAPE
```

Modify the symbolic values using the following parameter descriptions as guidelines.

Do not modify the value for TAPEUNT, it has been customized for your site.

TABLE 1.

Parameter	Description
PREFIX=	specifies the high-level prefix to use for the SAS/C Software data set names. (This value may be more than one level. For example, xxx.yyy.) For example, change PREFIX=SASC.C750 to PREFIX=SASC.C750.COMP if you want the SAS/C compiler load module library to be named SASC.C750.COMP.LOAD. Note: the PREFIX parameter should remain the same when installing the SAS/C Compiler, the SAS/C Resident library, the FSSL library, the C++ Development System, or the SAS/C Connectivity Support Library (CSL).
UPREFIX=	specifies the high-level prefix to use for the SAS/C maintenance utilities. (This value may be more than one level, e.g. xxx.yyy).
DISKVOL=	specifies the volume serial of the disk pack on which the data sets are to be created.
UNIT=	specifies the unit type of the storage device on which the data sets are to reside.
EXPDATE=	specifies the expiration date value used in accessing the installation medium.
INSTALL=	specifies the data set into which you installed the CUPDATE job and other input members (SYSUT2 DD card of IEBUPDTE in “Extracting the Install JCL from your Installation Medium” on page 3). The other input members are used as input in the CUPDATE job.

Note: Always use the CUPDATE job that is contained on the current installation medium. CUPDATE jobs retained from previous installations may be different.

Submitting the Install JCL

After modifying the JCL according to your site requirements, submit the CUPDATE job. The data sets are then loaded from the installation medium using one of the following IBM utilities: IEBCOPY, IEBGENER, and IEBUPDTE.

The CUPDATE job uses step condition codes to verify the successful execution of previous steps. If the condition code from any step is greater than zero, execution is halted. Possible reasons for nonzero condition codes are space allocation problems, incorrect JCL specifications, and so on. After the problem has been diagnosed, the job can be restarted on the problem step. If you need a complete rerun of CUPDATE, ensure that all data sets that were allocated or cataloged in previous executions have been deleted.

Note:

- The space for the data sets to be allocated is specified in blocks so that the allocation is device-independent.

Continuing the Installation

See “Customizing and Transferring Cataloged Procedures and CLISTS” on page 7 for more information on tailoring your cataloged procedures and CLISTS.

3

Customizing and Transferring Cataloged Procedures and CLISTs

Customizing cataloged procedures and CLISTs consists of:

- Tailoring the customization JCL.
- Submitting the customized JCL.
- Installing the cataloged procedures and CLISTs.

Tailoring the Customization JCL

The job SASCEDIT (member SASCEDIT of *prefix*.CNTL), which was loaded in “Submitting the Install JCL” on page 5, tailors all the JCL supplied in *prefix*.CNTL for use during further installation and customization of the SAS/C Software.

The EDIT step of the SASCEDIT job creates a copy of the SASCCNTL member of *prefix*.CNTL in temporary storage. This temporary copy is then input to IEBUPDTE to create an edited copy and then saved as members in *prefix*.CNTL. Like symbolic parameters in a JCL cataloged procedure, all occurrences of *%keyword* in the SASCCNTL member are replaced by the value in the corresponding *keyword=* parameter statement.

Before you can run job SASCEDIT, you must edit the symbolic parameters used for input. These parameters specify required alterations to the installation JCL and are located in the SASCEDTP member of *prefix*.CNTL.

It is imperative that you edit the symbolic parameters in SASCEDTP, edit SASCEDIT, and finally execute SASCEDIT before proceeding to the next installation step.

There are two formats for SASCEDTP symbolic parameters. The formats are

```
keyword=value    comment
```

or

```
* comment
```

In the first format *keyword* is a symbol defined in the JCL to be edited. *value* is a character string that either contains no blanks or is enclosed in apostrophes (' '). Note that no blanks can appear before the equal sign and that a blank following the equal sign assigns a null (that is, zero-length) value to the keyword. The second form of the parameter statement is a comment statement, indicated by an asterisk in column 1.

Note: Values assigned to keyword parameters can also be keyword parameters; therefore, symbolic replacement can take place in the default parameter statements. For instance, the parameter statements

```
VALUE1=ABCDEFGH
VALUE2=%VALUE1.
```

used on the following input line

```
SYMBOLIC REPLACEMENT OF %VALUE2. HAS OCCURRED.
```

would result in the output line shown below:

```
SYMBOLIC REPLACEMENT OF ABCDEFGH HAS OCCURRED.
```

Caution: Never use members or JCL from previous SAS/C software releases. The current release is listed on “Products” on page v.

Editing the SASCEDTP Member of prefix.CNTL

The SASCEDTP member of *prefix*.CNTL contains symbolic parameters which will tailor the SASCINST member of *prefix*.CNTL to modify the CLISTs, cataloged procedures, REXX execs, and ISPF skeleton library. To simplify editing of the SASCEDTP member, the symbolic parameters can be categorized into the following groups.

- job card information
- data set prefixes
- UNIX System Services (USS) OS/390 Hierarchical File System (HFS) installation location
- miscellaneous JCL parameters
- transient library name
- cataloged procedure names
- CLIST library names
- REXX exec library names
- ISPF skeleton library

Editing Job-Card Parameters in SASCEDTP

The JOBCARD1 through JOBCARD5 parameters create job cards for all the installation jobs affected by the JCL-tailoring process. Notice that the default JOBCARD1 value contains a jobname value of %SYSNAME. This symbolic value is replaced in the tailoring process by the name of the member being edited. If you want all installation jobs to have the same name, you can replace %SYSNAME with the value you desire. Update the remainder of the JOBCARDs with the appropriate /*JOBPARM, /*ROUTE, /*MAIN, or /*FORMAT values for your site.

Caution: Leave the job card values enclosed in quotes(" ").

Editing Data Set Prefixes in SASCEDTP

Changes made to the default data set prefixes in SASCEDTP will be reflected in the CLISTs, cataloged procedures, REXX execs, and ISPF skeletons. All references to data sets in a CLIST, cataloged procedure, exec, or skeleton are tailored to the values specified in SASCEDTP. Default values are as follows:

Default Data Set Prefixes	Description
PREFIX= ' SASC . C750 '	Data set name high level prefix of SAS/C Software data sets.
UPREFIX= ' SASC . C750 '	Data set name high level prefix of SAS/C maintenance data sets.
CICSLIB= ' h1q . SDFHLOAD '	If CICS support is being installed, this fully qualified data set name specifies the name of the CICS LOADLIB for use with the cataloged procedures and TSO CLISTs. This IBM supplied data set contains the command-level stubs used for linking CICS applications. Its name varies between CICS releases and probably has been modified locally on your system.
TDPREFIX= ' CICS . SASC '	If CICS support is being installed, this specifies the high level prefix for the CICS extra-partition transient data queues used by the SAS/C library.

Note: CICS support is only installed with the Compiler or Resident library products.

Editing the UNIX System Services (USS) OS/390 HFS Installation Location in SASCEDTP

Modify the DIR symbolic variable to change the job SASCOEMV, which installs the SAS/C USS OS/390 components into the Hierarchical File System (HFS).

Note: The value specified for the DIR parameter is case sensitive.

Default USS OS/390 HFS Installation Location	Description
DIR= ' /usr/local '	The location of the 'bin' installation directory for the SAS/C USS OS/390 components.

Note: USS OS/390 support is installed only with the Compiler product.

Editing Miscellaneous JCL Parameters in SASCEDTP

The miscellaneous JCL parameters section of SASCEDTP lists the symbolic variables which will tailor the DD statements in the cataloged procedures and CLISTS. See “Modifying the Install JCL” on page 4 for detailed descriptions of each of these symbolic variables. Modify these parameters in accordance with your site requirements.

Note: The PREFIX, DISKVOL, UNIT, and EXPDATE parameters must match the in-stream procedure (<===VERIFY values) specified in the CUPDATE job. If they do not, you will receive JCL errors.

Editing Library Names in SASCEDTP

The library names sections of SASCEDTP list the data set names of installed libraries containing cataloged procedures, CLISTS, and ISPF panel code. The *prefix*. PROCLIB, *prefix*. RD . PROCLIB, *prefix*. CLIST, and other libraries are always created NEW, unless you are installing an add-on product, such as the SAS/C FSSL or the SAS/C C++ Development System. The SASCINST JCL jobstream will install add-on catalogue procedures and CLISTS into existing libraries allocated by an installation of the SAS/C Compiler or SAS/C Resident library.

Caution: Installers are no longer provided renaming capabilities in SASCEDTP for individual CLISTS or cataloged procedures.

Submitting the Customized JCL

Submitting the customized JCL consists of the following steps:

1. Editing the SASCEDIT job
2. Submitting the SASCEDIT job

Editing the SASCEDIT Job

The first five lines of the SASCEDIT job contain the job cards to be used for executing the job. Modify the *account-code* and *programmer-name* values to reflect those used by your site. Additional job cards have been provided for you to update with any site specific information. As with the CUPDATE job, additional values that should be changed appear at the beginning of the in-stream procedure and are denoted by <=== VERIFY to the right of the symbolic values to be replaced.

The PREFIX= value specifies the high-level prefix where the SAS/C software data sets reside. Be sure that this value matches both the value in the CUPDTE job and in the SASCEDTP member of *prefix*.CNTL.

The TEMPUNI= value specifies the unit type for temporary disk data sets used for your site.

Submitting the SASCEDIT Job

Submit the SASCEDIT job (member SASCEDIT of *prefix*.CNTL) to tailor the SASCINST job. The SASCEDIT job will tailor the SASCINST job so that it may be submitted later to install cataloged procedures, CLISTs, REXX execs, and ISPF dialogs.

Installation JCL for Cataloged Procedures and CLISTs

The cataloged procedures, CLISTs, and REXX execs will be installed (using the IEBGENER utility) into the data sets specified in the SASCEDTP member of *prefix*.CNTL (see “Editing the SASCEDTP Member of *prefix*.CNTL” on page 8). If you rename any of the following: CLISTs, REXX execs, and ISPF dialogs, you will have to edit the panel, tutorial, skeleton, and message files to correspond to the new names of the members or these files will not behave correctly. You should avoid renaming these ISPF dialogs if at all possible.

Note: The *prefix*.DBGHELP library is **NOT** a TSO format help file, therefore it cannot be installed into SYS1.HELP or your SAS System Help Library. The *prefix*.DBGHELP data set can be made available to users by placing the *prefix*.CLIST member L\$DBGHELP in your system CLIST library.

Submitting the SASCINST Job

Submit the SASCINST job (member SASCINST of *prefix*.CNTL) to load the cataloged procedures, CLISTs, and REXX execs into the specified libraries.

4

Customizing SAS/C Installation

When customizing your SAS/C installation, you can choose to:

- Make the transient library available to C programs
- Customize SAS/C help information
- Install load modules into LPA
- Customize temporary file defaults
- Customize dynamic allocation defaults
- Customize data-in-virtual defaults

Making the Transient Library Available to C Programs

The SAS/C product includes several transient libraries, which provide dynamically loaded run-time support for programs compiled with the SAS/C Compiler. Two versions of the transient library are provided: one for CICS applications and the other for non-CICS applications.

In order to increase the efficiency of C programs, the SAS/C runtime library is divided into two parts: resident and transient. Resident library modules are linked in with the executable program. Transient library modules are loaded as needed at runtime. This library structure reduces the memory requirements of C programs by including modules as needed during execution, instead of including all possibly required modules in the program at link-edit time.

Additionally, this organization allows transient library modules to be shared among many programs (if the modules are loaded into the LPA), which reduces real memory use and the I/O overhead involved in loading. Perhaps most importantly, this organization allows OS/390 C programs to run above the 16-megabyte line, while transient library modules that interface to I/O are loaded below the line (as required by OS/390).

In order to be available for loading at runtime, the appropriate transient libraries must be available on every system on which a SAS/C program is run. **To facilitate this process, the SAS/C transient libraries can be redistributed royalty-free to sites which use programs that were developed with the SAS/C product.**

Note: By default, most of the SAS/C transient library is loaded above the 16-megabyte line, but some pieces (for example, low-level I/O modules) have `AMODE 24` requirements and so must reside below the line.

Due to improvements in the SAS/C library, current versions of the runtime library are incompatible with `AMODE=24` applications linked using releases prior to release 5.00G. SAS Institute has provided a JCL jobstream for sites with a requirement to run these old load modules. This job relinks portions of the runtime library to reside below the 16-megabyte line. If your site requires compatibility with these old applications, edit the jobstream found in the `LIBRMD24` member of the `prefix.CNTL` data set and then submit the job. This job creates the `prefix.BTLLINK` data set, which is a copy of the SAS/C transient library with all necessary modules marked `RMODE=24`.

Using More than One Version of the Transient Library

You can create multiple versions of the transient library by changing the default prefix to one of your choosing. To do so, run the supplied REXX exec, `prefix.CNTL(LSCPRFX)`, which renames the transient modules and copies them to a new library. The new module names will begin with a three-letter prefix that you supply.

To cause an existing user program to load its transient modules from the new transient library, run JCL similar to following, replacing 'HELLO' on the name statement with the name of the user program:

```

//APPLYZAP      JOB
//              MSGCLASS=A
/*JOBPARM  FETCH
//ZAP          EXEC PGM=AMASPZAP
//SYSPRINT DD  SYSOUT=*
//SYSLIB      DD DSN=USER.LOAD,DISP=SHR
//SYSIN       DD *
NAME HELLO L$C$PFXO
VER 0004 4040,4000      '  '
REP 0004 D5C5,E600     'NEW'
/*
//

```

To cause user programs, when they are linked, to load transient modules from the new transient library, run JCL similar to the following:

```

//LIBZAP JOB
//              MSGCLASS=A
/*JOBPARM  FETCH
//ZAP          EXEC PGM=AMASPZAP
//SYSPRINT DD  SYSOUT=*
//SYSLIB      DD DSN=prefix.STDLIB,DISP=SH
//SYSIN       DD *
NAME L$C$PFX L$C$PFXO
VER 0004 4040,4000      '  '
REP 0004 D5C5,E600     'NEW'
/*
//

```

Finding the Transient Library

In “Submitting the Install JCL” on page 5, you copied the transient library into a disk data set. In order for C programs to find transient modules at runtime, you must make the modules available in one of the places where C programs search for transient modules. The search order is listed below. See “Making the CICS Transient Library Available to CICS” on page 29 for instructions on making the Transient Library available under CICS.

Search Order for Non-ISPF Users

1. Data sets allocated to CTRANS (if any)
2. Any defined tasklib data sets. For instance, when a SAS/C program is run under ISPF, the DDname ISPLLIB is defined as a tasklib.
3. STEPLIB (if any)
4. JOBLIB (if any)
5. The LPA (Link Pack Area)
6. Linklist libraries

Search Order for ISPF Users

1. CTRANS
2. tasklib (ISPLLIB)
3. STEPLIB
4. LPA
5. LINKLIST

Finding the Transient Library under the UNIX System Services (USS) Shell

For programs running under the USS Shell (or invoked via the USS exec function), the following search order is used.

1. A data set named by the ddn_CTRANS environment variable, if any.
2. Any data set defined by the STEPLIB environment variable.
3. The LPA (Link Pack Area)
4. Linklist libraries

If you do not place the transient library in the linklist or LPA and intend to use the USS shell, we recommend you modify `/etc/profile` to define the `ddn_CTRANS` variable to identify the transient library data set.

Also note that if you expect to run `setuid` or `setgid` programs written in SAS/C, you should put the library in either linklist or LPA. Alternatively, you can include the transient library in the STEPLIB environment variable list. In this case, you must also include the transient library data set name in the USS "sanction list" defined by the `BPXPRMnn` member of `SYS1.PARMLIB`.

You can make the transient library available to a C program using any of these locations. The following describes the considerations that apply to each location.

Note: If running under ISPF, a possible search order is listed below, but consult your ISPF documentation for your exact search path.

CTTRANS

Use of the CTRANS DDname is convenient for testing purposes when you are installing a new version of the transient library. Since C programs search the libraries allocated to CTRANS first, you can test a new version of the transient library while an older version remains installed. Sites that are evaluating a SAS/C program and may therefore want to remove the transient library at any time could also use CTRANS to access the transient library. CTRANS is particularly useful in TSO because, unlike STEPLIBs, CTRANS can be allocated dynamically and thus requires no modifications to logon procedures.

STEPLIBs or JOBLIBs

STEPLIBs or JOBLIBs are useful for sites that do not want to put the transient library in the LPA or linklist. They are also useful for testing and evaluation.

LPA

The LPA is the recommended location for SAS/C transient library modules. Installing the transient library into the LPA decreases memory requirements for C programs and reduces system paging if the C library is heavily used.

Installing the transient library into the LPA simplifies JCL and TSO procedures for calling C programs since CTRANS, JOBLIBs and STEPLIBs are not required. If you do not have enough virtual storage available to copy the entire library to an LPA library, it is recommended that you copy a select list of modules to the LPA library. A recommended minimal list of modules to place in the LPA library is `LSCARGE`, `LSCDIND`, `LSCDMGR`, `LSCOSEQ`, `LSCOTERM`, `LSCSIO`, `LSCIDDN`, `LSCIDSN`, and `LSCITSO`. The debugger load module `L$DEBUG` is also a very good candidate for inclusion if your site also has the SAS/C Compiler and makes frequent use of the debugger. Of course, any modules that are not copied into the LPA must be made available through one of the other locations (in this case, the linklist would usually be the most appropriate).

IMPORTANT: The modules `LSCEVRG` and `L$DCST` must NOT be moved into the LPA. Placement of these modules into LPA will cause execution errors.

LINKLIST

Depending on your local site procedures, you would make transient library modules available in the linklist either by naming the transient library data set (copied from the installation medium in “Submitting the Install JCL” on page 5) in the LNKLISTxx member of SYS1.PARMLIB or by copying the modules into some other library whose name is already in the linklist (perhaps SYS3.LINKLIB). This has none of the performance advantages of using the LPA but does not have the disadvantage of the LPA which decreases your system's available private area. It does share the advantage of making the transient library modules available without the need to code CTRANS, JOBLIBs, or STEPLIBs.

Customizing SAS/C Help Information

prefix.DBGHELP

The debugger help file *prefix*.DBGHELP is NOT a TSO format help file, but has a special internal format which the SAS/C Debugger uses. From an OS/390 standpoint, the file is a standard physical sequential file, it cannot be a member of a PDS, and it must have the following DCB attributes when it is copied or allocated:

```
DCB=(DSORG=PS,RECFM=FBS,LRECL=80,BLKSIZE=4080)
```

The *prefix*.DBGHELP can be made available to users by placing the *prefix*.CLIST member L\$DBHELP in your system CLIST library. When debugger help is requested and no DBGHELP DDname is allocated, the debugger attempts to invoke the L\$DBHELP CLIST to allocate the help file. The file tailoring procedures automatically update this CLIST to reflect the installation's name chosen for *prefix*.DBGHELP.

prefix.HELP

prefix.HELP needs to be made available to TSO by either concatenating *prefix*.HELP to the DDname SYSHELP or by copying the members to some larger general-purpose library.

Installing Load Modules into LPA

Sites that frequently run the compiler and have adequate virtual storage may elect to place the compiler and some of the utilities in the LPA. The following modules in the library *prefix*.LOAD are good candidates (ordered by value of inclusion): LC1370, LC2370, LCGO, COOL#, OMD370N, AR370#, LC370B, COOLB. The debugger load module L\$DEBUG is also a very good candidate for inclusion if your site makes frequent use of the debugger. This load module is installed into *prefix*.LINKLIB in “Submitting the Install JCL” on page 5.

Notes:

- Placing compiler and utility modules in the LPA is never as effective as placing the transient library modules in the LPA. We strongly recommend against doing this if the space required would prevent you from placing the transient library in the LPA.
- Only the compiler or utility modules that are marked reentrant should be placed in the LPA.
- Unpredictable errors will occur at program execution if you relink any members of the libraries.

Customizing Temporary File Defaults

This section describes a library zap that can be used by OS/390 sites that want to change the defaults for temporary file UNIT and SPACE. The default UNIT is VIO, and the default SPACE allocation is (TRK,(50,0)).

C programs that issue UNIX-style I/O requests use temporary files for work areas. If the SYSTMPnn DDnames are not provided, temporary files are dynamically allocated to UNIT=VIO. This causes a problem for sites without UNIT=VIO defined.

The following generic zap must be applied to the transient library (*prefix*.LINKLIB) using APPLYZAP (see *A Guide for the SAS/C Compiler Consultant* for details on using the APPLYZAP program) in order to change the temporary unit name or space allocation. Additionally, if you are using the all-resident library, the zap must be applied to *prefix*.ARESOBJ using APPLYZAP. All-resident programs must be relinked. Instead of relinking, you may apply the zap to the L\$C\$VIOO CSECT in the appropriate user load module. Note that you must change the module name in the NAME statement to refer to your user load module when applying the zap to a user program.

If you distribute software linked with the all-resident library, and the customer desires to change the defaults, you will need to distribute this zap to your customers.

```
*   NAME: Z7503101   PRODUCT: SASC       CATEGORY: SPEC   SYSTEM: MVS
*   DATE: 19JUL02   STATUS:  DZ+UT      USAGE-ID: LIBRARY-C3101
*
*   VIO as a unit name not defined at some installations
*
*   NOTE:           APPLY TO SASC.LINKLIB (LOAD MODULE FORM TRANSIENT
*                   LIBRARY), AND SASC.ARESOBJ (ALL-RESIDENT OBJECT FORM
*                   RUNTIME LIBRARY) USING THE APPLYZAP
*                   UTILITY.
*
*
*                   NOTE: THE USER MUST CODE THEIR OWN VALUES FOR UNIT,
*                   PRIMARY, AND SECONDARY SPACE.
*                   XXXXXX - PRIMARY TRACK ALLOCATION
*                   YYYYYY - SECONDARY TRACK ALLOCATION
*                   ZZZZZZZZ - UNIT NAME, must be right justified with
*                               blanks (0x40) to the full eight characters
*   END
NAME      L$CITMP  L$C$VIOO      LINKLIB ARESOBJ
CHECKSUM
VER       0000      0000,0032
VER       0004      0000,0000
VER       0008      E5C9,D640,4040,4040
*
REP       0000      00XX,XXXX
REP       0004      00YY,YYYY
REP       0008      ZZZZ,ZZZZ
IDRDATA  Z7503101
```

Note: You do not have to replace data that you are not changing. For example, to change the unit name to DISK in the transient library, use the following:

```
NAME L$CITMP L$C$VIOO      LINKLIB ARESOBJ
VER  0008 E5C9D640
REP  0008 C4C9E2D2
```

Customizing Dynamic Allocation Defaults

This section describes a zap that can be used by OS/390 sites that wish to change defaults for the dynamic allocation of new permanent data sets. (This is performed when “DSN:” or “TSO:” style file names are used by C programs calling `fopen()` or other library functions that accept a filename.) The current default is equivalent to `SPACE=(1000,(10,3,5))`. The following are examples of why you might wish to apply this zap:

- The default space allocation is too large or too small.
- Standards at your site call for use of a particular unit name or volume serial for such data sets.
- You wish to allocate data sets created by C programs using a specific SMS data class or storage class.

The following generic zap must be applied to the transient library (`prefix.LINKLIB`) using `APPLYZAP` (see *A Guide for the SAS/C Compiler Consultant* for details on using the `APPLYZAP` program) in order to change the default allocations. Additionally, if you are using the all-resident library, the zap must be applied to `prefix.ARESOBJ` using `APPLYZAP`. All-resident programs must be relinked. Instead of relinking, you may apply the zap to the `LCDSN=CSECT` in the appropriate user load module. Note that you must change the module name in the `NAME` statement to refer to your user load module when applying the zap to a user program.

If you distribute software linked with the all-resident library, and your customers need to change the defaults, you will need to distribute this zap to them.

Certain areas being zapped are applicable only to sites using SMS (IBM's Storage Management Subsystem). Do not change these fields if SMS is not installed and operational.

```
*   NAME: Z7504048   PRODUCT: SASC       CATEGORY: SPEC   SYSTEM: MVS
*   DATE: 19JUL02   STATUS:  DZ+UT      USAGE-ID: LIBRARY-C4048
*
*   How to change the defaults for alcunit=, space= and extend= amparms
*
*   NOTE:           APPLY TO SASC.LINKLIB (LOAD MODULE FORM TRANSIENT
*                   LIBRARY), AND SASC.ARESOBJ (ALL-RESIDENT OBJECT FORM
*                   RUNTIME LIBRARY) USING THE APPLYZAP UTILITY.
*
*                   NOTE: THE USER MUST CODE THEIR OWN VALUES FOR PRIMARY AND
*                   SECONDARY SPACE, AVERAGE BLOCK LENGTH, NUMBER OF
*                   DIRECTORY BLOCKS, FLAGS, UNIT NAME, VOLUME, DATA CLASS,
*                   STORAGE CLASS AND MANAGEMENT CLASS.  IF PADDING IS
*                   NEEDED, THESE VALUES SHOULD BE PADDED WITH BLANKS.
*
*                   IF SMS IS NOT INSTALLED AND ACTIVE, DO NOT INSTALL A DATA
*                   CLASS, STORAGE CLASS OR MANAGEMENT CLASS, AND DO NOT SET
*                   THE 08, 04, OR 02 FLAG BITS.
*
*                   XXXXXX - PRIMARY TRACK ALLOCATION
*                   YYYYYY - SECONDARY TRACK ALLOCATION
*                   LLLLLL - AVERAGE BLOCK LENGTH
*                   NNNNNN - NUMBER OF DIRECTORY BLOCKS
*                   FF      - FLAGS (80=TRACKS, 40=CYLINDERS, 20=BLOCKS
*                               08=RECORDS, 04=KILO-RECORDS,
*                               02=MEGA-RECORDS.  SET THE 10 BIT TO
*                               DEFAULT RLSE=YES.)
*                   UUUUUUUUUUUUUUUU - UNIT NAME
*                   ZZZZZZZZ - VOLUME
*                   DDDDDDDD - DEFAULT DATA CLASS
*                   SSSSSSSS - DEFAULT STORAGE CLASS
*                   MMMMMMMM - DEFAULT MANAGEMENT CLASS
*
*   END
```

```

NAME      L$CIDSN  L$C$DSN=      LINKLIB  ARESOBJ
CHECKSUM
VER       0008      0000,0A
VER       000B      0000,03
VER       000E      0003,E8
VER       0011      0000,05
VER       0014      20
VER       0015      0000,0000,0000,0000
VER       001D      0000,0000,0000
VER       0023      0000,0000,0000,0000
VER       002B      0000,0000,0000,0000
VER       0033      0000,0000,0000,0000
*
REP       0008      XXXX,XX
REP       000B      YYYY,YY
REP       000E      LLLL,LL
REP       0011      NNNN,NN
REP       0014      FF
REP       0015      UUUU,UUUU,UUUU,UUUU
REP       001D      ZZZZ,ZZZZ,ZZZZ
REP       0023      DDDD,DDDD,DDDD,DDDD
REP       002B      SSSS,SSSS,SSSS,SSSS
REP       0033      MMMM,MMMM,MMMM,MMMM
IDRDATA  Z7504048

```

Note: You do not have to replace data that you are not changing. For example, to make the default space allocation for new permanent data sets two cylinders of primary space and one cylinder of secondary space, use the following:

```

NAME  L$CIDSN  L$C$DSN=      LINKLIB  ARESOBJ
VER   0008   00000A
VER   000B   000003
VER   0014   20
REP   0008   000002
REP   000B   000001
REP   0014   40

```

Customizing Data-in-Virtual Defaults

This section describes a zap that is useful for OS/390 sites that wish to change the default values used by the library in its algorithm for buffering Data-In-Virtual objects. The algorithm is described in detail in the section “I/O Functions” in the manual “*SAS/C Library Reference, Volume 1.*” You might wish to apply this zap if the default buffer size and/or maximum number of buffers are too large or too small for your site.

The following generic zap must be applied to the transient library (*prefix*.LINKLIB) using APPLYZAP (see *A Guide for the SAS/C Compiler Consultant* for details on using the APPLYZAP program) in order to change the default buffer size and/or maximum number of buffers. Additionally, if you are using the all-resident library, the zap must be applied to *prefix*.ARESOBJ using APPLYZAP. All-resident programs must be relinked. Instead of relinking, you may apply the zap to the L\$C\$LD SO CSECT in the appropriate user load module. Note that you must change the module name in the NAME statement to refer to your user load module when applying the zap to a user program.

If you distribute software linked with the all-resident library, and the customer desires to change the defaults, you will need to distribute this zap to your customers.

```

*   NAME:  Z7505342  PRODUCT:  SASC      CATEGORY:  SPEC      SYSTEM:  MVS
*   DATE:  19JUL02  STATUS:   DZ+UT   USAGE-ID:  LIBRARY-C5342

```

```

*
*   How to change the defaults for Data-In-Virtual amparms
*
*   NOTE:      APPLY TO SASC.LINKLIB (LOAD MODULE FORM TRANSIENT
*              LIBRARY), AND SASC.ARESOBJ (ALL-RESIDENT OBJECT FORM
*              RUNTIME LIBRARY) USING THE APPLYZAP UTILITY.
*
*              THE USER MUST CODE THEIR OWN VALUES FOR THE DEFAULT
*              BUFFER SIZE AND/OR THE DEFAULT MAXIMUM NUMBER OF BUFFERS.
*
*              XXXXXXXX - DEFAULT NUMBER OF 4K PAGES
*              YYYYYYYY - DEFAULT MAXIMUM NUMBER OF BUFFERS
*
*   END
NAME      L$CWPSO  L$C$LDZO   LINKLIB  ARESOBJ
CHECKSUM
VER       0000    0000,0040
VER       0004    0000,0004
REP       0000    XXXX,XXXX
REP       0004    YYYY,YYYY
IDRDATA  Z7505342

```

Note: You do not have to replace data that you are not changing. For example, to make the default buffer size 1024K (256 - 4K pages), use the following:

```

NAME L$CWPSO L$C$LDZO   LINKLIB  ARESOBJ
VER   0000 0000,0040
REP   0000 0000,0100

```

sascc.cfg Configuration File

The `sascc.cfg` configuration file was created in your `dir/bin` directory by the `SASCOEMV` job. The compiler drivers, `sascc370` and `sasCC370`, use the values returned from `sascc.cfg` to determine the installation location of SAS/C USS OS/390 components, as well as the installation location of the OS/390 components. The following is an example of a `sascc.cfg` file:

```

# PATH is the installation location of the USS-specific
# SAS/C components. It contains the 'bin' subdirectory,
# which is where the executables reside.

PATH=/u/userid/7.50/test

# PREFIX is the default prefix used to find PDS load
# modules, library PDSs, and MACLIB PDSs for the
# system include files.

PREFIX=DEPT.C750

# INCLUDE can override the default value of PREFIX for the
# system INCLUDE PDSs. (Note, this also overrides the C++
# include files.)

INCLUDE=DEPT.C750.MACLIBH

# LIBPREFIX overrides the default prefix for locating the
# PDSs to resolve references during linking.

```

```
LIBPREFIX=DEPT.C750
```

```
# CXXPREFIX overrides PREFIX for locating the C++ translator.
```

```
CXXPREFIX=DEPT.C750
```

```
# CXXLIBPREFIX overrides the default LIBPREFIX for finding the  
# C++ include files and library.
```

```
CXXLIBPREFIX=DEPT.C750
```


5

Customizing ISPF Panel Support

ISPF dialogs have been included to enable users to invoke the SAS/C Software from ISPF. The SAS/C Software ISPF dialogs were loaded from the installation medium in “Submitting the Install JCL” on page 5. If you intend to install these dialogs along with the standard ISPF libraries, please read ALL of the instructions in this chapter before continuing.

In order for ISPF to invoke the SAS/C Software products, there are two steps which must be performed.

1. Install the SAS/C ISPF dialogs in conjunction with the standard ISPF libraries.
2. Modify certain ISPF dialogs so that SAS/C can be chosen as an option.

Installing the SAS/C ISPF Dialogs Using Standard ISPF Libraries

To install the SAS/C ISPF dialogs either concatenate the SAS/C ISPF libraries with the standard ISPF libraries, **OR** copy the SAS/C ISPF library members into the corresponding standard ISPF libraries. For instance, copy the members in *prefix . ISPPLIB* into the standard ISPPLIB library. All SAS/C ISPF library members begin with LSC or LSF, so there will be no overlap with any existing IBM members.

The SAS/C ISPF panels were designed to be compatible with the distributed ISPF version 2.3 (or higher). Therefore, the following instructions refer to placing the SAS/C dialogs as item number 15. If item number 15 has been previously used, then replace all occurrences of 15 with the appropriate item numbers. Otherwise, no change is required.

The ISPF installation includes ISPF panels, messages, skeletons, and CLIST libraries. These files must be made accessible to the user in order for the ISPF environment to be complete.

The SAS/C Software products dialogs are provided in both CLIST and REXX EXEC form. You may choose to use either form, depending on your requirements or preferences.

For additional details on how to add a new foreground or batch option refer to the *IBM z/OS VIR5.0 ISPF Planning and Customization* publication, GC34-4814-02, or equivalent publication for the current version of the operating system.

Modifying Standard Dependent ISPF Files

There are several data set members which must be modified in order to allow ISPF to invoke the SAS/C compiler products dialogs. A brief discussion of each modification follows this section. For complete details on how to add a new foreground or batch option refer to the IBM manual listed in “Installing the SAS/C ISPF Dialogs Using Standard ISPF Libraries” on page 21. The statements that follow are for reference only. Your ISPF version may vary slightly.

Customizing Foreground Panels

You may place the SAS/C ISPF dialogs in conjunction with the standard foreground dialogs or in any other location as desired.

Panel ISRFPFA (Foreground Selection)

The menu item for the SAS/C ISPF dialogs should be added to the PANEL definition. For example:

```
15 - SAS/C Compiler Products
```

Now, add the SAS/C compiler products panel to the selection list. For example:

```
&ZSEL = TRANS (TRUNC (&ZCMD, ' .')
.
.
.
15, 'PANEL (LSCF) '
.
.
.
```

Panel ISR40000 (Foreground Help)

The menu item for the SAS/C ISPF dialogs should be added to the PANEL definition. For example:

```
15 - SAS/C Compiler Products
```

Add the SAS/C compiler products panel to the selection list. For example:

```
&ZSEL = TRANS (TRUNC (&ZCMD, ' .')
.
.
.
15, LSC##TUT
.
.
.
```

Other Panels

To add the SAS/C compiler products dialogs to some location other than the standard foreground panel simply modify the necessary panels to invoke the SAS/C panels LSCF and LSC##TUT.

Notes:

- The debugger does NOT require ISPF to run in fullscreen mode. However, it can take advantage of ISPF services when installed in accordance with the previous installation steps and invoked from the SAS/C ISPF dialog.
- The LSC#FPRT panel makes use of panel ISRFPPRT (The standard ISPF print program and panel) to print the listing, if requested. Depending on how ISPF was installed, the LSC#FPRT may need to be changed to reflect another print program/panel.

Customizing Batch Panels

You may place the SAS/C ISPF dialogs in conjunction with the standard batch dialogs or in any other location as desired.

Panels ISRJPA and ISRJPB (Batch Selection)

The menu item for the SAS/C ISPF dialogs should be added to the PANEL definition, for instance:

```
15 - SAS/C Compiler Products
```

Now, add the SAS/C compiler products panel to the selection list. For example:

```
&ZSEL = TRANS (TRUNC (&ZCMD, ' . ' )
.
.
.
15, 'CMD (LSCB) '
.
.
.
```

Panel ISR50000 (Batch Help)

The menu item for the SAS/C ISPF dialogs should be added to the PANEL definition, for example:

```
15 - SAS/C Compiler Products
```

Now, add the SAS/C compiler products panel to the selection list. For example:

```
&ZSEL = TRANS (TRUNC (&ZCMD, ' . ' )
.
.
.
15, LSC##TUT
.
.
.
```

Other Panels

To add the SAS/C compiler products dialogs to some location other than the standard batch panel the following modification will need to be performed.

Modify the necessary panel and add the SAS/C compiler products as an option. For example:

```
C - SAS/C Compiler Products
```

Add the following line in order for the SAS/C compiler products to be invoked:

```
PGM (ISRJB1) PARM (LSCB1) NOCHECK
```

Modifying SAS/C ISPF Dialogs for HyperText Tools

This step should not be performed if the SAS/C FSSL product has not been installed. In order to invoke the HyperText compiler and viewer from ISPF there are two dialog panels which must be modified. The modification is discussed below:

Foreground

Change the lines in LSCF from

```
4A, 'CMD (LSC#NO, LSC#NOFS) '
4B, 'CMD (LSC#NO, LSC#NOFS) '
```

to

```
4A, 'CMD (LSCFHSTR) '
4B, 'CMD (LSCFVSTR) '
```

Batch

Change the lines in LSCB, LSCB1, and LSCB2 from

```
4A,    'CMD(LSC#NO,LSC#NOFS) '
```

to

```
4A,    'CMD(LSCBHSTR) '
```

Modifying SAS/C ISPF Dialogs for C++ Tools

This step should not be performed if the SAS/C C++ product has not been installed. In order to invoke the SAS/C C++ Development System from ISPF there are two dialog panels which must be modified. The modification is discussed below:

Foreground

Change the lines in LSCF from

```
1C,    'CMD(LSC#NO,LSC#NOXX) '
```

to

```
1C,    'CMD(LSCFXSTR) '
```

Batch

Change the lines in LSCB, LSCB1, and LSCB2 from

```
1C,    'CMD(LSC#NO,LSC#NOXX) '
```

to

```
1C,    'CMD(LSCBXSTR) '
```

6

Customizing TSO Support

The SAS/C optional TSO support is made up of the following parts:

- ❑ Command processor support, allowing C programs to be called directly from the TSO command line from libraries other than STEPLIB.
- ❑ TSO commands GETENV and PUTENV, allowing inspection and modification of C environment variables from TSO.

These two features are independent. You may install either one separately or both together.

TSO Command Processor Support

The TSO command processor support feature provides a mechanism for calling C programs as TSO commands from the command line. Sites that choose not to install TSO command processor support must use the TSO CALL command to invoke C programs from the TSO command line (unless you already have other software that provides this capability). Using CALL would require using syntax such as `CALL myclib.load(mycmd) 'x y'`.

The implementation of TSO/E command processor support is different between TSO/E Version 2 and releases prior to TSO/E Version 2. For TSO/E Version 2, support is provided by an EXEC command exit (IKJCT43I). For earlier releases of TSO, support is provided by a C command processor and a front-end to the IBM EXEC processor. The behavior from a user standpoint is identical from release to release, only the implementation differs. For any system, the modules supporting command processing must be placed in the search path for TSO command processors. This includes STEPLIBs in logon procedures, the LPA, and the linklist libraries. We recommend installation into the LPA because these modules are likely to be invoked frequently. Furthermore, they are small and use very little virtual storage.

TSO Releases Earlier than TSO/E Version 2

For systems running versions of TSO earlier than TSO/E Version 2, command processor support is implemented via two modules named C and EXEC. The C module allows users to invoke C programs (found in the library allocated to the DDname CPLIB) from the TSO command line using syntax such as `C mycmd x y`. This allows easier program invocation than would be possible using the TSO CALL command. Note that the C keyword must always precede the program invocation.

The EXEC module is a front end for the IBM EXEC module. It performs the same function as the IBM EXEC module except that it first searches for modules in the library allocated to the DDname CPLIB. The net effect is to allow users to invoke C programs from the command line using syntax such as `mycmd x y`. This is even easier (and more natural to most C programmers) than the syntax supported by the C module, but it does require the replacement of the IBM EXEC module. If you would like further information on these two modules, “Logic for SAS/C EXEC and C Modules” on page 26 contains a description of the logic that is used.

Note: Do NOT install both IKJCT43I and C/EXEC on the same system. IKJCT43I and C/EXEC are incompatible. Always choose the correct one based on your TSO level.

To install the C module on your system, copy the module EXECP2 from the *prefix*.TSOLOAD data set to the target system data set, and rename it to C. To install the EXEC module on your system, copy the module EXECP1 from the *prefix*.TSOLOAD data set to the target system data set,

and rename it to EXEC. You must install C if you install EXEC, and you must install EXEC into a library earlier in the search order than the IBM EXEC command. Be sure to choose a different data set than the one containing the IBM EXEC processor, to avoid overwriting it. The IBM module is most likely to be in SYS1.CMDLIB.

We recommend placing the SAS/C EXEC module in the LPA. If the IBM EXEC happens to be in the LPA already, you should make sure that you have a copy of it in a linklist library (probably SYS1.CMDLIB) and then remove the IBM EXEC module from the LPA before installing the SAS/C version. If you do need to move the IBM EXEC module, take care that you also move the aliases (such as EX).

TSO/E Version 2 and Later

If you have a system running TSO/E Version 2 or later, copy the module L\$UEXEX from the *prefix*.TSOLOAD data set to the target system data set, renaming it to IKJCT43I. Source for L\$UEXEX may be found in the *prefix*.SOURCE data set on the SAS/C product installation medium. If your site has its own version of IKJCT43I, you can modify your site's exit to call L\$UEXEX, or modify L\$UEXEX to call your site's exit, as you prefer. See the source to this module for details.

Note that L\$UEXEX supplies all the functionality of the C and EXEC modules for previous releases. In particular, the C prefix can be used to explicitly call a C program, even though C has not been installed as a system command.

Logic for SAS/C EXEC and C Modules

This section describes the logic used by the SAS/C EXEC and C modules that provide TSO command processor support for versions of TSO prior to TSO/E Version 2. For the logic used by the SAS/C IKJCT43I exit for more recent releases, see *prefix*.SOURCE member L\$UEXEX.

Note: Use of the SAS/C EXEC replacement module does not interfere with the operation of the PCF-II command authorization facility.

The SAS/C TSO command processor support contains the following logic:

- EXEC Command Processor
- C Command Processor

EXEC Command Processor Logic

The following list shows the logic used by the EXEC command processor:

1. If EXEC was entered on the command line, control is passed to the IBM EX module (EX is an alias of the IBM EXEC).
2. If there is an IKJSCAN error parsing the command line, control is passed to the IBM EX module.
3. If the command name is C, control is passed to the SAS/C C command processor.
4. If the DDname CPLIB does not exist, control is passed to the IBM EX module.
5. Otherwise, control is passed to the SAS/C C command processor.

C Command Processor

If the called load module does not exist in the library defined by DDname CPLIB, control is passed to the IBM EX module, when C is called from the SAS/C EXEC.

Installing SAS/C Environment Variable Commands

If you wish to install the TSO commands for C environment variable access, copy the members ENVUTIL, GETENV and PUTENV from *prefix*.TSOLOAD to an appropriate system library (i.e., one contained in the normal TSO search path). You should also update any lists of installed TSO commands, such as ISPTCM in ISPF, to enable the use of GETENV and PUTENV. Note that if you do not choose to install GETENV and PUTENV into system libraries, they can still be executed via the TSO CALL command, but this is far less convenient than direct use from the TSO command line.

7

Customizing CICS Library Support

In “Submitting the Install JCL” on page 5, the CICS transient library was copied into a disk data set. In order to run C programs in a CICS environment you must make the CICS transient library available to CICS, update the CICS PPT, PCT, and DCT tables, and modify the CICS start-up JCL.

Making the CICS Transient Library Available to CICS

There are two ways to make the transient library available to CICS. One way is to add the CICS transient library data set name to the DFHRPL concatenation of libraries in the CICS start-up JCL. Here is a sample DD statement:

```
//          DD DISP=SHR,DSNAME=prefix.CICSLOAD
```

A second method is to place the CICS transient library routines in the OS/390 Link Pack Area (LPA) and take the necessary steps to build a CICS Application Load Table (ALT). Refer to the appropriate CICS manuals for further details on the steps necessary to implement an ALT.

Defining the CICS Transient Library Programs in the CICS PPT

If your site uses the RDO (Resource Definition On-line) facilities of CICS, you can use the RDO definitions in the CICSCSD member which was unloaded from the installation medium into the *prefix.CNTL* data set in “Submitting the Install JCL” on page 5, as input to the IBM utility program DFHCSDUP to define the required PPT entries in your CSD (CICS System Definition) file. The RDO definitions as they appear in that member are as follows:

```
DELETE GROUP (SASC750)
DEFINE PROGRAM (LSHABTRH) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHALNK) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHARGP) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHCST) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHDBUG) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHDBUGM) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHDCICS) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHDMGR) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHDSPL) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHISPL) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHITD) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHITDB) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHKOPR) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHLDBCS) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHNCOM) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHNDBA) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHOSEQ) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHRCOM) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHSGNL) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHSIO) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHSTG) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHSTGM) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHWIO) LANGUAGE (ASSEMBLER) GROUP (SASC750)
```

```

DEFINE PROGRAM (LSHWNG1) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHWNG2) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHWNG3) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHWNG4) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHWNG5) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHWNG6) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHWNG7) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHWNG8) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHWNG9) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE PROGRAM (LSHXEVV) LANGUAGE (ASSEMBLER) GROUP (SASC750)
DEFINE TRANSACTION (DEBUG) PROGRAM (LSHDEBUG) GROUP (SASC750)
LIST GROUP (SASC750) OBJECTS

```

After the group SASC750 is added to your CSD the group must be installed before CICS can use the definitions. Either use RDO to install the group each time CICS is initialized or add the group name to a list that appears in the GRPLIST operand of the SIT (System Initialization Table). See the appropriate CICS manuals for more information on RDO.

Alternatively, you can use the CICSPT member to update your installation's PPT (processing program table). CICSPT was unloaded from the installation medium into the *prefix*.CNTL data set in "Submitting the Install JCL" on page 5. The PPT entries as they appear in that member are as follows:

```

DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHABTRH
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHALNK
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHARGP
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHCST
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHDEBUG
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHDEBUGM
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHDCICS
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHDMGR
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHDSPL
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHISPL
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHITD
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHITDB
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHKOPR
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHLDBCS
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHNCOM
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHNDBA
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHOSEQ
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHRCOM
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHSGNL
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHSIO
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHSTG
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHSTGM
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHWIO
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHWNG1
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHWNG2
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHWNG3
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHWNG4
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHWNG5
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHWNG6
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHWNG7
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHWNG8
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHXEVV
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHWNG9
DFHPPT TYPE=ENTRY, PGMLANG=ASSEMBLER, PROGRAM=LSHXEVV

```

Defining the SAS/C Debugger Front-end Transaction in the CICS PCT

If you plan to use the SAS/C Debugger on CICS then you must define the front-end transaction either in the PCT (Program Control Table) or via an RDO definition. If your site uses RDO to define transactions then you can use the transaction definition that is included with all the program definitions in the CICSCSD member which was previously unloaded from the installation medium into the *prefix.CNTL* data set. The definition which appears in that file is as follows:

```
DEFINE TRANSACTION (DEBUG) PROGRAM (LSHDEBUG) GROUP (SASC750)
```

Alternatively, if your site uses tables to define transactions then you can use the CICS PCT member of the *prefix.CNTL* data set to update your PCT. The PCT entry in that member appears as follows:

```
DFHPCT TYPE=ENTRY, TRANSID=DEBUG, PROGRAM=LSHDEBUG
```

Defining CICS Extrapartition Queues in the CICS DCT

You can use the CICS DCT member to update your installation's DCT (Destination Control Table). The CICS DCT member was unloaded from the installation medium into the *prefix.CNTL* data set in "Submitting the Install JCL" on page 5. The DCT entries as they appear in that member are as follows:

```
STDOUT  DFHDCTTYPE=SDSCI,          USED BY SAS/C FOR STDOUT      +
        DSCNAME=STDOUT,          +
        BLKSIZE=1370,            +
        RECSIZE=133,             +
        RECFORM=VARBLKA,         +
        TYPEFLE=OUTPUT,         +
        BUFNO=1                  +
STDERR  DFHDCTTYPE=SDSCI,          USED BY SAS/C FOR STDERR      +
        DSCNAME=STDERR,         +
        BLKSIZE=1370,            +
        RECSIZE=133,             +
        RECFORM=VARBLKA,         +
        TYPEFLE=OUTPUT,         +
        BUFNO=1                  +
*
*      RECSIZE IS FOR WIDE REPORTS, PREFIXED WITH TERMID/TRANID,
*      PLUS FOUR BYTES FOR RECORD LENGTH (LLBB)
*
STGRPT  DFHDCTTYPE=SDSCI,          USED BY SAS/C FOR =STORAGE REPORTS +
        DSCNAME=STGRPT,         +
        BLKSIZE=1450,            +
        RECSIZE=145,             133 + 8 + 4 +
        RECFORM=VARBLK,         +
        TYPEFLE=OUTPUT,         +
        BUFNO=1                  +
SASO    DFHDCTTYPE=EXTRA,          USED BY SAS/C FOR STDOUT      +
        DESTID=SASO,            +
        DSCNAME=STDOUT,         +
        RSL=PUBLIC              ALLOW ANY TRANSACTION TO ACCESS
SASE    DFHDCTTYPE=EXTRA,          USED BY SAS/C FOR STDERR      +
        DESTID=SASE,            +
        DSCNAME=STDERR,         +
        RSL=PUBLIC              ALLOW ANY TRANSACTION TO ACCESS
```

```

SASR      DFHDCTTYPE=EXTRA,                                     +
          DESTID=SASR,          USED BY SAS/C FOR =STORAGE REPORTS +
          DSCNAME=STGRPT,      +
          RSL=PUBLIC           ALLOW ANY TRANSACTION TO ACCESS

```

The RECFORM, RECSIZE, and BLKSIZE parameters specified on the DFHDCT TYPE=SDSCI macros contain our recommended values. They may be freely modified at your discretion. You will have to COLD start your newly modified DCT for the changes to take effect.

The symbolic names of the extrapartition destinations (SASE, SASO, and SASR) are the default names used by the library. These names may be changed if that is required. See “Changing Default Names for CICS Modules and TD Queues” on page 33 for more information before making any changes to these names.

Allocating the CICS Extrapartition Transient Data Queues

This installation step consists of allocating data sets for the CICS extrapartition transient data queues. You can use the CICSALOC member which was unloaded from the installation medium into the *prefix*.CNTL file in “Submitting the Install JCL” on page 5 to allocate these files.

The following is a listing of the job CICSALOC extracted from your installation medium:

```

//CICSALOC JOB (ACCOUNT INFORMATION) , 'PROGRAMMER' ,MSGCLASS=A,
//          MSGLEVEL=( 1 , 1 ) , TIME=( 5 , 00 )
// *
// *****
// **
// * DOC:   ALLOCATE THE CICS EXTRAPARTITION TRANSIENT DATA FILES.
// * REFER:
// * DATE:
// * NOTE:  MODIFY THE SPACE REQUIREMENTS AS APPROPRIATE
// *****
// **
// *
//ALLOCATE EXEC PGM=IEFBR14
//STDERR  DD DSN=&TDPREFIX..STDERR,
//          DISP=( NEW ,CATLG) , UNIT=&UNIT,
//          VOL=SER=&DISKVOL,
//          SPACE=( CYL , ( 2 , 2 ) ) ,
//          DCB=( RECFM=VBA ,LRECL=133 ,BLKSIZE=1370)
//STDOUT  DD DSN=&TDPREFIX..STDOUT,
//          DISP=( NEW ,CATLG) , UNIT=&UNIT,
//          VOL=SER=&DISKVOL,
//          SPACE=( CYL , ( 2 , 2 ) ) ,
//          DCB=( RECFM=VBA ,LRECL=133 ,BLKSIZE=1370)
//STGRPT  DD DSN=&TDPREFIX..STGRPT,
//          DISP=( NEW ,CATLG) , UNIT=&UNIT,
//          VOL=SER=&DISKVOL,
//          SPACE=( CYL , ( 2 , 2 ) ) ,
//          DCB=( RECFM=VB ,LRECL=145 ,BLKSIZE=1490)

```

Review space allocations for the data sets and change them as appropriate.

Modifying the CICS Start-up JCL

Once the extrapartition transient data queues have been allocated you must include DD statements that reference them in the CICS start-up JCL. The following are sample DD statements. Be sure to use the data set names that were specified in “Allocating the CICS Extrapartition Transient Data Queues” on page 32.

```
//STDERR      DD DISP=SHR,DSNAME=CICS.SASC.STDERR
//STDOUT      DD DISP=SHR,DSNAME=CICS.SASC.STDOUT
//STGRPT      DD DISP=SHR,DSNAME=CICS.SASC.STGRPT
```

Changing Default Names for CICS Modules and TD Queues

This section describes a zap that is useful for CICS sites that wish to change the default values used by the library for the names of the CICS transient load modules and/or the Extrapartition Transient Data queues. Unless you have a very good reason for doing so, we recommend that you DO NOT change these defaults.

You might wish to apply this zap if the default extrapartition transient data queue names (SASE, SASO, and SASR) are already in use at your site.

Note: If you change the default three character prefix for the transient data queue names and this is not done during product installation, then all relevant user application programs must be relinked. Instead of relinking, you may apply the zap to the L\$C\$PFXH CSECT in the appropriate user load module. Note that you must change the module name in the NAME statement to refer to your user load module when applying the zap to a user program.

Two CICS transient library programs must also be zapped: LSHSTG and LSHSTGM. These programs are invoked by the library to support the =storage run-time option. Similarly, you might wish to apply this zap if the default CICS transient library program names (which, by default, all start with the three character prefix LSH) are already in use at your site.

Note: If you decide to change the three character load module name prefix you must also modify the PPT entries (see “Defining the CICS Transient Library Programs in the CICS PPT” on page 29). In addition, the load modules in *prefix*.CICSLOAD must be renamed as well. It is important that the alias names for these members be preserved, otherwise it will be impossible to apply zaps and other maintenance that you receive from the Institute.

The following generic zap may be applied to the CICS resident libraries (*prefix*.CICSOBJ, *prefix*.CICSLIB, and *prefix*.VSEOBJ) using APPLYZAP in order to change the default transient data queue and/or transient load module names.

```
*   NAME: Z7500158   PRODUCT: SASC       CATEGORY: SPEC   SYSTEM: MVS
*   DATE: 19JUL02   STATUS:  DZ+UT      USAGE-ID: LIBRARY-C0158
*
*   How to change default CICS TD queue and transient library names
*
*   NOTE:           APPLY TO SASC.CICSOBJ (OBJECT MODULE FORM CICS RESIDENT
*                   LIBRARY), SASC.CICSLIB (LOAD MODULE FORM CICS RESIDENT
*                   LIBRARY), AND SASC.VSEOBJ (OBJECT MODULE FORM CICS/VSE
*                   RESIDENT LIBRARY), AND SASC.CICSLOAD (LOAD MODULE FORM
*                   CICS TRANSIENT LIBRARY) USING THE APPLYZAP UTILITY.
*
*                   THE USER MUST CODE THEIR OWN VALUES FOR THE
*                   DEFAULT THREE CHARACTER PREFIX OF THE CICS
*                   EXTRAPARTITION TRANSIENT DATA QUEUE NAMES AND/OR
*                   THE CICS TRANSIENT LIBRARY PROGRAM NAMES.
*
*                   XXXX,XX - DEFAULT EXTRAPARTITION TD QUEUE NAME PREFIX
*                   YYYY,YY - DEFAULT TRANSIENT LIBRARY PROGRAM NAME PREFIX
*
```

```

*
*   END
NAME      L$C$PFXH L$C$PFXH      CICSOBJ CICS LIB CICS VSEO
CHECKSUM
VER       0000      E2C1, E200
VER       0004      D3E2, C800
REP       0000      XXXX, XX
REP       0004      YYYY, YY
ALIAS     L$C$PFX
CHECKSUM
IDRDATA   Z7500158
NAME      L$DSTG   L$C$PFXH      CICSLOAD
VER       0000      E2C1, E200
VER       0004      D3E2, C800
REP       0000      XXXX, XX
REP       0004      YYYY, YY
CHECKSUM
IDRDATA   Z7500158
NAME      L$DSTGM  L$C$PFXH      CICSLOAD
VER       0000      E2C1, E200
VER       0004      D3E2, C800
REP       0000      XXXX, XX
REP       0004      YYYY, YY
CHECKSUM
IDRDATA   Z7500158
NAME      L$CDBUG  L$C$PFXH      CICSLOAD
VER       0000      E2C1, E200
VER       0004      D3E2, C800
REP       0000      XXXX, XX
REP       0004      YYYY, YY
CHECKSUM
IDRDATA   Z7500158

```

Note: You do not have to replace data that you are not changing. For example, to change the default three character prefix of the extrapartition transient data queue names from SAS to USR use the following:

```

NAME L$C$PFXH L$C$PFXH CICSOBJ CICS LIB CICS VSEO
VER  0000 E2C1, E200
REP  0000 E4E2, D9

```

8

Customizing Default Character Representation Tables

Modifying the Default Character Set Representation Tables

Since the C language uses some characters that might not be available on all terminals and printers, the SAS/C compiler, the SAS/C C++ Development System, the Object Module Disassembler utility, and the source level debugger accept alternate representations for the common 'problem' characters. The translation table for the SAS/C compiler and the OMD utility is kept in a table that forms part of the compiler. You can change the default table representations to suit the needs of your site.

For each character, the table contains four possible representations. The SAS/C compiler and the OMD utility accept either of two representations of each character in a C source file. These are known as the 'primary' and 'alternate' representations. The SAS/C compiler and the OMD utility can produce either of two representations of each character in a listing file. The first is known as the 'print' representation and is a single character. The second representation is produced by overstriking two characters. That is, a second character is printed directly over the first.

The default table uses the following representations:

Character	Source File Representations		SYSPRINT File Representations	
	Primary	Alternate	Print	Overstrike
Left brace	'C0'	'8B'	'8B'	'4C' '4F'
Right brace	'D0'	'9B'	'9B'	'6E' '4F'
Left bracket	'AD'	'AD'	'AD'	'4C' '60'
Right bracket	'BD'	'BD'	'BD'	'6E' '60'
Circumflex	'5F'	'71'	'5F'	'5F' '40'
Tilde	'A1'	'A1'	'A1'	'7D' '60'
Backslash	'E0'	'BE'	'BE'	'7E' '61'
Bar	'4F'	'6A'	'4F'	'4F' '40'

The primary and alternate representations do not have to be different. For instance, the primary and alternate representations of the tilde are the same. Also, the second overstrike character can be a blank (X'40'), as for the bar above.

You can change the default representations either by changing the table in source code form or by replacing the table with one of the alternate tables provided. The default table member is L\$DCST. It is provided in source code form in the installation medium data set *prefix*.SOURCE, which is loaded into the library *prefix*.SOURCE.

The following alternate tables are provided as members of the *prefix*.SOURCE library.

- L\$DCSP is an alternate table suitable for use in compiling C source files copied from PC DOS via a PC-3270.
- L\$DCSC is an alternate table that creates the effect of the CENTS compiler option that was available in Release 2.10.
- L\$DCSD is an alternate table supporting the use of IBM EBCDIC code pages 1047 and 37.

The translation table is defined using the CHARSET macro. The macro has eight keyword parameters, one for each character. Each parameter takes five operands, one for each possible representation: primary, alternate, print, overstrike(1), and overstrike(2).

After modifying the source table, perform the following:

1. Assemble the source code with the *prefix*.MACLIBA macro library as SYSLIB.
2. Save the resultant object module produced from the assembly as member L\$DCST.
3. Copy this module into the *prefix*.ARESOBJ library and the *prefix*.CICS.ARESOBJ library.
4. Linkedit the L\$DCST object module just assembled.
5. Copy the resulting load module into the *prefix*.LINKLIB library with the name LSCCST and provide the L\$DCST alias for the module.
6. Copy the same load module into the *prefix*.CICSLOAD library with the name LSHCST and provide the L\$DCST alias for the module.

Note: If you have renamed the transient library routines in the *prefix*.CICSLOAD library, then be sure to rename LSHCST. See “Changing Default Names for CICS Modules and TD Queues” on page 33 for more information on renaming the *prefix*.CICSLOAD library members.

Note: If you have copied a previous version of L\$DCST to the link pack area library, you should copy this new version to replace the old one.

If you or your users want to use one of the alternate tables provided, it should be assembled, linked, and renamed as previously described.

Translating Header Files to an Alternate Character Set

Any C source compiled with the SAS/C compiler must use the source file representations specified in the translation table. Modifying the source file representation of some characters in the tables may require that you edit existing C source that contains the default representation to replace occurrences of the changed character. Otherwise, syntax errors will occur when the source is compiled. This includes any source code that was shipped with the product, such as header files.

For example, if the L\$DCSP table is used instead of the default table, the solid bar is changed to a broken bar. However, some of the header files shipped with the SAS/C product contain the solid bar, and therefore, will not compile unless you edit them first and replace the solid bar with a broken bar.

We have supplied a utility, XLTHDR, to help you translate libraries of C header files. This utility is member XLTHDR in the *prefix*.LOAD library.

Using XLTHDR

The SAS/C utility XLTHDR is a translator for libraries of C header files. The program converts a library of C header files using the customer-modified Compiler translate table and creates a new library of header files. This new library of header files is written to ddname NEW. XLTHDR replaces all occurrences of the primary values of the special characters with what is specified in the customer-modified translate table. To use XLTHDR:

1. Allocate the ddname NEW to refer to your output library of header files.
2. Invoke XLTHDR by entering:

```
XLTHDR datasetname
```

where:

datasetname - is the name of the library to be translated

We have provided a sample job that uses the XLTHDR utility to translate the SAS/C C header file library. This sample job is member XLTHDR of *prefix*.CNTL.

Providing Character Translation for C++ Tracebacks

Tracebacks for C++ programs may include function names which contain unusual EBCDIC characters. If your site uses C++, and has defined a special character translate table due to non-standard definitions of such characters, you may need to modify the runtime library to enable use of this table when C++ tracebacks are produced. This change introduces some overhead into the execution of all C programs, so you should make the change only if you use C++, do not use the standard tables, and are concerned about the readability of special characters in C++ tracebacks.

This change is in the form of a zap applied to the SAS/C resident library data sets (*prefix*.STDOBJ and *prefix*.STDLIB). If you expect to run C++ programs under CICS, you should also apply the zap to the CICS resident library data sets (*prefix*.CICSOBJ, *prefix*.CICSLIB, and *prefix*.VSEOBJ).

```

*   NAME: Z7500492   PRODUCT: SASC       CATEGORY: SPEC   SYSTEM: MVS
*   DATE: 19JUL02   STATUS:  DZ+UT      USAGE-ID: LIBRARY-C0492
*
*   Providing character translation tables for C++ tracebacks
*
*   NOTE:           APPLY TO SASC.STDOBJ (OBJECT MODULE FORM STANDARD RES-
*                   IDENT LIBRARY), SASC.STDLIB (LOAD MODULE FORM STANMDARD
*                   RESIDENT LIBRARY), SASC.CICSOBJ (OBJECT MODULE FORM
*                   CICS RESIDENT LIBRARY), SASC.CICSLIB (LOAD MODULE FORM
*                   CICS RESIDENT LIBRARY), AND SASC.VSEOBJ (OBJECT MODULE
*                   FORM CICS/VSE RESIDENT LIBRARY) USING THE APPLYZAP *
UTILITY. THE FAILING PROGRAM SHOULD BE RELINKED.
*
*   END
CHECKSUM
NAME      L$CMAIN  L$C$CXT  STDOBJ  STDLIB
VER       0000    00
REP       0000    01
IDRDATA   Z7500492
NAME      L$CMAINH  L$C$CXT  CICSOBJ  CICSLIB  CICSVSEO
VER       0000    00
REP       0000    01
IDRDATA   Z7500492

```


9

Customizing TCP/IP Defaults

Specifying the High-Level Prefix for TCP/IP Data Sets

On a UNIX operating system there are several data sets which contain site-dependent configuration information for TCP/IP. OS/390 file systems differ from UNIX file structure, and local security or organization considerations can affect how data sets are named. "Network Administration" of *SAS/C Library Reference Volume 2* describes how the SAS/C Socket library searches for these data sets. The following zap is used to change the default TCP/IP prefix as described in "Network Administration." Before using this zap please read and understand "Network Administration."

```

*   NAME: Z7504151   PRODUCT: SASC       CATEGORY: SPEC       SYSTEM: MVS
*   DATE: 19JUL02   STATUS:  DZ+UT      USAGE-ID: LIBRARY-C4151
*
*   Specifying the high-level prefix for TCP/IP data sets
*
*   NOTE:           APPLY TO SASC.LINKLIB (LOAD MODULE FORM TRANSIENT
*                   LIBRARY), AND SASC.ARESOBJ (ALL-RESIDENT OBJECT FORM
*                   RUNTIME LIBRARY) USING THE APPLYZAP UTILITY.
*
*                   THIS ZAP SHOULD BE USED ONLY AS A LAST RESORT. SEVERAL
*                   OTHER METHODS ARE AVAILABLE FOR FINDING TCP/IP
*                   CONFIGURATION DATA SETS. SEE SAS/C LIBRARY REFERENCE,
*                   VOLUME 2, CHAPTER 17 FOR DETAILS.
*
*                   NOTE: CODE YOUR OWN VALUE FOR THE PREFIX.
*                   UP TO 26 BYTES MAY BE SPECIFIED (REPLACING THE
*                   "XX"'s IN THE ZAP BELOW). ALL UNUSED BYTES MUST
*                   BE CHANGED TO "00". THE LAST BYTE OF THE REP
*                   SECTION MUST REMAIN "00".
*
*                   EXAMPLE: CHANGE PREFIX TO "SYS2.TCPIP2"
*                   REP 0000      E2E8,E2F2
*                   REP 0004      4BE3,C3D7
*                   REP 0008      C9D7,E5F2
*                   REP 000C      0000,0000
*                   REP 0010      0000,0000
*                   REP 0014      0000,0000
*                   REP 0018      0000,00
*
*   END
NAME      L$CNDBA  L$CSKFN$      LINKLIB ARESOBJ
CHECKSUM
VER       0000      E3C3,D7C9
VER       0004      D750,0000
VER       0008      0000,0000
VER       000C      0000,0000
VER       0010      0000,0000
VER       0014      0000,0000
VER       0018      0000,00
*
REP       0000      XXXX,XXXX
REP       0004      XXXX,0000

```

```

REP      0008      0000,0000
REP      000C      0000,0000
REP      0010      0000,0000
REP      0014      0000,0000
REP      0018      0000,00
IDRDATA  Z7504151

```

Configuring the SAS/C Library to use the IBM z/OS Name Resolver

Starting with z/OS V1R2, IBM introduced a new DNS Name Resolver designed to replace all previous IBM Name Resolvers. This new Name Resolver is based on the latest version of the BIND DNS Name Resolver, version 9. Instead of a Name Resolver running in the application's Address Space, the new IBM Name resolver runs in a separate Address Space and is executed by a Started Task. In order to access this new Name Resolver, IBM also introduced two new UNIX System Services (USS) Assembler Callable Service calls for the name resolver functions, `gethostbyname(BPX1GHN)` and `gethostbyaddr(BPX1GHA)`.

By default the SAS/C Library will use the SAS/C Name Resolver when `gethostbyname()` or `gethostbyaddr()` is called. To enable the SAS/C Library to use the new USS calls for these functions and therefore use IBM's new z/OS Name Resolver the following zap needs to be applied:

```

*   NAME: Z7502143   PRODUCT: SASC       CATEGORY: SPEC   SYSTEM: MVS
*   DATE: 30MAY03   STATUS:  DZ+UT      USAGE-ID: LIBRARY-C2143
*
*   7.50C Transient Library zap to enable using IBM z/OS 1.2 Resolver (BIND)
*
*   NOTE:           APPLY TO SASC.LINKLIB (TRANSIENT RUN-TIME LIBRARY IN
*                   LOAD MODULE FORMAT), AND SASC.ARESOBJ (ALL RESIDENT
*                   OBJECT FORM RUNTIME LIBRARY) USING THE APPLYZAP UTILITY.
*   -----
*   END
NAME      L$CNCOE  L$CNCOE=   LINKLIB  ARESOBJ
CHECKSUM
VER       000008   0000,0000
REP       000008   0000,0001
CHECKSUM  00090800
IDRDATA  Z7502143

```

10 Customizing CSL NFS Defaults

Note: Before running any CSL NFS application, some customization will need to occur at your site. SAS Technical Report C-113, *SAS/C Connectivity Support Library* documents various customizations which can be performed on a per-site basis. Please read and understand Technical Report C-113 before modifying CSL NFS defaults with the following zaps.

Before any user can access an NFS file, he must be authorized by a login server running on an NFS Server. There are several methods to obtain this authorization. These methods are documented in Technical Report C-113. If no alternate method is used, the default login host name will be NFSLOGINHOST. If your site wants to change this default, you can tailor and apply the following zap:

```

*   NAME: Z7500934   PRODUCT: SASC       CATEGORY: SPEC   SYSTEM: MVS
*   DATE: 19JUL02   STATUS:  DZ+UT      USAGE-ID: CSL-C0934
*
*   Custom zap to change the default NFS login server name
*
*   NOTE:           APPLY TO SASC.CSL.LOADLIB (CSL TRANSIENT LIBRARY)
*                   USING THE APPLYZAP UTILITY.
*
*                   THIS ZAP SHOULD BE USED ONLY AS A LAST RESORT.  SEVERAL
*                   OTHER METHODS ARE AVAILABLE FOR FINDING THE NFS LOGIN
*                   HOST NAME.  SEE C-113 FOR DETAILS.
*                   THE DEFAULT VALUE IS "NFSLOGINHOST" (LOWER CASE).
*
*                   NOTE: CODE YOUR OWN VALUE FOR THE HOSTNAME.
*                   UP TO 256 BYTES MAY BE SPECIFIED.  THE NEW VALUE
*                   MUST BE NULL TERMINATED.
*
*                   EXAMPLE: CHANGE HOSTNAME TO "LOGINSERV.ABC.COM"
*                   REP 0000   D3D6,C7C9
*                   REP 0004   D5E2,C5D9
*                   REP 0008   E54B,C1C2
*                   REP 000C   C34B,C3D6
*                   REP 0010   D400
*
*   END
NAME      L$NNC0A  L$NNLOG$  CSLLOAD
CHECKSUM
VER       0002   9586,A293
VER       0006   9687,8995
VER       000A   8896,A2A3
VER       000E   0000,0000
VER       0012   0000,0000
VER       0016   0000,0000
VER       001A   0000,0000
VER       001E   0000,0000
*
REP       0002   XXXX,XXXX
REP       0006   XXXX,XXXX
REP       000A   XXXX,XXXX
REP       000E   0000,0000
REP       0012   0000,0000
REP       0016   0000,0000

```

```

REP      001A      0000,0000
REP      001E      0000,0000
IDRDATA  Z7500934

```

At session or program startup, a remote file system can be mounted automatically. The remote file system is designated as such in the "fstab" file (/etc/fstab on UNIX systems). We have provided a zap to specify the high level prefix of the "fstab" file on OS/390. The default value of the fstab high level prefix is NFS. If your site wants to change this default, you can tailor and apply the following zap:

```

*   NAME: Z7500935  PRODUCT: SASC      CATEGORY: SPEC      SYSTEM: MVS
*   DATE: 19JUL02  STATUS:  DZ+UT      USAGE-ID: CSL-C0935
*
*   Custom zap to change the NFS data set prefix on MVS
*
*   NOTE:          APPLY TO SASC.CSL.LOADLIB (CSL TRANSIENT LIBRARY)
*                  USING THE APPLYZAP UTILITY.
*
*                  THIS ZAP SHOULD BE USED ONLY AS A LAST RESORT. SEVERAL
*                  OTHER METHODS ARE AVAILABLE FOR FINDING THE NFS ETC.FSTAB
*                  NAME. SEE C-113 FOR DETAILS.
*
*                  THE DEFAULT VALUE IS "NFS" (LOWER CASE) .
*
*                  NOTE: CODE YOUR OWN VALUE FOR THE PREFIX.
*                  UP TO 26 BYTES MAY BE SPECIFIED. THE NEW
*                  VALUE MUST BE NULL TERMINATED.
*
*                  EXAMPLE: CHANGE PREFIX TO "SYS2.NFS"
*                  REP 0000      E2E8,E24B
*                  REP 0004      D5C6,E200
*                  REP 0008      0000,0000
*                  REP 000C      0000,0000
*                  REP 0010      0000,0000
*                  REP 0014      0000,0000
*                  REP 0018      0000,00
*
*   END
NAME     L$NNCOA  L$NNFNM$      CSLLOAD
CHECKSUM
VER      0000      D5C6,E200
VER      0004      0000,0000
VER      0008      0000,0000
VER      000C      0000,0000
VER      0010      0000,0000
VER      0014      0000,0000
VER      0018      0000,00
*
REP      0000      XXXX,XX00
REP      0004      0000,0000
REP      0008      0000,0000
REP      000C      0000,0000
REP      0010      0000,0000
REP      0014      0000,0000
REP      0018      0000,00
CHECKSUM
IDRDATA  Z7500935

```

By default, the SAS/C CSL NFS client library `nfslogin` process will communicate with one of the reserved ports (≤ 1023) or port 2049. Port 2049 has been registered by Sun Micro Systems for use by NFS.

To cause the `nfslogin` process to only communicate with an NFS login server that has registered a specific port for communication, refer to Usage Note 1900.

NFS Security Considerations

This section discusses security risks associated with using the SAS/C CSL NFS client implementation. These risks are being clearly documented in order to help administrators of NFS servers assess the risks involved in granting access to OS/390 users.

Use of the NFS client feature requires that a login server be installed on one of the NFS server systems. The source for two login servers is available in (UNIX format) tar files. Your organization will need to install one or both of these on an NFS server system if it does not already have one running (such as for PC-NFS). More information on login server installation and configuration is contained in Technical Report C-113, Chapter 11, and the login server source distributions. Please note that the login servers must be run with root authority to be recognized as authoritative by the SAS/C CSL NFS client library.

Note that while the methods of attack are documented here in order to reduce their distribution beyond system administrators, programmers who understand mainframe operating systems, TCP/IP, RPC and NFS might comprehend these weaknesses without reading about them. See the discussions of `nfslogin` and NFS security in the SAS/C CSL documentation for a more complete discussion of NFS security.

The risks discussed here are to files on NFS servers. NFS server security problems pose no threat to the mainframe unless files on the server contain mainframe passwords. Furthermore, managers of NFS servers generally have some flexibility in choosing the file that they make available to clients on a particular remote host.

Two methods of attack are practical for knowledgeable mainframe programmers:

- The SAS/C CSL NFS client library has strict requirements for login on an NFS login server. NFS file accesses are based on UNIX user information obtained from this login. A programmer could write an NFS client implementation using an RPC library and use that (lax) implementation instead of the SAS/C implementation. On UNIX this is prevented by the requirement that "root" users bind reserved ports in TCP/IP. Current mainframe TCP/IP implementations do not have restrictions on who can bind a reserved port. Thus this method of attack becomes possible. Note that this problem is present regardless of whether or not the SAS/C CSL product is installed on your system.
- Because the SAS/C CSL product runs in the user's address space, a mainframe programmer might be able to disassemble the code and zap appropriate instructions so that access is allowed without a valid login. We have taken steps to make this harder to do, but it will always be possible as long as the SAS/C CSL libraries run in the user's address space.

A third method of attack is applicable only to sites that are not running a RACF-compatible security system.

- A user could setup a login server on a machine that he controls (a PC for instance), and then direct his logins to that machine. This method of attack is not possible if the SAS/C CSL generalized resource security is enabled, because generalized resource mechanism allows the security administrator to control the IP addresses of machines that can be used as login servers.

Other methods of attack would be relatively difficult compared to these.

11

Installing UNIX System Services (USS) OS/390 SAS/C Components

The job SASCOEMV (member SASCOEMV in *prefix.CNTL*) can be used to install the SAS/C USS OS/390 components into the Hierarchical File System (HFS). The SAS/C USS OS/390 components are provided on your installation media in the file, *prefix.OEMVS.TAR*. SASCOEMV copies *prefix.OEMVS.TAR* into the HFS and installs the components at the location you specified with the symbolic parameter *dir* in the SASCEDTP job.

Submit the SASCOEMV job to install the following SAS/C USS OS/390 components:

- ar370
- binder
- cool
- omd
- pdscall
- sasCC370
- sascc370
- sascdbg
- sheller

When you install the USS SAS/C components you should modify */etc/profile* to define a PATH environment variable which includes the name of the directory in which the components were installed. (This name is *dir/bin*, where *dir* is the name you specified in the SASCEDTP job.)

Note: To allow applications executing under UNIX System Services to access the correct version of the SAS/C transient library, the environment variable *ddn_CTRANS* will need to be enabled for the USS environment. For example, assuming the installation high-level-qualifier is: SAS.C750F, export the following variable:

```
export ddn_CTRANS=SASC.C750F.LINKLIB
```

Refer to the SAS/C Compiler and Library User's Guide, Chapter 8, "Executing C Programs" for additional information.

12 Validating Installation

To validate the installation of the SAS/C products you will need to:

- Perform SAS/C Compiler validation
- Perform SAS/C C++ Development System validation
- Resolve validation notes

SAS/C Compiler Validation

The job NTVALID (member NTVALID of *prefix*.CNTL) validates your installation by invoking the cataloged procedures installed in “Installation JCL for Cataloged Procedures and CLISTS” on page 10. Input to the procedures is a sample C program. If the condition codes for all steps are zero, then the product has been installed successfully.

Check the JCL, and run the NTVALID job.

SAS/C C++ Development System Validation

The job CCVALID (member CCVALID of *prefix*.CNTL) validates your installation by invoking the cataloged procedures installed in “Installation JCL for Cataloged Procedures and CLISTS” on page 10. Input to the procedures is a sample C++ program. If the condition codes for all steps are zero, the product has been installed successfully. Check the JCL, and run the CCVALID job.

To further validate the SAS/C C++ product installation of the Standard C++ Library support, the job RWVALID (member RWVALID of *prefix*.CNTL) should be invoked. If the condition codes for all steps are zero, the product has been installed successfully. Check the JCL, and run the RWVALID job.

Validation Notes

If you receive the error message “LSCX607 Temporary file not created, unit not defined: VIO” or a dynamic file allocation error with the job NTVALID, see “Customizing Temporary File Defaults” on page 15 and “Customizing Dynamic Allocation Defaults” on page 16 to change temporary file allocation defaults.

If you believe there is a problem with your SAS/C software, contact Technical Support at SAS Institute. Be sure you have all the necessary information when you call, such as site number, product release number, installation JCL, and error messages.

If validation was successful, installation of your SAS/C software is now complete. For additional information on using the SAS/C product, refer to the detailed documentation provided with the *SAS/C Online Documentation* CD that was delivered with your software.

13 Maintenance of SAS/C Products

To maintain your SAS/C products, you will need to perform:

- SAS/C code maintenance
- Product license maintenance (SETINIT)
- System maintenance

SAS/C Code Maintenance

A utility for applying zaps called APPLYZAP is distributed in the *uprefix.UTIL.LOAD* data set. In addition, a JCL for using APPLYZAP is distributed in the *uprefix.SASC.CNTL* data set. Please refer to *A Guide for the SAS/C Compiler Consultant* for information on using the APPLYZAP utility, the guide is available on the *SAS/C Online Documentation* CD that was delivered with your software.

Product License Maintenance (SETINIT)

Before your SAS/C compiler product license expires, you will be invoiced for the renewal fee. When SAS Institute receives the renewal fee, you will be mailed source statements (SETINIT instructions) that must be used to authorize the SAS/C compiler for the renewal period. The sample job RENEW (member NTRENEW of *prefix . CNTL*) can be edited with the renewal information and then run to renew your SAS/C compiler authorization. You can refer to NTINITP (member NTINITP of *prefix . CNTL*) to view the SETINIT instructions that were applied to your installation medium.

If your site licenses the SAS/C C++ Development System, you will also be invoiced for its renewal fee. When SAS Institute receives the renewal fee, you will be mailed source statements (SETINIT instructions) that must be used to authorize the SAS/C C++ Development System for the renewal period. The sample job RENEW (member CCRENEW of *prefix . CNTL*) can be edited with the renewal information and then run to renew the SAS/C C++ Development System authorization. You can refer to CCINITP (member CCINITP of *prefix . CNTL*) to view the SETINIT instructions that were applied to your installation medium.

System Maintenance

To determine if there are any system modifications or maintenance required for compatibility with SAS/C, refer to SAS Technical Support services and search *SAS Notes*, using the "SAS/C Notes" database. SAS Technical Support services are available online at: <http://support.sas.com> or by telephone support at: (919) 677-8008. Please have your SAS/C site number available when contacting SAS Technical Support services. Your site number is found in the documentation received with your SAS/C software package or in SAS/C and SAS/C++ compiler listings.

Appendix A Dumping the Limited Distribution Library to Tape

The job DUMPRLDB (member DUMPRLDB of *prefix*.CNTL) can be used to dump the Limited Distribution Library (freely redistributable data sets) for the SAS/C Software products to a tape. The DUMPRLDB job needs some tailoring. You must specify a valid volser and verify the unit value for both proc definitions in the job.

The DUMPRLDB job will dump the following data sets to tape:

DSN on DASD:	DSN on TAPE:	Description
<i>prefix</i> .LINKLIB	SASC.LINKLIB	Transient library. Contains modules loaded as needed at runtime.
<i>prefix</i> .LINKLIB.OBJ	SASC.LINKLIB.OBJ	Transient library in object module format.
<i>prefix</i> .TSOLOAD	SASC.TSOLOAD	TSO command processor support and TSO environment variable support library.
<i>prefix</i> .CICSLOAD	SASC.CICSLOAD	CICS Transient load library.
<i>prefix</i> .DBGHELP	SASC.DBGHELP	Debugger help library.
<i>prefix</i> .CLIST or <i>clstdsn</i> (<i>L\$DBHELP</i>)	SASC.CLIST	Debugger help file clist.
<i>prefix</i> .CNTL(<i>LPSXDBG</i>)	SASC.CNTL	JCL to move the remote debugger into the UNIX System Services Shell.

Note: *prefix*.CLIST is the default value for *clstdsn* (your CLIST data set name). If you changed the value of *clstdsn* in the SASCEDTP job, the “DSN on DASD” listed above will reflect this change.

Your installation package includes the document *Managing the SAS/C Transient Library* which provides the requirements for vendors who plan to redistribute the transient library with their product. This document details the incompatibility issues that occurred when most of the SAS/C transient library was moved above the 16Mb line starting with release 5.50. Additional copies of the document are available by fax, email, or hardcopy by contacting SAS/C Technical Support and requesting a copy of TS496.

Appendix B Dumping the SAS/C CSL Limited Distribution Library to Tape

The job DUMPRCSL (member DUMPRCSL of *prefix*.CNTL) can be used to dump the SAS/C Connectivity Support Library (CSL) Limited Distribution Library (freely redistributable data sets) for the SAS/C Software products to a tape. The DUMPRCSL job needs some tailoring. You must specify a valid volser and verify the unit value for both proc definitions in the job

The DUMPRCSL job will dump the following list of freely redistributable data sets to tape. These data sets are part of the SAS/C CSL Limited Distribution Library. This list of freely redistributable CSL data sets also exist in *prefix*.CNTL(CSLLDLB). For a description of each data set see Appendix C, "List of Installed Data Sets" on page 55.

DSN on DASD:	DSN on TAPE:
<i>prefix</i> .CSL.APP.DEFAULTS	SASCS.APP.DEFLT
<i>prefix</i> .CSL.BITMAPS	SASCS.BITMAPS
<i>prefix</i> .CSL.LOADLIB	SASCS.LOADLIB
<i>prefix</i> .CSL.PCNFSDV2.TAR	SASCS.PCNFSDV2.TA
<i>prefix</i> .CSL.RD.EXEC	SASCS.RD.EXEC
<i>prefix</i> .CSL.RD.LOAD	SASCS.RD.LOAD
<i>prefix</i> .CSL.RD.PROCLIB	SASCS.RD.PROCLIB
<i>prefix</i> .CSL.SASCUIDD.TAR	SASCS.SASCUIDD.TA
<i>prefix</i> .CSL.XCMS.TXT	SASCS.XCMS.TXT
<i>prefix</i> .CSL.XERROR.DB	SASCS.XERROR.DB
<i>prefix</i> .CSL.XKEYSYM.DB	SASCS.XKEYSYM.DB

Appendix C List of Installed Data Sets

SAS/C Compiler Data Sets

The following SAS/C Compiler data sets are installed.

SAS/C Compiler Data Sets	Description
<i>prefix</i> .ARESOBJ	Transient library in object code form.
<i>prefix</i> .BASELIB	SAS/C compiler resident library in load module format, including those modules which may be used in any of the four environments - STD, GOS, SPE and CICS.
<i>prefix</i> .BASEOBJ	SAS/C compiler resident library in object form, including those modules which may be used in any of the four environments - the standard C environment(STD), the Generalized Operating System Environment(GOS), the IBM Customer Information Control System environment(CICS), and the System Programmer Environment(SPE).
<i>prefix</i> .CICS.ARESOBJ	CICS Transient library in object code form.
<i>prefix</i> .CICS.SPELIB	Load module resident library for the CICS System Programmer Environment (SPE).
<i>prefix</i> .CICS.SPEOBJ	Object module resident library for the CICS System Programmer Environment (SPE).
<i>prefix</i> .CICSLIB	SAS/C compiler CICS resident library in load module format, including those modules which may be used in the CICS environment.
<i>prefix</i> .CICSLOAD	CICS Transient load library.
<i>prefix</i> .CICSOBJ	SAS/C compiler CICS resident library in object form, including those modules which may be used in the CICS environment.
<i>prefix</i> .CLIST	SAS/C compiler CLIST library, including ISPF dialogs.
<i>prefix</i> .CNTL	A partitioned data set containing members used in the installation of SAS/C products. These members are loaded into the installation JCL disk data set called <i>prefix</i> .CNTL in the job CUPDATE. Included within the <i>prefix</i> .CNTL members are the object modules that will be link edited to perform the JCL-tailoring. This JCL-tailoring program uses the parameters specified and modifies the <i>prefix</i> .CNTL members so as to produce JCL which is customized for your site's needs. After tailoring is complete, the members are available for further installation and/or customization of SAS/C products.
<i>prefix</i> .DBGHELP	Debugger help library.

SAS/C Compiler Data Sets	Description
<i>prefix</i> .ERRMSG	Error message texts for the GOS feature.
<i>prefix</i> .EXEC	SAS/C compiler REXX library.
<i>prefix</i> .GOSOBJ	Resident object library for GOS.
<i>prefix</i> .HELP	Help library.
<i>prefix</i> .ILCOBJ	Interlanguage Communication (ILC) feature's object library.
<i>prefix</i> .ILCSUB	ILC feature's load library.
<i>prefix</i> .ISPMLIB	ISPF message library containing the necessary messages for foreground and batch compilations.
<i>prefix</i> .ISPPLIB	ISPF panel library containing SAS/C foreground, batch, debugger, CICS, and help panels.
<i>prefix</i> .ISPSLIB	ISPF batch library containing JCL skeleton for SAS/C products.
<i>prefix</i> .LINKLIB	Transient library. Contains modules loaded as needed at runtime.
<i>prefix</i> .LINKLIB.OBJ	Transient library in object module format.
<i>prefix</i> .LOAD	SAS/C compiler and utilities load library.
<i>prefix</i> .MACLIBA	Assembler macros.
<i>prefix</i> .MACLIBC	C macros (header files).
<i>prefix</i> .OEMVS.TAR	SAS/C Open Edition OS/390 components.
<i>prefix</i> .PROCLIB	SAS/C Products catalogued procedure library.
<i>prefix</i> .SAMPLE.C	C samples.
<i>prefix</i> .SAMPLE.CXX	C++ samples.
<i>prefix</i> .SAMPLE.H	Header file samples.
<i>prefix</i> .SAMPLE.ILC	ILC samples.
<i>prefix</i> .SAMPLE.ASM	Assembler samples.
<i>prefix</i> .SAMPLE.AUX	JCL and other miscellaneous samples.
<i>prefix</i> .SOURCE	User-modifiable source code for the GOS and SPE features and the character set representation tables.
<i>prefix</i> .SPELIB	Load module resident library for SPE.
<i>prefix</i> .SPEOBJ	Object module resident library for SPE.
<i>prefix</i> .STDLIB	SAS/C compiler resident library in load module format, including those modules which may be used in the STD environment.

SAS/C Compiler Data Sets	Description
<i>prefix</i> .STDOBJ	SAS/C compiler resident library in object form, including those modules which may be used in the STD environment.
<i>prefix</i> .TSOLOAD	TSO command processor support and TSO environment variable support library.
<i>prefix</i> .VSEOBJ	SAS/C compiler CICS/VSE resident library in object form, including those modules which may be used in the CICS/VSE environment.

SAS/C Resident Library Data Sets

The following SAS/C Resident Library data sets are installed.

SAS/C Resident Library Data Set	Description
<i>prefix</i> .ARESOBJ	Transient library in object code form.
<i>prefix</i> .BASELIB	SAS/C compiler resident library in load module format, including those modules which may be used in any of the four environments - STD, GOS, SPE and CICS.
<i>prefix</i> .BASEOBJ	SAS/C compiler resident library in object form, including those modules which may be used in any of the four environments - the standard C environment(STD), the Generalized Operating System Environment(GOS), the IBM Customer Information Control System environment(CICS), and the System Programmer Environment(SPE).
<i>prefix</i> .CICS.ARESOBJ	CICS Transient library in object code form.
<i>prefix</i> .CICS.SPELIB	Load module resident library for the CICS System Programmer Environment (SPE).
<i>prefix</i> .CICS.SPEOBJ	Object module resident library for the CICS System Programmer Environment (SPE).
<i>prefix</i> .CICSLIB	SAS/C compiler CICS resident library in load module format, including those modules which may be used in the CICS environment.
<i>prefix</i> .CICSLOAD	CICS Transient load library.
<i>prefix</i> .CICSOBJ	SAS/C compiler CICS resident library in object form, including those modules which may be used in the CICS environment.
<i>prefix</i> .CLIST	SAS/C compiler CLIST library, including ISPF dialogs.

SAS/C Resident Library Data Set	Description
<i>prefix</i> .CNTL	A partitioned data set containing members used in the installation of SAS/C products. These members are loaded into the installation JCL disk data set called <i>prefix</i> .CNTL in the job CUPDATE. Included within the <i>prefix</i> .CNTL members are the object modules that will be link edited to perform the JCL-tailoring. This JCL-tailoring program uses the parameters specified and modifies the <i>prefix</i> .CNTL members so as to produce JCL which is customized for your site's needs. After tailoring is complete, the members are available for further installation and/or customization of SAS/C products.
<i>prefix</i> .EXEC	SAS/C compiler REXX library.
<i>prefix</i> .DBGHELP	Debugger help library.
<i>prefix</i> .GOSOBJ	Resident object library for GOS.
<i>prefix</i> .HELP	Help library.
<i>prefix</i> .ILCOBJ	Interlanguage Communication (ILC) feature's object library.
<i>prefix</i> .ILCSUB	ILC feature's load library.
<i>prefix</i> .LIBCXX.A	C++ resident library in AR370 format.
<i>prefix</i> .LIBSTD.A	Rogue Wave Standard C++ Library.
<i>prefix</i> .LIBTOOLS.A	Rogue Wave Standard C++ Library.
<i>prefix</i> .LINKLIB	Transient library. Contains modules loaded as needed at runtime.
<i>prefix</i> .LINKLIB.OBJ	Transient library in object format.
<i>prefix</i> .LOAD	SAS/C compiler and utilities load library.
<i>prefix</i> .PROCLIB	SAS/C Products catalogued procedure library.
<i>prefix</i> .SPELIB	Load module resident library for SPE.
<i>prefix</i> .SPEOBJ	Object module resident library for SPE.
<i>prefix</i> .STDLIB	SAS/C compiler resident library in load module format, including those modules which may be used in the STD environment.
<i>prefix</i> .STDOBJ	SAS/C compiler resident library in object form, including those modules which may be used in the STD environment.

SAS/C Resident Library Data Set	Description
<i>prefix</i> . TSOLOAD	TSO command processor support and TSO environment variable support library.
<i>prefix</i> . VSEOBJ	SAS/C compiler CICS/VSE resident library in object form, including those modules which may be used in the CICS/VSE environment.

SAS/C FSSL Data Sets

The following SAS/C FSSL data sets are installed.

SAS/C FSSL Data Sets	Description
<i>prefix</i> .CNTL	A partitioned data set containing members used in the installation of SAS/C products. These members are loaded into the installation JCL disk data set called <i>prefix</i> .CNTL in the job CUPDATE. Included within the <i>prefix</i> .CNTL members are the object modules that will be link edited to perform the JCL-tailoring. This JCL-tailoring program uses the parameters specified and modifies the <i>prefix</i> .CNTL members so as to produce JCL which is customized for your site's needs. After tailoring is complete, the members are available for further installation and/or customization of SAS/C products.
<i>prefix</i> .FSSL.MACLIBA	FSSL assembler DSECT mapping macro for datastream exits.
<i>prefix</i> .FSSL.MACLIBC	FSSL header files.
<i>prefix</i> .FSSL.SASCINTR.HYP	Compiled hypertext file for SAS/C Online Introduction.
<i>prefix</i> .FSSL.VIEWHLP.HYP	Help file for hypertext viewer.
<i>prefix</i> .FSSL.WDHYPOUT.HYP	Compiled hypertext file for WDSAMPLE program.
<i>prefix</i> .LOAD	FSSL Utilities for the hypertext compiler and viewer.
<i>prefix</i> .STDLIB	Full screen resident library in load module format is copied into the SAS/C compiler resident library in load module format installed earlier with resident library or compiler data sets.
<i>prefix</i> .STDOBJ	Full screen resident library in object format is copied into the SAS/C compiler resident library in object form installed earlier with resident library or compiler data sets.

SAS/C C++ Data Sets

The following SAS/C C++ data sets are installed.

SAS/C C++ Data Sets	Description
<i>prefix</i> .LIBCXX.A	C++ resident library in AR370 format.
<i>prefix</i> .CNTL	Same as SAS/C Compiler data set.
<i>prefix</i> .HELP	C++ translator help library.
<i>prefix</i> .LOAD	C++ translator utilities and load.
<i>prefix</i> .MACLIBC	C++ header files.
<i>prefix</i> .RW.LIBSTD.A	Rogue Wave Standard C++ Library.
<i>prefix</i> .RW.LIBTOOL.A	Rogue Wave Tools.h++ Library.
<i>prefix</i> .RW.MACLIBC	Rogue Wave Standard C++ and Tools.h++ Includes.
<i>prefix</i> .RW.SAMPLE.CXX	Rogue Wave Standard C++ and Tools.h++ Samples.

SAS/C CSL Data Sets

The following SAS/C CSL data sets are installed.

SAS/C CSL Data Sets	Description
<i>prefix</i> .CSL.APP.DEFAULTS	Application defaults for X11.
<i>prefix</i> .CSL.BITMAPS	Bitmaps for X11.
<i>prefix</i> .CSL.CLIST	SAS/C CSL CLIST library.
<i>prefix</i> .CSL.CSL.HYP	Hypertext help for NFS and other CSL components.
<i>prefix</i> .CSL.LIBCSSL.A	Resident library for NFS, SNMP-DPI and rexec() in AR370 archive format.
<i>prefix</i> .CSL.LIBOLDX.A	X10 compatibility library in AR370 archive format.
<i>prefix</i> .CSL.LIBRPC.A	Resident library for RPC and XDR in AR370 archive format.
<i>prefix</i> .CSL.LIBXAU.A	Resident library for X authorization in AR370 archive format.
<i>prefix</i> .CSL.LIBXAW.A	Resident library for Athena Widgets in AR370 archive format.
<i>prefix</i> .CSL.LIBXEXT.A	Resident library for X extensions in AR370 archive format.
<i>prefix</i> .CSL.LIBXI.A	Resident library for input extensions in AR370 archive format.
<i>prefix</i> .CSL.LIBXM.A	Resident library for Motif in AR370 archive format.
<i>prefix</i> .CSL.LIBXMU.A	Resident library for X miscellaneous utilities in AR370 archive format.
<i>prefix</i> .CSL.LIBXT.A	Resident library for the X Intrinsic Toolkit in AR370 archive format.
<i>prefix</i> .CSL.LIBX11.A	Resident library for Xlib in AR370 archive format.
<i>prefix</i> .CSL.LOAD	Programs and utilities for X, RPC, and NFS which can not be redistributed.
<i>prefix</i> .CSL.LOADLIB	CSL transient library.
<i>prefix</i> .CSL.LINKLIB.OBJ	CSL transient library in object format.
<i>prefix</i> .CSL.MACLIBC	Header files for all CSL components.
<i>prefix</i> .CSL.PCNFSDV2.TAR	A source distribution for the SUN PCNFSD Version 2 login server in UNIX TAR format.
<i>prefix</i> .CSL.PROCLIB	SAS/C CSL catalogued procedures library.
<i>prefix</i> .CSL.RD.EXEC	CSL redistributable REXX exec library.

SAS/C CSL Data Sets	Description
<i>prefix</i> .CSL.RD.LOAD	Programs and utilities for X, RPC, and NFS which can be redistributed. These include X clients and programs for logging in and mounting file systems via NFS.
<i>prefix</i> .CSL.RD.PROCLIB	CSL catalogued procedure library.
<i>prefix</i> .CSL.RPC.HYP	Hypertext help file for RPC.
<i>prefix</i> .CSL.SAMPLE.C	C source code sample program library.
<i>prefix</i> .CSL.SAMPLE.CLIST	Sample program CLIST's.
<i>prefix</i> .CSL.SAMPLE.JCL	Sample program JCL.
<i>prefix</i> .CSL.SAMPLE.H	Sample program include files.
<i>prefix</i> .CSL.SAMPLE.LOAD	C sample program executables.
<i>prefix</i> .CSL.SAMPLE.X	Sample program rpcgen source files.
<i>prefix</i> .CSL.SASCUIDD.TAR	A source distribution for the SASCUIDD login server in UNIX TAR format.
<i>prefix</i> .CSL.XCMS.TXT	Default X color management system definitions.
<i>prefix</i> .CSL.XERROR.DB	Default X error database.
<i>prefix</i> .CSL.XKEYSYM.DB	Default X keysyms database.
<i>prefix</i> .CSL.X11.HYP	Hypertext help for X windows.

SAS/C Usage Notes Data Sets

The following SAS/C usage notes data sets are installed.

Usage Note Data Sets	Description
<i>uprefix</i> .SASC.CNTL	SAS/C maintenance JCL data set.
<i>uprefix</i> .UTIL.LOAD	SAS/C maintenance utilities load library.

Appendix D Reference Publications

With Release 7.50, we are continuing our move to online documentation. You will find these books on the CD-ROM titled *SAS/C OnlineDoc™, Release 7.00*, which is included with the SAS/C software distribution package:

SAS/C® 7.50 Changes and Enhancements

Introducing SAS/C® Software, Release 7.00

SAS/C® Software: Changes and Enhancements, Release 7.00

SAS/C® CICS User's Guide, Release 7.00

SAS/C® Compiler and Library User's Guide, Release 7.00

SAS/C® Cross-Platform Compiler and C++ Development System User's Guide, Release 7.00

SAS/C® C++ Development System User's Guide, Release 7.00

SAS/C® Debugger User's Guide and Reference, Release 7.00

SAS/C® Software Diagnostic Messages, Release 7.00

A Guide for the SAS/C® Compiler Consultant, Release 7.00

SAS/C® Library Reference, Volume 1, Release 7.00

SAS/C® Library Reference, Volume 2, Release 7.00

Standard C++ Libraries, Rogue Wave titles:

Standard C++ Library General User's Guide - OEM Edition

Standard C++ Library Class Reference

Tools.h++ 8.0 Class Reference

Tools.h++ 8.0 User's Guide

For additional information refer to these documents:

SAS/C® Compiler Full-Screen Support Library User's Guide, Second Edition, Release 5.01

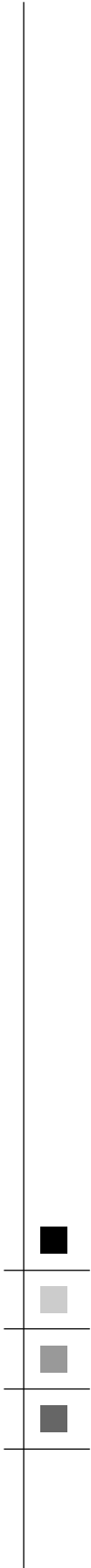
SAS/C® Compiler Interlanguage Communication Feature User's Guide, Release 4.00

SAS/C® Compiler and Library Quick Reference Guide, First Edition, Release 6.00

SAS® Technical Report C-113 SAS/C® Connectivity Support Library, Release 1.00

A Guide for the SAS/C® Compiler Consultant, Release 7.00

SAS® Technical Report C-115 The Generalized Operating System Interface for the SAS/C® Compiler Run-Time System, Release 5.50



www.sas.com/service/techsup/intro.html

SAS is the world leader in e-intelligence software and services. SAS partners with customers to turn raw data, including the vast quantity generated by e-business, into usable knowledge, and makes it available to decision-makers across the enterprise. SAS enables informed business decisions, helping its customers gain competitive advantage in their markets.