

Configuration Guide for SAS[®] 9.2 Foundation for z/OS[®]



Copyright Notice

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *Configuration Guide for SAS® 9.2 Foundation for z/OS®*, Cary, NC: SAS Institute Inc., 2009.

Configuration Guide for SAS® 9.2 Foundation for z/OS®

Copyright © 2009, 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 the copyrighted material in your system and display it on terminals, print only the number of copies required for use by those persons responsible for installing and supporting the SAS programming and licensed 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 — Setting Up SAS 9.2 Foundation	1
Implementing SAS TSO Support	2
Architectural Changes Overview.....	3
Remote Browsing Replaces Remote Help.....	3
Threaded Kernel.....	5
SAS Entry Points	5
Return Codes.....	6
TK Options	7
TK Use of HFS.....	9
TKMVSJNL DDNAME.....	10
Support for Environment Variables on z/OS.....	10
Customizing Default Options and System Configuration Files.....	11
Selecting a Bundled Configuration.....	18
z/OS non-LPA (ENTRY=SASB)	18
z/OS LPA (ENTRY=SASLPA)	18
Installing SAS 9.2 Foundation into the LPA/ELPA	19
System Configuration for Using SAS with TCP/IP.....	21
Planning for TCP/IP.....	21
Overview and Software Requirements	21
Overview.....	21
Software Requirements	22
Configuring SAS to Communicate with TCP/IP	22
1. Verify TCP/IP Stack Definition	22
2. Verify System and USS Process Limits	24
3. Verify TCP/IP Host Name Configuration	25
4. Verify TCP/IP Configuration Files Affecting SAS	27
5. Host Name Resolution for Systems that Run Multiple TCP/IP Stacks	29
6. Customizing UNIX System Services (USS) - Open Edition.....	31
7. SAS Environment Variables	32
8. Diagnosing Configuration Problems.....	36
References	36
Customizing Your SAS CLIST, REXX Exec, and Cataloged Procedure	36
Customizing Your NEWS File	40
Installing the SAS 9.2 SVC Routine.....	41
Installing the SAS SMF Exit	44
Installing UNIX File System Components	46
Configuring SAS Software for Use with the Java Platform.....	51
Locating Hot Fixes.....	55
Chapter 2 — Post-Installation Configuration for National Language Support (NLS)	57
Chinese, Japanese, and Korean DBCS Support	57
DBCS System Option	57
DBCSLANG System Option.....	57
DBCSTYPE System Option	57
Asian Font Catalogs	58
Specifying the Font Catalog in a SAS Session for Traditional Chinese Fonts	58

European Language Support	58
Shipping Map for Countries Based on Encoded Media	59
Using NONLSCOMPATMODE Versus NLSCOMPATMODE	60
New-Line Character and Line-Feed Character	60
Configuring Your System for Locale	61
Changing the Default LOCALE Option Setting.....	61
Running SAS in a Different Locale.....	61
Additional Information	62
Locale Setup on the Remote Server.....	62
Devmaps and Keymaps for SAS/GRAPH Software.....	63
Appendix A — Implementing the SAS/ACCESS Interface to ADABAS	69
Customizing the SAS CLIST, REXX exec, and Cataloged Procedure.....	69
Using the NATURAL Security Interface	70
Using the Samples.....	70
SAS 9.2 Foundation Options for this Interface.....	70
NATURAL Date and Time Support	71
Reentrancy	71
Appendix B — Implementing the SAS/ACCESS Interface to CA-	
DATACOM/DB Software	73
Customizing the SAS CLIST, REXX exec, and Cataloged Procedure.....	73
Using the Samples	74
SAS 9.2 Foundation Options for this Interface.....	74
Appendix C — Implementing the SAS/ACCESS Interface to CA-IDMS	75
Customizing the SAS CLIST, REXX exec, and Cataloged Procedure.....	75
Step 1: Update your SAS CLIST (optional)	75
Step 2: Update your SAS REXX exec (optional).....	76
Step 3: Update your SAS cataloged procedure (optional)	76
SAS/ACCESS DATA Step Interface Enhancement.....	76
Appendix D — Implementing the SAS/ACCESS Interface to DB2	77
Defining the Interface to DB2 and DB2 Users	77
Customizing the SAS CLIST, REXX exec, and Cataloged Procedure.....	78
Creating and Loading the Sample Tables	80
SAS 9.2 Foundation Options for this Interface.....	80
Special Consideration for Using the RRS Attachment Facility.	81
Appendix E — Implementing the SAS/ACCESS Interface to IMS-DL/I	83
Customizing the SAS CLIST and Cataloged Procedure	83
Verifying Installation of the SAS/ACCESS Interface to IMS.....	85
SAS 9.2 Foundation Options for this Interface.....	85
Appendix F — Implementing the SAS/ACCESS Interface to Oracle.....	87
Customizing the SAS CLIST, REXX Exec, and Cataloged Procedure	
(required)	87
Creating and Loading the Sample Tables (optional)	89
Appendix G — Implementing the SAS/ACCESS Interface to R/3.....	91

Appendix H — Implementing the SAS/ACCESS Interface to SYSTEM 2000.....	93
Customizing the SAS CLIST, REXX Exec, and Cataloged Procedure.....	93
Appendix I — Implementing the SAS/ACCESS Interface to Teradata.....	95
Defining the Interface to Teradata	95
Customizing the SAS CLIST, REXX exec, and Cataloged Procedure.....	95
Creating and Loading the Sample Tables (optional)	96
Configuration for FastExporting (optional).....	97
Configuration for MultiLoading (optional)	97
Appendix J — Post-Installation Setup for SAS/ASSIST Software	99
Adding a Master Profile	99
Installing Sample DB2 Tables and a Sample Query Manager	101
Appendix K — Installing the BMDP Interface.....	103
Introduction	103
Installation of the BMDP Interface	103
Appendix L — Post-Installation Setup for SAS/CONNECT Software	105
Configuring SAS/CONNECT.....	105
1. Communication Access Methods Supported for SAS/CONNECT on z/OS.	105
A. System Configuration for the TCP Access Method	105
B. System Configuration XMS (Cross Memory Services) Access Method	106
2. Storing and Locating SAS/CONNECT Script Files	106
3. Types of connections available with SAS/CONNECT on z/OS.....	106
SAS/CONNECT Basic Telnet session	106
SAS/CONNECT Spawner for z/OS.....	107
Benefits to Using the Spawner:.....	107
SAS/CONNECT Spawner Security Configuration.....	107
Security Requirements for Client/User	107
RACF Security	108
Setting up the SAS/CONNECT Spawner Procedure	109
Defining the SAS Startup Shell Script	112
Additional TKM/SENV Configuration Note.....	113
Starting/Stopping the SAS/CONNECT Spawner for z/OS Program.....	114
SAS/CONNECT to the Same Multi-Process Machine on z/OS.....	115
System Configuration for SAS/CONNECT to Same Multi-Processor Machine	115
Considerations for Using XMS with SAS/CONNECT to Same Multi-Processor Machine	116
Appendix M — Implementing SAS/GRAPH Software	117
Understanding the Organization of this Appendix	117
Part 1 — Accessing the SAS/GRAPH Maps Data Sets	117
Part 2 — Customizing Devices	117
Setting up a SAS/GRAPH Translate Table	117
Using SAS/GRAPH Software with ASCII Terminals and ASCII Terminal Emulators	117
Using SAS/GRAPH Software with ASCII Printers.....	118
Installing the Linkable Driver	118
Using SAS/GRAPH Software with IBM 3270-Type Terminals and 3270 Emulators	119
Using SAS/GRAPH Software with IBM 3287, 3268, and 4224 Printers.....	119
Using SAS/GRAPH Software with GDDM.....	119

Part 3 — Setting Up and Modifying Device Catalogs.....	120
How Device Catalogs Are Used	120
How and When to Modify Catalog Entries	121
Part 4 — Device HELP Screens	123
Part 5 — JAVAIMG – Server-side Java Graphs	123
Appendix N — Post-Installation Configuration for SAS/IntrNet Software	125
Overview	125
Installing and Configuring SAS/IntrNet Software	126
Install Your Web Server Software	126
Install Your SAS Software.....	126
CGI Tools Installation Dialogs on PC and UNIX	126
CGI Tools Installation on z/OS.....	130
Installing CGI Tools and SAS Foundation on Machines with Different Operating Systems	130
Test the Web Server	131
Enabling the SAS/GRAPH Java Applets Documentation.....	131
Test the Application Broker	131
Configure a Socket Service	132
Starting the Socket Service.....	134
Testing the Socket Service.....	134
Configure Additional Services	136
Appendix O — Post-Installation Setup for SAS/SECURE Software	137
SAS/SECURE Client for Windows	137
SAS/SECURE Client for Java	137
Client Components	137
Appendix P — Implementing SAS/SESSION Software	139
Introduction	139
Defining SAS/SESSION to the VTAM System	140
Define the VTAM Applications	140
Define the VTAM Logon Mode	140
Defining SAS/SESSION to APPC/MVS.....	141
Security Considerations.....	142
Defining SAS/SESSION to CICS	143
Activating the Interface	146
SAS/SESSION on APPC/MVS	146
SAS/SESSION on CICS.....	146
Executing SAS 9.2 Foundation.....	147
Appendix Q — Implementing SAS/SHARE Software	149
Special Files for Use with SAS/SHARE Software	149
Customizing the Started Task JCL Procedure for a Server	149
Configuration File for a Server	149
Customizing the SAS/SHARE Autocall Macros	149
Selecting Communications Access Methods to Use	150
System Configuration for the Cross-Memory Access Method	151
Installing the SASVXMS Load Module	151
Defining an Anchor Point	152
System Configuration for TCP/IP	153
Specify SAS 9.2 Foundation Option TCPSEC=_SECURE_ for the Server Execution	153

Client-Side Components	153
SAS/SHARE Data Provider	154
SAS ODBC Driver	154
SAS/SHARE Driver for JDBC.....	154
SAS/SHARE SQL Library for C	154
Special Consideration for the SECPROFILE System Option	154
Appendix R — Implementing SAS/TOOLKIT Software	155
Appendix S — Customizing SAS System Forms	157
Customizing the Printer Selection List.....	157
Appendix T — Licensing the SAS 9.2 Foundation	159
Processing Renewal of the SAS System	159
SETINIT Troubleshooting	160
For More Information	161
Requesting a Temporary License Extension.....	161
Appendix U — Logging Directly on to the SAS System	163
Installing the Direct Logon Procedure	163
Example Logon Procedure	163
Using Direct Logon	164
Logging onto the SAS Display Manager System	164
Logging onto a Windowing Application.....	164
Restrictions	164
Accounting Considerations	165

Chapter 1 — Setting Up SAS 9.2 Foundation

Note: If you are familiar with previous SAS releases, please note that the SAS/C Transient Library is now integrated into the SAS Foundation library and is no longer shipped as a separate library.

This document describes the configuration instructions for SAS 9.2 Foundation, which is made up of server-side Base SAS and a variety of server-side SAS products (the exact products vary by customer). Information about the configuration of mid-tier and client-side products is available from your SAS Software Navigator.

The server-side configuration instructions contained in this document are for the configuration of a generic SAS server. If you wish to configure your server for more specific functions, such as a Workspace Server or Stored Process Server, please refer to “Appendix 2: Installing and Configuring the SAS Servers on z/OS” in the document entitled *SAS 9.2 Intelligence Platform: Installation Guide* at

<http://support.sas.com/documentation/configuration/biig.pdf>.

Part 1 describes how to tailor your SAS 9.2 Foundation installation to suit your particular site configuration. The following topics are discussed in this section:

- Implementing SAS TSO Support (pg. 2) (recommended)
- Architectural Changes Overview (pg. 3) (recommended)
- Customizing Default Options and System Configuration Files (pg. 11) (recommended)
- Selecting a Bundled Configuration (pg. 18) (recommended)
- Installing SAS 9.2 Foundation into the LPA/ELPA (pg. 19) (recommended)
- System Configuration for Using SAS with TCP/IP (pg. 21) (recommended)
- Customizing Your SAS CLIST, REXX Exec, and Cataloged Procedure (pg. 36) (recommended)
- Customizing Your NEWS File (pg. 40) (recommended)
- Installing the SAS 9.2 SVC Routine (pg.41) (recommended)
- Installing the SAS SMF Exit (pg. 44) (optional)
- Installing UNIX File System Components (pg. 46) (**highly recommended**)
- Configuring SAS Software for Use with the Java Platform (pg. 51) (recommended)
- Locating Hot Fixes (pg. 55) (recommended)

Important: Each of the ‘recommended’ tasks should be reviewed by someone at your site who can determine whether or not the suggested steps are appropriate and if changes made would be consistent with your operating procedures and site-dependent options and parameters. At the beginning of many of these recommended tasks is a note indicating that ‘You should complete this task if appropriate for your installation.’ Some of the tasks and actions might require knowledge of z/OS operating system principles.

Implementing SAS TSO Support

Note: This task is required if you are planning to run SAS 9.2 Foundation under TSO.

If you plan to run SAS 9.2 Foundation under TSO, you must install the SASCP TSO command processor as outlined below, even if you previously installed it in an earlier release. If you do not install the SASCP TSO command processor, you can run SAS 9.2 Foundation in batch mode only. You must also use either the REXX exec or the CLIST supplied with this installation tape to run SAS 9.2 Foundation under TSO. **Do not try to use a SAS CLIST from a previous release due to new file allocations and other changes.** The REXX exec is new with SAS 9.2 Foundation. It is an alternative to the SAS CLIST for TSO users, and is the default SAS startup command for the SAS/CONNECT Spawner and the SAS/Integration Technologies Object Spawner.

SAS TSO support includes two different facilities. The SAS TSO command processor allows you to invoke SAS 9.2 Foundation from a TSO session. The SAS TSO command support feature provides a SAS statement for executing TSO commands from a SAS session and allows SAS DATA step programs to execute TSO commands conditionally.

SAS 9.2 Foundation supports these features through installation-modifiable modules. These modules contain all of the TSO service routine dependent functions and make no reference to SAS service routines or data areas. These modules are:

- **SASCP** the TSO command processor for invoking SAS 9.2 Foundation.
Note: SASCP is backward compatible and replaces the modules from previous releases. You can continue to execute previous releases of SAS 9.2 Foundation and SASCP with the Release 9.2 SASCP installed on your system.
- **SASTSO** the TSO command executor for executing TSO commands.
- **SASCALL** the TSO CALL command processor used by SASTSO.
- **SASTSMAC** the macros necessary to assemble these modules.

These source modules are unloaded from the tape during installation into the `&prefix.BAMISC` library. The default load modules `SASCP`, `SASTSO`, and `SASCALL` are unloaded into your `&prefix.LIBRARY`. These modules work correctly under all levels of TSO.

Most sites will not need to modify the modules. If you do not need to customize the modules, proceed to STEP 3. However, should your site have special needs, STEPs 1 and 2 describe how to modify and reassemble these modules.

STEP 1: Determine if the distributed TSO support modules require installation customization for your site. Perform customization if needed. (Optional)

The Assembler source code for the `SASCP`, `SASTSO`, and `SASCALL` modules is available in your `&prefix.BAMISC` library for modification. Assembly requires that the `SYSLIB` concatenation contain the general use system macro library `SYS1.MACLIB`, the product-sensitive system macro library `SYS1.AMODGEN` or `SYS1.MODGEN`, and the `BAMISC` library. Examine the source code for the TSO support modules for further details. Sample JCL for assembling and linking these modules is included with the comments of the modules.

If your site has modified the source code and re-linked the `SASCP` module, you can use the TSO `TEST` command to test `SASCP` directly from the SAS library by specifying the `CP` option of `TEST`. Create a test version of the SAS9 CLIST and insert the `TEST` command with the `CP` option immediately before the `SASCP` invocation at the end of the CLIST.

STEP 2: Modify system tables as needed. (Optional)

This installation might also require that you modify certain system tables, such as the PCF and/or ISPF Command Authorization tables. These modifications might require the assistance of local systems or technical support personnel.

The TSO command executor, `SASTSO`, contains support for TSO command validation by both PCF and ACF2. The use of ACF2 command validation must be explicitly enabled either by modifying and reassembling `SASTSO` (following the instructions given in the program header), or by using the z/OS service aid, `AMASPZAP`. Contact the Technical Support division at SAS if you have any questions concerning these procedures.

The functionality that will allow you to interface SAS 9.2 Foundation with IBM's ISPF is included in this release. For this reason, you should examine the ISPF Command Table, `ISPTCM`. For more information about this functionality, refer to Chapter 8, "SAS Interfaces to ISPF and REXX" in *SAS 9.2 Companion for z/OS*.

If the SAS command processor, `SASCP`, is defined within your installation's `ISPTCM`, the flag bit to allow a function pool to be created for the command must be on `x'40'`. If `SASCP` is not in your `ISPTCM`, then either the default flag value must include the preceding bit, or you should do one of the following:

- change the default flag value
- add `SASCP` to the table with the flag on

Note that the bit is `ON` in the default flag value in the sample `ISPTCM` that is distributed by IBM.

STEP 3: Copy the SASCP module to your TSO command load library.

Note: *This step is required if you are running under TSO.*

Installation of SAS TSO support requires that the module `SASCP` be copied from the `&prefix.LIBRARY` to a load library that contains TSO commands. This can be a `STEPLIB` library defined in a `LOGON` procedure, a system link list library, or a link pack area library. (The `SASTSO` and `SASCALL` modules should remain in `&prefix.LIBRARY`.)

Use job `BASASCP` in the `CNTL` data set to copy `SASCP` to your TSO command load library. The `CMDDSN`, `CMDUNIT`, and `CMDVOL` JCL procedure parameters specify the user command load library into which you copy the `SASCP` module. Modify these values to specify your TSO command library. If you have customized the `SASCP` module and are storing it in a library other than `&prefix.LIBRARY`, you also need to specify its new location in the `//SASLIB DD` statement.

Check the JCL and run the `BASASCP` job.

Architectural Changes Overview**Remote Browsing Replaces Remote Help**

In SAS 9.1, remote help was used to display documentation. Documentation was installed on a Java Application server, and the remote browser server was installed on each Windows client.

In SAS 9.2 Foundation, remote help is replaced by remote browsing. An external Java Application server is no longer required to view documentation. Instead, a documentation server is integrated into SAS 9.2 Foundation. The documentation files are contained in the UNIX File System Components. The Remote Browser Server component is still used to display HTML content on a Web browser.

Remote browsing also enables users to view ODS HTML output on their desktop PC. With the UNIX File System Components installed and the HELPLOC system option properly set, ODS HTML files will display in a browser either by clicking on the files in the **Results** window or as they are generated by ODS.

SAS 9.2 Foundation also includes an updated Remote Browser Server for Windows, with an enhanced user interface. It is highly recommended that users upgrade to the new version, which can be downloaded directly from their SAS session. Exit the RBS running on your system and run SAS, then select **SAS Help and Documentation** from the Help menu. The installation URL will be printed to the SAS log.

Additional configuration of your system may be required to use remote browsing. If your TCP/IP networking is configured to use a hosts table instead of DNS (see "Using a Host Table" for details), and one or more SAS desktop clients resides outside the DNS domain of the SAS host, then the hosts table will need to be changed to specify the system's fully qualified domain name (FQDN) as the primary hostname for local network interfaces.

For example, SAS is installed on the host `sashost.mycompany.com`, two client computers exist in the same domain (`client1.mycompany.com` and `client2.mycompany.com`), and two other clients exist in another domain (`client3.domain2.mycompany.com` and `client4.domain2.mycompany.com`). This relationship is illustrated in Figure 1.

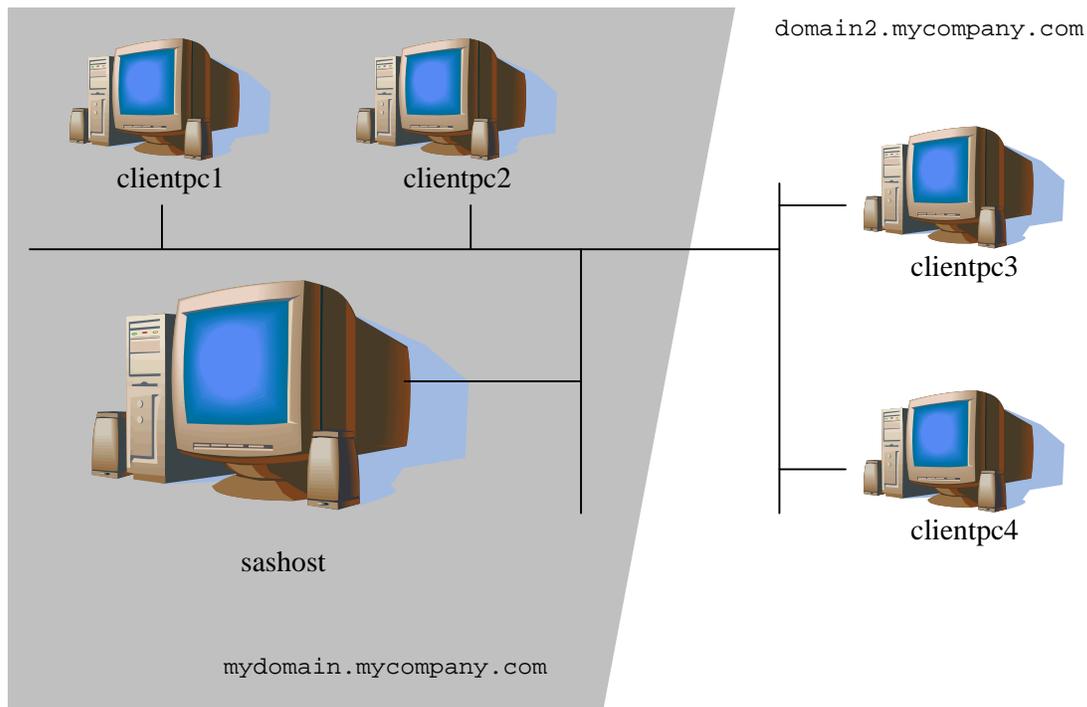


Figure 1. Desktop Clients Inside and Outside the Host's DNS Domain.

If the system `sas.mycompany.com` is not configured with a hostname that is the FQDN for the system, then `client1` and `client2` will be able to view HTML content from SAS, but `client3` and `client4` will not. This is because URLs generated for the SAS host will not include the domain, as in:

```
http://sashost:12345/output.html
```

Since `client1` and `client2` are in the same domain as SAS itself, their browser will build valid hostnames from their domain, `sashost.mycompany.com`.

But `client3` and `client4`, which are outside the domain of the SAS host, will use their domain names to construct a complete hostname, which results in the invalid name `sashost.domain2.mycompany.com`.

By configuring the SAS host with the system's FQDN, URLs for HTML display are valid from all of clients. From the example, the valid URL for all clients is:

```
http://sashost.mycompany.com:12345/output.html
```

Instructions for changing the host table are described in "Using an FQDN with a Host Table" on page 30.

Threaded Kernel

This section introduces a new component known as the threaded kernel (TK).

The threaded kernel (TK) is an independent internal interface to low-level OS interfaces such as memory, events, task creation, etc. The TK interface is booted one time in an address space and its services are available to any task in that address space. SAS 9.2 Foundation makes use of TK services. Moreover, SAS is itself initiated as a TK-created task rather than being entered directly as the job step task or via ATTACH. SAS 9.2 Foundation runs, in effect, as a TK application. The introduction of TK into the SAS environment enables SAS to invoke OS services in a portable manner on multiple platforms, thereby enabling concurrent operations on multiple processors from multiple OS tasks.

Several procs will exploit this new interface in SAS 9.2 Foundation and more will follow in future releases. As a reflection of these architectural changes, there are some changes in the SAS 9.2 Foundation initialization and termination details in the following areas:

- new entry points
- return codes
- TK options
- TK use of HFS
- TKMVSJNL DDNAME

SAS Entry Points

Note: This section describes changes between SAS Version 8 and SAS 9.1/9.2 Foundation. If you are upgrading from SAS 9.1 to SAS 9.2, you will already be familiar with the new entry point names.

Because SAS 9.2 Foundation runs as a TK application, the entry point names for SAS 9.2 Foundation have been changed to emphasize this fact. Since there were three primary entry points in SAS Version 8, there are three entry names in SAS 9.2 Foundation. Each of these new entry point names performs the same basic function.

1. Boot the TK interface in the address space.
2. Use TK task creation services to invoke SAS 9.2 Foundation as a TK application.
3. Wait for the SAS task to complete and respond to the return/abend code.

The three new entry points differ only in the name of the SAS entry point that they use in invoking the SAS application task.

SAS 9.2 Foundation Entry Point Name	SAS Version 8 Entry Point Invoked	Comment
SAS	SASHOST	This is the unbundled entry point loaded from the STEPLIB.
SASB	SASXA1	This is the bundled entry point loaded from the STEPLIB.
SASLPA	SASXAL	This is the bundled entry point typically loaded from the LPA.

It is important to note that the SAS 9.2 Foundation SASHOST/SASXA1/SASXAL entry point names will NOT execute correctly if they are invoked directly, outside of the TK environment created by the SAS 9.2 Foundation entry points. If you have JCL, CLISTs, or programs that invoke SAS directly, they will have to be changed to specify one of the SAS 9.2 Foundation entry point names in order to work correctly in SAS 9.2 Foundation.

Return Codes

The SAS application task created by the TK via the SAS 9.2 Foundation entry points will return the same basic set of documented return codes and abend codes as it did in SAS Version 8. In SAS Version 8, these return/abend codes were returned directly to either the batch initiator task or to the SAS TSO command processor (SASCP). In SAS 9.2 Foundation, the codes are returned instead to the new SAS 9.2 Foundation entry point programs which created them (SAS/SASB/SASLPA). The return/abend codes are processed by the SAS 9.2 Foundation entry point programs as follows:

SAS Version 8 SAS Return Codes	SAS 9.2 Foundation code passed to OS or SASCP
System Abend Codes	USER 998 Abend / Reason Code = SAS Abend Code
User Abend Codes	SAS Abend Code
All Other SAS Return Codes	SAS Return Code (unmodified)

Note that processing of SAS Version 8 return codes in JCL or elsewhere will be unaffected by the SAS 9.2 Foundation return codes changes since normal return codes are passed through unmodified to the OS. Also, any system dumps taken in response to the system or user abend in

the SAS application task will reflect the original abend, not the U998 abend which will not be accompanied by a dump. The U998 abend is intended to indicate that the TK application invoked ended abnormally as specified in the reason code and to ensure that job processing does not continue past the abending step.

Note that since SAS 9.2 Foundation is a dubbed UNIX Systems Services task, CPU excession abends are surfaced as SEC6 abends.

There are also some new TK abend codes for SAS 9.2 Foundation job steps. These codes will be seen only in special case circumstances as described below.

TK USER Abend Code	Meaning
U001 / Reason Code = nn	A task created by TK services abended with code nn AND the TKOPT_NOSTAEX or TKOPT_NOSTAEP TK option was active. These options request TK to abend the job step task, not just the task that suffered the abend (nn). These options will normally be active only as requested by SAS technical support for debugging purposes.
U997 / Reason Code = nn	A task created by TK services has encountered a condition from which it is considered unsafe to continue and still ensure system integrity. The condition is identified by the reason code (nn) which should be reported to SAS technical support. Note that the job containing the job step abending with U997 will report a S20D abend for the job as a whole.

TK Options

Although the TK interface is portable across platforms, there are no portable TK options in SAS 9.2 Foundation. Such portable options might appear in future releases. There is, however, a mechanism in SAS 9.2 Foundation to supply MVS platform-specific options to the TK interface. Note that TK options are independent and separate from SAS application options. They are not specified with the SAS application options on the JCL EXEC card or the SAS CLIST or REXX exec. TK options are provided instead by the standalone program TKMVSENV (see “Support for Environment Variables on z/OS” on page 10).

The TKMVSENV program maintains name/value pairs, similar to UNIX environment variables. These name/value pairs have a lifetime of the job step task. Thus, names defined in a batch job step will be undefined when the batch job step ends. Names defined during a TSO session, on the other hand, will be defined for the life of the TSO session (or until redefined or cleared by another invocation of the TKMVSENV program). The TK boot process makes TKMVSENV name/value pairs available to any application using TK services, including SAS.

The TKMVSENV program will obtain values from one of two sources. If the TKMVSENV program receives standard parms from the JCL EXEC statement or the TSO CALL command line, the program processes the command found in that parms image. If, however, the standard parms length is zero, the program will look for a data set allocated with the DDNAME of TKMVSENV. If the data set is found, the program processes each record in the data set as a command. If the TK boot process finds a TKMVSENV DDNAME allocated, then TK boot invokes the TKMVSENV program with a zero length parms specification. Since a TKMVSENV data set is included in the SAS 9.2 Foundation proc, CLIST, and REXX exec it is, in effect, a TK options file.

Note that the TKMVSENV shipped with SAS 9.2 Foundation is empty. This is because almost all of the supported option names are required only in special debug circumstances, primarily when SAS technical support needs information for problem determination purposes. The data set is included in the SAS proc, CLIST and REXX exec so that it will be available when needed. Also, certain SAS 9.2 Foundation applications (such as the JAVA interface) might make use of the TKMVSENV data set to provide options specific to the JAVA interface. The following is a list of the TKMVSENV name/value pairs supported by the TK interface. Note that a number of them are used as Boolean style options and as such do not require a value after the = sign on the set command, although a value can be provided if desired.

TKMVSENV Command	Meaning
set TKOPT_NOSTAE=	This Boolean option tells the TK interface NOT to provide an MVS ESTAE around TK created tasks. The MVS ESTAE normally provided to protect these tasks enables TK applications to potentially recover from abends. This option is analogous to the SAS NOSTAE option.
set TKOPT_NOSTAEX=	This Boolean option is similar to TKOPT_NOSTAE. It further requests, however, that the entire job step task be ended in the event that a TK created task abends. (See U001 TK abend description above).
set TKOPT_NOSTAEP=	This Boolean option is similar to TKOPT_NOSTAEX. It further requests, however, that any recovery routines registered by the TK application be run. Only if all such recovery routines percolate the abend will the job step be abended.
set TKOPT_DUMPPROL=	This Boolean option provides tasks created by TK with additional debug information in the save areas provided by the function prolog and epilog code, easing the dump reading process. This option is analogous to the SAS DUMPPROL option.
set TKOPT_MEMFILL=	This Boolean option fills memory provided by the TK memory management interface with special characters at memory get and memory free time. This option is intended to enhance the debug process and is analogous to the SAS \$VMMFILL option.
set TKOPT_SVCNO=nnn set TKOPT_SVCR15=nn	These options tell the TK interface how the SAS 9.2 Foundation SVC is installed at the user site. This is necessary because the TK interface might need to use some of the SVC services INDEPENDENT of the SAS application. These options should be specified the same as the SAS options of the same name.

TKMVSENV Command	Meaning
set TKOPT_NOHFS=	<p>This Boolean option is provided for those sites that are unable to provide basic HFS file system resources to the SAS 9.2 Foundation system. See the section below on SAS 9.2 Foundation use of HFS for further details. If this option is specified, then the TK interface will take one of the following actions when an HFS file open is requested by the TK interface:</p> <ul style="list-style-type: none"> • If the HFS file open is an INPUT open request, the file is treated as an empty file. No HFS file opens are performed. • If HFS file open is an OUTPUT open request, a SYSOUT data set is allocated with a DDNAME of TKHFSnnn, where nnn is a unique number which is increased throughout the TK session. The first record in the SYSOUT data set will contain the path name of the HFS file actually requested. The remaining records will contain the data intended for the named HFS file.
set TKOPT_LPANAME=xxxxxxxxx	<p>This option specifies the name of the SAS application entry point invoked by the SAS 9.2 Foundation SASLPA main entry point. If the installation placed the LPA resident module in the LPA with a name other than SASXAL, you will need to specify the same name for the TKOPT_LPANAME option value.</p>
set TKOPT_MEMLEAVE=nnnnnnn	<p>This option specifies the amount of memory in bytes that the TK memory subsystem will attempt to leave free in the address space for use by the system and other users.</p>

TK Use of HFS

One of the components of the TK interface is an IO interface known as TKIO. TKIO is intended to be a 'Simple IO' interface used primarily for tracing and debugging purposes. As such, the TKIO interface is simply not robust enough to handle the complexity of native MVS data set allocations and access methods. Consequently, the TKIO interface on MVS supports only the UNIX style HFS interface.

Given that many MVS installations are not set up to administer HFS file systems in a way that affords access to every potential SAS user, every effort has been made to ensure that no TKIO interface files are required in order to install SAS 9.2 Foundation and run existing SAS applications. Most of the usage is in the form of new LOG= parms on some SAS procedures. These options will expect the HFS path name to which log/debug information is to be written.

To ensure that HFS access will not be required in order to run existing applications with SAS 9.2 Foundation, the following additions have been made to the TKIO interface in the MVS implementation:

- Wherever a TKIO path name is documented to be specified in SAS 9.2 Foundation syntax, you can substitute a name of eight characters or less. If the name specified is allocated to the current SAS session as a DDNAME, then the TKIO data will be read or written to the allocated MVS data set via the QSAM access method.
- If the TKOPT_NOHFS TK option value is active for the session, then TKIO data will be redirected as described in the TK Options section above. This option might be required if

the user does not have the opportunity to specify the path name. This option overrides any DDNAME specifications.

TKMVSJNL DDNAME

TK includes the concepts of a global journal and a global IO. Sometimes, there will be error information placed in the buffer of the global journal and written to the global IO. The default IO will be opened first to the SASCLOG DDNAME, if it is present, and then to the TKMVSJNL DDNAME, if it is present. Otherwise, any information normally directed to the global IO will be lost. The SASCLOG DDNAME is part of the production SAS proc, CLIST and REXX exec so you should look there to find any messages written to the global journal. If you have trouble initializing some non-SAS programs like the V9 object spawner, you might receive error information by allocating a TKMVSJNL DDNAME. However, if the spawner has trouble initializing a spawned SAS process, you should also look to the SASCLOG of the spawned process for initialization error messages. This SASCLOG is most likely a JES SYSOUT file whose owner is the user ID of the spawned process. You can find these files using a JES Spool browser such as SDSF, IOF APPC, or EJES.

Support for Environment Variables on z/OS

TKMVSENV is a stand-alone program that maintains a set of name/value string pairs or environment variables. The name/value pairs only exist during the MVS jobstep task in which the program is invoked.

TKMVSENV takes a standard JCL EXEC style parm list. General register 1 is expected to point to a single parm which is the address of a half word length followed by a parm string of that length. The parm string is composed of a single command. If the half word length field is zero, then the program looks for an allocated DDNAME of TKMVSENV. If the DDNAME is allocated, the file is read and processed one record at a time. Each record is expected to contain a single command. The file must be a sequential data set or a PDS member with a record format of fixed blocked and a logical record length up to 32760. The record length of the sample TKMVSENV file provided with the install is 560, which is the recommended minimum record length.

There are five valid commands. Note that the string values are case sensitive. If you invoke TKMVSENV from a TSO CALL command or from batch JCL and you need to ensure mixed case values are passed, you should specify the ASIS parm on the TSO CALL command image or set CAPS OFF in an ISPF edit session before submitting the batch JCL. Valid commands must begin in column 1 followed by one or more blanks. No blanks are allowed on either side of the '=' sign.

Valid commands are:

Command	Function
set name=value	Set a name/value pair
get name	Get a name value
clear name	Clear a name from the list
reset	Reset the list to zero members
disp	Display all name/values in the list (TSO only)

For the `get` command, R15 returns the address of a half word length of the value, followed by the value string. If the requested name was not found, R15 will contain 0 (zero).

There are two interfaces to the TKMVSENV utility from MVS TK:

1. TK NAMED/REPOSITORY calls with the special prefix of "OSEN." will be mapped to TKMVSENV calls as described in the TK documentation.
2. TKMVSENV will be invoked by MVS TK initialization processing with a zero-length parm. Thus you can allocate a TKMVSENV DDNAME to your TK invocation environment and define name/value pairs which can be queried by the TK application via the OSEN NAMED REPOSITORY interface.

Customizing Default Options and System Configuration Files

Note: You should complete this task if appropriate for your installation.

In the process of installing SAS 9.2 Foundation under z/OS, you establish default SAS system options for all SAS invocations at your site. You can set these global defaults in any of the following places:

- Default Options Table
- System configuration file
- Restricted Options Table

Two sample system configuration files are shipped with SAS 9.2 Foundation. One has option settings set specifically for TSO (member TSOxx in the `CNTL` data set), while the other has settings for batch execution (member BATxx); in both file names, xx is the two-character media and data set code. These files are discussed in more detail later in this section.

Users can also specify options in any of the following locations:

- in a user configuration file, which is specified using the `CONFIG` operand in the CLIST or REXX exec and the `CONFIG` parameter in the cataloged procedure
- on the command line when invoking SAS 9.2 Foundation under TSO
- with the `OPTIONS` parameter in the cataloged procedure, if invoking it under batch
- from within a SAS session, either on an `OPTIONS` statement or from the `OPTIONS` window in the windowing environment

The following indicates the order in which SAS 9.2 Foundation processes options from the various sources:

1. Options in the Default Options Table (if assembled)
2. Options in a system configuration file (if one is used)
3. Options in a user configuration file (if one is specified)
4. Options supplied on the command line in the invocation of the CLIST or REXX exec (TSO) or using the `OPTIONS` parameter in the cataloged procedure (batch)

5. Options in the Restricted Options Table (if assembled)
6. Options specified on an `OPTIONS` statement or in the `OPTIONS` window

Note that later specification of an option overrides an earlier specification. For example, options set in a user configuration file override options set in the system configuration file. Depending upon where you set them, you can streamline system startup or restrict use of options by users.

To streamline system startup for users under TSO, for example, you can add all option defaults needed for TSO at your site to the Default Options Table. Then, since users need not allocate a system configuration file at startup time, you can remove its allocation from the CLIST. Note that in this case, you still need a system configuration file for batch jobs to override the values tailored for TSO in the Default Options Table.

To restrict use of options by users, you can set them in the Restricted Options Table. Since this table is processed last after the Default Options Table, configuration files, and command line options, values set here override all earlier specifications.

STEP 1: Determine default values for SAS system options at your site.

SAS system options that can be used in any operating system environment in which SAS 9.2 Foundation resides are described as *portable* and discussed in *SAS 9.2 Language Reference: Dictionary, Volumes 1 and 2*. z/OS-specific system options are discussed in the *SAS 9.2 Companion for z/OS*. The section “Summary Table of SAS System Options” in the *Companion* contains a table that lists all options available in SAS 9.2 Foundation both portable and z/OS-specific.

The options are listed in alphabetical order for easy reference. This table shows the system default value for each option and includes a reference to the appropriate document for further details.

Note: *z/OS-specific options that warrant special attention at installation time are discussed in the following. These options include the SMF- and SVC-related options, and the superblocking options.*

SMF and SVC Options

The following options should be set in the Restricted Options Table with values that you choose at installation time. The settings of these options are unlikely to require change. Because of the measurements that these options govern, it is prudent to remove these option settings from general user access.

- **SMF|NOSMF** causes an SMF record to be written for every PROC or DATA step containing resource usage statistics for CPU time, memory, and EXCP count. NOSMF disables writing of utilization statistics by SAS 9.2 Foundation. The default is NOSMF.

Note: SAS 9.2 Foundation option `STIMER` must also be on for SMF records to be written.

- **SMFEXIT=name** identifies the user SMF recording exit load module. This load module is loaded and given control before SMF records are written. It allows the user to modify the contents of the record to be written or to disallow the writing of the record (if SMF option is in effect). There is no default.

For information on installing the SMF exit, see “Installing the SAS SMF Exit” on page 43.

- **SMFTYPE=recnum** identifies the SMF record type to record and specifies the default user type for SAS 9.2 Foundation to place in the SMF records it generates when the SMF option is on. The value must be greater than 127 for the SAS 9.2 Foundation SVC to write the SMF record. The default is 128.
- **SVCOR15=value** specifies the value to be placed in Register 15 before invoking the SAS SVC. Only used if **SVC0SVC=109**. The default is 4.

For information about installing the SAS SVC, see “Installing the SAS 9.2 Foundation SVC Routine” on page 41.

- **SVC0SVC=number** specifies the SVC number invoked for functions requiring the SAS SVC. The default is 109.

For information about installing the SAS SVC, see “Installing the SAS 9.2 Foundation SVC Routine” on page 41.

Superblocking Options

To decrease memory fragmentation, SAS 9.2 Foundation has the ability to obtain large blocks of memory from the operating system to satisfy multiple requests for smaller blocks of memory. This scheme, which is called *superblocking*, not only reduces fragmentation but also reduces the number of system `GETMAIN` calls that are issued. This facility is controlled by setting the superblocking options. When they are set to zero, no superblocking is performed.

The superblocking options warrant special attention at installation time because useful values can depend on the mode (batch or TSO) in which SAS 9.2 Foundation runs. The default values for these options are based on early and fairly limited experience in running SAS 9.2 Foundation. In most cases you should not need to override the default values.

To see the current values of these options and other options related to memory, as well as where they were set, submit

```
proc options group=memory value; run;
```

SAS 9.2 Foundation issues superblock overflow warning messages if secondary (OSA) superblock memory is needed. These messages can help you to tune these values for your site.

The following options can be used at SAS invocation or in a configuration file to specify the size of the superblocks. The values can be specified in bytes, kilobytes (K), or megabytes (M).

- **PSUPISA=value** specifies the size of the Initial Size Allocation (ISA) for the portable supervisor.
- **PSUPOSA=value** specifies the size of the Overflow Size Allocation (OSA) for the portable supervisor.
- **VMCTLISA=value** specifies the size of the ISA for SAS 9.2 Foundation memory management control blocks.
- **VMNSISA=value** specifies the initial size allocation of NOSIG pools.
- **VMNSOSA=value** specifies the overflow size allocation of NOSIG pools.
- **VMPAISA=value** specifies the size of the ISA for permanent memory above the 16 Mb line. Permanent memory is memory that is used past procedure or task termination, typically by the host and core supervisor.

- **VMPAOSA=value** specifies the size of the OSA for permanent memory above the 16 Mb line.
- **VMPBISA=value** specifies the size of the ISA for permanent memory below the 16 Mb line.
- **VMPBOSA=value** specifies the size of the OSA for permanent memory below the 16 Mb line.
- **VMTAISA=value** specifies the size of the ISA for temporary memory above the 16Mb line. Temporary memory is that memory which needs to be resident only while the task or procedure is active. This type of memory is highly transient so it is always cleaned up at the end of the task or procedure. Almost all PROC step memory, DATA step memory, and I/O buffers come from this class of memory.
- **VMTAOSA=value** specifies the size of the OSA for temporary memory above the 16Mb line.
- **VMTBISA=value** specifies the size of the ISA for temporary memory below the 16Mb line.
- **VMTBOSA=value** specifies the size of the OSA for temporary memory below the 16Mb line.

WTO Options

There are three options that control the way system operator messages are issued if a SETINIT failure occurs. All of these options are invocation-only so that you can restrict them at your site by including them in the Restricted Options Table. If you work with a systems programmer at your site, you can set values for these options that cause SETINIT error messages to be trapped by operating system automation software.

To see the values of these options, specify the following:

```
proc options group=install; run;
```

Each of the WTO options corresponds to one of the keywords that can be supplied on the WTO system macro.

- **WTOSYSTEMDESC=n** Use this option to specify the message descriptor code. The value of this option is passed to the WTO macro with the DESC keyword.

See IBM documentation for the meaning of the various values (0 to 16) that can be supplied.

- **WTOSYSTEMMCSF=(list-of-keywords)** With this option you supply keywords that control the display of the message. Multiple values are permitted. If you specify more than one value, you must enclose them in parentheses; if you specify only one value, then the parentheses are optional.

Here are the keywords that you can supply:

BRDCST	Broadcast the message to all active consoles.
HRDCPY	Queue the message for hard copy only.
NOTIME	Do not append time to the message.
BUSYEXIT	Do not wait for WTO buffers.

This option corresponds to the MCSFLAG keyword on the WTO macro.

WTOSYSTEMROUT=*n* Use this option to specify the message routing code. The value of this option is passed to the WTO macro with the ROUTCDE keyword.

See IBM documentation for the meaning of the various values (0 to 16) that can be supplied.

The BNDLSUFFIX= Option

specifies a character that is to be appended to every bundle load module name before it is searched for or loaded. The character is appended to the name of every bundle load module (these modules have a prefix of SAB). If the name of the bundle is eight characters long already, the suffix character replaces the last character. The value for the BNDLSUFFIX= option can be enclosed in quotes, but does not have to be. See “Selecting a Bundled Configuration” on page 18 for more information.

The BNDLSUFFIX= option is typically used only by system administrators, and not by the general user.

The SUBSYSID= Option

tells the cross memory services communication facility to use the z/OS subsystem ID that was chosen in the installation process to anchor its resource descriptors. The default value is SAS0. This option is used in conjunction with SAS/SHARE software.

The OPRESTRICTIONS= Option

sets the name of the Restricted Options Table load module, which sets initial options and prevents the user from overriding them. The syntax is OPRESTRICTIONS=AAAAAAAA where AAAAAAAAA is the name of a z/OS load module that must be in LPA or the linklist. See “Step 5: Create a Restricted Options Table” for more information.

The DLINITDEFER Option

suppresses synchronization of VTOC entry at library creation time. If your site uses SMS management classes which specify partial release = yes immediate, or if you utilize a system exit to release space when data sets are closed, you might want to consider specifying DLINITDEFER as a default option for your site.

STEP 2: Determine where to set your option defaults.

Review the procedures for setting default option values in the next three steps. Decide which options should be set in the Default Options Table, the system configuration files, and the Restricted Options Table. Save these lists for use in later steps.

STEP 3: Customize the supplied DFLTOPTS table.

Customizing `DFLTOPTS` (Default Options Table) is optional. If you decide to customize it, edit the `DFLTOPTS` assembler source by adding the options to the table that you would like to include and removing those you do not want. For example, you can put options that have the same value in all execution modes in the Default Options Table.

The source for the `DFLTOPTS` table resides in the `&prefix.BAMISC` library member `DFLTOPTS`. This source represents the `DFLTOPTS` table that is linked into the SAS load modules on your installation tape. Instructions for modifying the `DFLTOPTS` table are included in comments in the source code. The JCL to assemble and link it is in member `BAOPTS1` in the `CNTL` data set. Run the job to assemble the `DFLTOPTS` CSECT and link it into `SASHOST` individually, and into the bundles of which `SASHOST` is a part. Be sure that if you run a bundled configuration, you re-link the bundles that you use.

Since the `DFLTOPTS` table is linked with `SASHOST`, it does not have to be loaded to be read. If you can put all the default options that you need in the `DFLTOPTS` table, you do not have to use a system configuration file.

The `DFLTOPTS` table can contain as many option length/value pairs as needed. An option length/value pair consists of a half word length field, followed by a character string of the form `option`, `NOoption`, or `option=value`. The `OPT` macro in the assembler source calculates the length fields given the character strings. The table must be terminated by a pair with a length field of 0.

STEP 4: Customize the system configuration files.

Set up system configuration files to establish installation-wide default values for commonly-used options. The default SAS CLIST, REXX exec, and cataloged procedure installed from the tape always allocate a system configuration file and allow for specification of a user configuration file using the `CONFIG` operand and parameter. They use sample system configuration files that are created in the `CNTL` data set; these samples are described below.

All SAS 9.2 Foundation installations, whether domestic or foreign, are now encoded images. Encoded images allow SAS 9.2 Foundation to be invoked using different character sets. For example, `W0` is English and `W3` is German, etc. Other examples of the encoding abbreviations can be found in the Languages, Encodings and Installation Codes table from the Installation Instructions.

Create your system configuration files by customizing the following default system configuration files that are supplied with the installation:

- `TSOxx` running under TSO
- `DTSOxx` running under TSO with double-byte support
- `BATxx` running in batch mode
- `DBATxx` running in batch mode with double-byte support

These default system configuration files are installed into the `CNTL` data set as samples for you to review. They contain some of the options for which you might want to establish installation-wide default values, depending on batch or interactive execution mode. However, not all of these options are required.

The `CNTL` data set, where the sample configuration files reside, is a blocked partitioned data set with fixed-length, 80-byte records. You can create a system or user configuration file as any sequential data set or member of a PDS.

The sample configuration files contain option settings separated into logical groups by comments. Records in a configuration file are either comment lines (indicated by an asterisk in column 1) or option lines. In the sample configuration files, options are listed one per line to make them easier to read and maintain. However, this is not required; more than one option can be included on a single line. You can also add a comment anywhere between options by enclosing it between `/*` and `*/`.

For options that require a value, the option must be specified as `option=value` with no blanks before or after the equal sign. Any SAS system option can be specified in the system configuration file.

Once you have entered the options in the system configuration file, no further processing is necessary (unlike options specified in the Default Options Table or Restricted Options Table). However, if you rename or move the configuration files from the installation `CNTL` data set, be sure to update your CLIST, REXX exec, and cataloged procedure accordingly to reference the new data set names.

STEP 5: Create a Restricted Options Table (optional).

Options specified in the Restricted Options Table cannot be overridden by the user. For options in the Restricted Options Table to be processed, the restricted options module must come from a linklist library but does not have to be APF authorized.

The format of the Restricted Options Table is the same as that for the `DFLTOPTS` table, except that the option length/value pairs must be preceded by the header `***SASOPTRS***`. Like the Default Options Table, the Restricted Options Table is an assembler source module that must be assembled and linked.

The source for the sample Restricted Options Table is in the `&prefix.BAMISC` library member `SASOPTRS`. The JCL to assemble and link this table is in member `BAOPTS2` in the `CNTL` data set. The Restricted Options Table is optional. If you decide to install it, edit the `SASOPTRS` member containing the assembler source, adding the options to the table that you would like to include and removing those that you do not want to use. Run the `BAOPTS2` job to assemble the `SASOP910` module and link it into a linklist library.

Be sure that the `SYSLMOD DD` statement in the `BAOPTS2` job points to a linklist library since the `SASOP910` module must come from a linklist library in order to be processed.

Note: *The sample table provided with this installation restricts access to VSAM data sets. You should not run this sample "as is" unless you want to restrict the use of this feature.*

SAS checks for the restricted options module, `SASOP920`, but it is not required. Do not create the module unless you have customized the sample contents to meet your installation requirements.

In SAS 9.2 Foundation, you can apply different restricted options tables in different situations. This might be useful, for example, in a scenario that required multiple releases of SAS 9.2 Foundation to run concurrently. Modify the JCL in the `BAOPTS2` member to assemble another restricted options module. Simply change the name on the `NAME` statement from `SASOP920` to a name of your choice. Then specify the following in the config file:

OPRESTRICTIONS=name_of_your_choice

OPRESTRICTIONS must be specified to override the default restricted options table identifier SASOP920. Should OPRESTRICTIONS be omitted, SAS continues to assume module SASOP920 as the restricted options module table identifier.

STEP 6: Verify your default option settings.

After performing the various steps described in this section to set default option values for your site, you should run `PROC OPTIONS VALUE; RUN;` to verify that the desired defaults are in place. The z/OS-specific SMF and SVC options are not normally displayed by `PROC OPTIONS`. Specify `PROC OPTIONS VALUE GROUP=INSTALL; RUN;` to review these options.

Selecting a Bundled Configuration

Note: You should complete this task if appropriate for your installation.

Note: If the Double Byte Character Set (DBCS) is to be used in conjunction with a bundled configuration, then the modules that are to be renamed **MUST** come from the `&prefix.DBCS.LIBRARY`. This holds true for the non-LPA bundle configuration when using the `BNDLSUFFIX=` system option. The renamed modules are in the `&prefix.DBCS.LIBRARY`.

SAS 9.2 Foundation for z/OS is distributed in two bundled configurations that are tailored for execution in the z/OS environment, and one unbundled configuration. The two bundled configurations differ in that one is tailored for execution with some modules installed in the Link Pack Area (LPA), while the other is tailored for execution with no modules installed in the LPA. Note that the code is identical across the three versions; the only difference is in the packaging.

The `ENTRY` parameter of the JCL cataloged procedure or TSO CLIST or REXX exec determines which configuration is used. The default entry point name is `SAS`, which runs the unbundled configuration. If you want to run a bundled configuration, which is highly recommended, edit your installed SAS CLIST, REXX exec and cataloged procedure to specify the appropriate entry point name for your site as described in the following section. (See “Customizing Your SAS CLIST, REXX Exec, and Cataloged Procedure” on page 36.)

z/OS non-LPA (ENTRY=SASB)

The bundled components of the z/OS non-LPA configuration consist of the following modules located in your installed SAS LIBRARY data set: `SASXA1`, `SABXSPL`, `SABXINI`, `SABXTRM`, `SABDSC`, `SABDSX`, `SABZPLM`, `SABZPLC`, as well as others when other Institute Program Products (IPPs) are installed. `SABXINI` and `SABXTRM` are transient modules used during initialization and termination, respectively. `SABDSC` and `SABDSX` are the DATA step compilation and execution modules, respectively.

`SASB` is the entry point. To execute SAS 9.2 Foundation using this configuration, specify `ENTRY=SASB` in your `SASEDITP` member before you run the installation job(s), or directly in your SAS CLIST, REXX exec and PROC.

z/OS LPA (ENTRY=SASLPA)

The bundled components of the z/OS LPA configuration consist of the modules listed in the following section.

All modules that are eligible to be loaded into the LPA will reside above the 16Mb line. SASLPA is the entry point. To execute SAS 9.2 Foundation using this configuration, you will need to follow the procedures outlined in the next section.

Installing SAS 9.2 Foundation into the LPA/ELPA

Note: You should complete this task if appropriate for your installation.

You can install the bundled modules in the LPA/ELPA. If the bundled modules are not installed in the LPA/ELPA, they are loaded into the address space of each SAS 9.2 Foundation user. This can cause a significant increase in the working set size, placing a heavy burden on the paging subsystem. If you have many users of SAS 9.2 Foundation, this might be an important consideration.

STEP 1: Decide whether to install SAS 9.2 Foundation in the LPA/ELPA.

Contact your systems programming staff to discuss the particular considerations involved at your site. You can install just the basic supervisor bundles (SASXAL, SABXSPL, SABXDML, SABDS), or the supervisor bundles and the other bundles listed below as optional. The module sizes are as follows:

z/OS Configuration:

Bundled Modules for ELPA	Size
SASXAL	2450K
SABXSPL	6295K (recommended)
SABXDML	1038K (recommended; Display Manager)
SABDS	853K (recommended; DATA step)
SABSCLL	2708K (optional; used by SAS/ASSIST and SCL applications)
SABDBGM	294K (optional; SCL debugger)
SABZPLH	78K (optional; printing routines)
SABXGPH	1805K (optional; part of SAS/GRAPH)

Total for ELPA 15521K

STEP 2: Install the modules into the LPA/ELPA using the standard procedure at your site.

STEP 3: Ensure that modules in LPA/ELPA do not have the same names as modules in the installed SAS LIBRARY data set.

Note: STEP 3 is necessary to prevent z/OS from loading LPA/ELPA modules into the user's address space when a JOBLIB or STEPLIB DD statement (batch), or a LOAD or SASLOAD CLIST parameter (TSO) references a SAS 9.2 Foundation LOAD LIBRARY.

You can do this in one of three ways:

- Delete the bundled modules from the installed SAS LIBRARY data set.
- Rename the bundled modules in the installed SAS LIBRARY data set.
- Rename the bundled modules in the LPA/ELPA, leaving them in the installed SAS LIBRARY data set with their original names.

This approach has two advantages. It simplifies the application of maintenance by allowing you to apply maintenance directly to the SAS LIBRARY data set. It also facilitates the concurrent running of different releases of SAS 9.2 Foundation.

To rename the bundles, do the following:

- Choose a single character (0-9, A-B, D-E, G, I-W, Y-Z, #, @, \$) as your suffix character. This will be the value of the `BNDLSUFFIX=` SAS system option. Specify `BNDLSUFFIX=character` in default CONFIG files, default options tables, or restricted options tables.
- Rename the bundles, adding the suffix character to the old name to get the new name. For example, if you choose the character 0 as your suffix character, rename the modules listed as follows:

Original Name	New Name
SASXAL	SASXAL0
SABXSPL	SABXSPL0
SABXDML	SABXDML0
SABDS	SABDS0
SABSCLL	SABSCLL0
SABDBGM	SABDBGM0
SABZPLH	SABZPLH0
SABXGPH	SABXGPH0

Note: Your list of modules might not include all those in this example. Your list will depend on the SAS 9.2 Foundation products that you have licensed.

If you change the name of SASXAL, you must specify the new name in the TKMVSENV member located in the `&prefix.TKMVSENV` data set. For example, using the example above, add the following line to the TKMVSENV member:

```
set TKOPT_LPANAME=SASXAL0
```

Step 4: Make sure the ENTRY parameter of the JCL cataloged procedure and TSO CLIST and REXX exec defaults to the appropriate name, SASLPA.

If you plan to make further CLIST or procedure customizations, edit the SASEDITP member of the CNTL data set to specify your name for the ENTRY parameter.

Step 5: If you have renamed the modules in the LPA/ELPA, use the BNDLSUFFIX= SAS system option to tell SAS which set of bundled modules to use:

BNDLSUFFIX=character



If you use `BNDLSUFFIX=` for any of the bundled modules, you must rename them all, including those bundles which are added later. A mixture of bundles which have been renamed and those that have not will cause the system to fail. This applies to bundles in both the LPA/ELPA and the SAS LIBRARY data set.

System Configuration for Using SAS with TCP/IP

Planning for TCP/IP

Here are the primary configuration issues to consider when preparing your site for TCP/IP to run under the z/OS operating environment.

1. For the IBM IP Communications Server, configure or verify the host name configuration by adding a `HOSTNAME` statement in the appropriate `IBM TCPIP.DATA` file.
For information about TCP/IP stacks and how to determine whether a system uses single or multiple TCP/IP stacks, see “1. Verify TCP/IP Stack Definition” on page 22.
2. For SAS 9.2, you will use the IBM z/OS Resolver for the resolution of domain names.
3. Verify that appropriate services are configured for `SAS/CONNECT` or `SAS/SHARE` in the `SERVICES` file. For details, see “The Services File” on page 35.

Starting with SAS 9.2, Internet Protocol Version 6, IPv6, is supported, which is the successor to the Internet Protocol, IPv4. For information about IPv6, see *z/OS V1R9.0 Communication Server: IP Configuration Guide*, IBM SC31-8775-11.

The steps in this section are required if you are to take full advantage of base SAS Software’s functionality. TCP/IP must be configured if you will be using certain features of base SAS software, such as `EMAIL`, `URL`, `Socket` and `FTP ACCESS` methods. You will not be able to use these features if you have not properly configured SAS for use with TCP/IP. If you choose not to complete the steps in this section, you should notify SAS users that this functionality is not available.

In addition, many SAS solutions and products might require the steps in this section to be completed. A partial list of these products would include `SAS/SHARE`, `SAS/CONNECT`, `SAS/IntrNet Software`, `SAS IT Resource Management`, `SAS Integration Technologies` and `SAS OLAP Server Software`. Please refer to product-specific appendices for further details.

Overview and Software Requirements

Overview

TCP/IP is a set of layered protocols that enable cooperating computers to perform tasks and to share resources across a network. TCP/IP is comprised of TCP and IP.

TCP is a set of routines that applications use to communicate with another computer over a network. All applications do not use TCP. However, all network applications require the services

that are provided in IP. IP is a set of routines that TCP calls, but the IP routines are also available to applications that do not use TCP. SAS uses both TCP and IP, and requires that certain types of information be made available to the operating environment.

Although you might refer to a computer by using its host name, TCP/IP applications refer to computers by using their IP addresses. To facilitate the use of host names in a network, the Domain Name System translates host names to IP addresses. This Domain Name System provides host-to-IP address mapping through network server hosts, which are called domain name servers. The Domain Name System also provides other information about server hosts and networks, such as the TCP/IP services that are available to the server host and the location of the domain name servers in the network.

Software Requirements

Verify that these software requirements have been met:

- The IBM z/OS V1R7 IP Communications Server TCP/IP package has been installed.
Note: SAS supports any vendor's TCP/IP software that is functionally compatible with the IBM z/OS IP Communications Server package.
- The UNIX System Services (USS) file system is available.
- A default OE segment (or an individual OE segment for each user ID) is required and must be defined in the security software (such as RACF).
- The IBM z/OS DNS Resolver must be active.

Configuring SAS to Communicate with TCP/IP

To configure SAS to communicate with TCP/IP or to verify your configuration, the following steps **MUST** be completed, and the results made available to SAS:

1. Verify TCP/IP Stack Definition (below)
2. Verify System and USS Process Limits (on page 24)
3. Verify TCP/IP Host Name Configuration (on page 25)
4. Verify TCP/IP Configuration Files Affecting SAS (on page 27)
5. Host Name Resolution for Systems that Run Multiple TCP/IP Stacks (on page 29)
6. Customizing UNIX System Services (USS) - Open Edition (on page 31)
7. SAS Environment Variables (on page 32)
8. Diagnosing Configuration Problems (on page 36)

1. Verify TCP/IP Stack Definition

TCP/IP Communication Stack Definition

TCP/IP stack is a term for the set of protocols that comprise TCP/IP. A TCP/IP Communication stack that runs under the z/OS operating environments is implemented as a UNIX System

Services (USS) physical file system (PFS). An operating environment can run using one or more TCP/IP stacks. The stack definitions are located in SYS1.PARMLIB(BPXPRMnn).

The IBM INET physical file system type supports a single TCP/IP stack. The IBM CINET physical file system type supports multiple stacks.

Note: If you will configure only one TCP/IP stack and you have access to both INET and CINET, it is advisable to configure the stack under INET because of its efficiency over CINET. For complete details, see the IBM documentation z/OS UNIX System Services Planning.

Sample Definitions of TCP/IP Stacks

USS physical file systems are configured in the IBM parmlib member BPXPRMnn. These examples show typical entries in the BPXPRMnn parmlib member for INET and CINET physical file systems.

- Defining a Single IBM TCP/IP Stack

This example shows the statements in the IBM parmlib member BPXPRMnn that define an INET physical file system for a single IBM TCP/IP stack system.

```
FILESYSTYPE TYPE(INET) ENTRYPOINT(EZBPFINI)
  NETWORK DOMAINNAME(AF_INET)
    DOMAINNUMBER(2)
    MAXSOCKETS(64000)
    TYPE(INET)
```

- Defining a Single IBM TCP/IP Stack that supports IPv4 and IPv6

This example shows the statements in the IBM parmlib member BPXPRMnn that define an INET physical file system for a single IBM TCP/IP stack that enables IPv4 and IPv6. For details, see “About TCP/IP Internet Protocol (IP) Addressing” on page 24.

```
FILESYSTYPE TYPE(INET) ENTRYPOINT(EZBPFINI)
  SUBFILESYSTYPE NAME(TCPIP)
  TYPE(INET)
  ENTRYPOINT(EZBPFINI)
  NETWORK DOMAINNAME(AF_INET)
    DOMAINNUMBER(2)
    MAXSOCKETS(64000)
    TYPE(INET)
  NETWORK DOMAINNAME(AF_INET6)
    DOMAINNUMBER(19)
    TYPE(INET)
```

- Defining Multiple IBM TCP/IP Stacks

This example shows the statements in the IBM parmlib BPXPRMnn member that define a multiple TCP/IP stack system. This example includes two IBM stacks, TCPIP and TCPIP2.

The values of the FILESYSTYPE substatements, TYPE and ENTRYPOINT, define the PFS type and the entry point for the CINET PFS. CINET requires a SUBFILESYSTYPE statement for each stack. The NAME substatement names the TCP/IP stack that is being

defined. This name is also used as the name of the Started Task that invokes the TCP/IP stack.

Note: *CINET requires the NAME substatement.*

An optional SUBFILESYSTYPE substatement named DEFAULT defines the default TCP/IP stack for a multiple stack system. If DEFAULT is not specified or if the default stack is not active, the first stack that is activated is the default stack.

```
FILESYSTYPE TYPE(CINET) ENTRYPOINT(BPXTCINT)
  NETWORK DOMAINNAME(AF_INET)
    DOMAINNUMBER(2)
    MAXSOCKETS(64000)
    TYPE(CINET)
    INADDRANYPORT(63000)
    INADDRANYCOUNT(1000)

SUBFILESYSTYPE NAME(TCPIP)
  TYPE(CINET)
  ENTRYPOINT(EZBPFINI)
  DEFAULT

SUBFILESYSTYPE NAME(TCPIP2)
  TYPE(CINET)
  ENTRYPOINT(EZBPFINI)
```

About TCP/IP Internet Protocol (IP) Addressing

TCP/IP applications refer to networked computers via their fully qualified domain names (FQDN) and their IP addresses. Because IP addresses can change easily, SAS applications that contain hard-coded IP addresses are prone to maintenance problems. To avoid such problems, use of an FQDN is preferred over an IP address. The name-resolution system that is part of the TCP/IP protocol is responsible for locating the IP address that is associated with the FQDN.

SAS 9.2 introduces support for the "next generation" of Internet Protocol, IPv6, which is the successor to the current Internet Protocol, IPv4. Rather than replacing IPv4 with IPv6, SAS 9.2 will support both protocols. There will be a lengthy transition period during which the two protocols will coexist. A primary reason for the new protocol is that the limited supply of 32-bit IPv4 address spaces was being depleted. IPv6 will use a 128-bit address scheme, which provides more IP addresses than did IPv4.

Here are examples of an FQDN, an IPv6 address, and an IPv4 address:

```
d6292.us.company.com
db8::01
10.23.2.3
```

2. Verify System and USS Process Limits

The following IBM system values are set in the SYS1.PARMLIB member BPXPRMnn and affect the number of TCP/IP sockets that SAS can use.

MAXSOCKETS system limit; specifies the maximum number of sockets that can be obtained for a given file system type. IBM recommends that this value be set to 10000.

MAXFILEPROC process limit; specifies the maximum number of file descriptors that a single process can have open concurrently, such as all open files, directories, sockets, and pipes. This value is usually set to 256. However, for heavy server use, it is advisable to set this number to 64000.

Please consult your IBM manual, *z/OS UNIX System Service Planning*, to get the most current IBM recommendations for your z/OS version.

For z/OS versions 1.7 and above, you may consult the *IBM Health Checker for z/OS User's Guide*. This will help you identify potential problems with your settings before they impact your availability.

You can check these IBM system values by using the following commands:

D OMVS ,OPTIONS is a z/OS command that will allow you to check the value of MAXFILEPROC.

D OMVS ,PFS is a z/OS command that will allow you to check the value of MAXSOCKETS.

Note: You can use the RACF ALTUSER or ADDUSER system commands to set MAXFILEPROC on a per-user basis. For complete details about MAXSOCKETS and MAXFILEPROC, see the IBM documentation *z/OS UNIX System Services Planning*.

3. Verify TCP/IP Host Name Configuration

IP Addresses

In order for a process to connect to a computer via TCP/IP, the process must know the IP address of the computer host name. To obtain the IP address, the process calls these name resolver functions:

getnameinfo() retrieves a string that contains its host name.

getaddrinfo() resolves the host name string to its IP address.

Because each host name is associated with a TCP/IP stack, it is critical that the host name be configured correctly for each TCP/IP stack.

TCP/IP Host Name Configuration for Communications Servers

When an IBM TCP/IP stack starts, the configuration process searches for the host name in the TCPIP.DATA data set.

- Search Order to Locate Stack Host Name
 1. If the IBM stack reads a TCPIP.DATA HOSTNAME configuration statement, it saves this value as the stack's host name.

2. If a TCPIP.DATA HOSTNAME configuration statement is not read, the TCP/IP stack searches for the Virtual Machine Communication Facility (VMCF) node name from VMCF and uses its node name as the stack's host name.

Note: VMCF should be running before any TCP/IP stacks are started.

3. If VMCF is not running when the TCP/IP stack is started, the TCP/IP stack's host name is set to the CVTSNAME, which is the SYSNAME=value in IEASYSnn that was used when the system was started.

- Multiple Host Names in a Single File

As an option, you can insert a prefix *system_name* in TCPIP.DATA configuration statements. Using prefixes enables you to configure multiple hosts in a single TCPIP.DATA data set. The *system_name* prefix is matched against the system name that the TCP/IP stack is started under. The *system_name* is identical to the VMCF node name. The TCP/IP stack reads and processes the TCPIP.DATA configuration statements in the order that they appear in the data set.

This example shows a HOSTNAME statement in a TPCIP.DATA data set:

```
SDCMVS:  HOSTNAME  PROD
SDCESA:  HOSTNAME  TEST
S390DEVA: HOSTNAME  DEV
```

The *system_name* is specified in the first column; the associated *host_name* is specified in the final column.

A TCP/IP stack that was started on the system named SDCMVS would set its host name to PROD. A TCP/IP stack that started on the system named SDCEA would set its host name to TEST. A TCP/IP stack that started on the system named S390DEVA would set its host name to DEV.

The following rules are used to process HOSTNAME statements:

1. If the *system_name* prefix does not match a host name, the configuration statement is ignored.
2. If the *system_name* prefix is not located, the configuration statement is applied to all hosts.
3. If the *system_name* matches a host name, the associated configuration statement is applied to that host.
4. The final configuration statement that matches a *system_name* remains in effect.

A HOSTNAME statement is ignored under these conditions:

- the *system_name* prefix did not match the VMCF node name
- VMCF is unavailable
- The system name does not match the MVS name of the system that the TCP/IP stack started under

Therefore, it is critical that VMCF is running before any TCP/IP stacks are started.

- IBM System Name Configurations

The Virtual Machine Communication Facility (VMCF) node name is used as the *system_name* prefix when processing IBM TCPIP.DATA configuration statements. The VMCF can be configured in two ways:

- as a restartable subsystem

If you have configured VMCF as a restartable subsystem, the node name is obtained from the value of the P= parameter in the EZAZSSI started procedure.

- as a non-restartable subsystem

If you configured VMCF as a non-restartable subsystem, the node name is specified in the IEFSSNnn member of PARMLIB.

Note: IBM recommends that the MVS system name be used for the VMCF node name specification. For details about configuring VMCF, refer to the IBM document z/OS Communication Server: IP Configuration Guide.

4. Verify TCP/IP Configuration Files Affecting SAS

When a TCP/IP stack is started, the TCP/IP stack reads one or more configuration files that contain statements that define its default behavior. Here are the configuration files:

IBM PROFILE.TCPIP File

The following PROFILE.TCPIP statements can restrict the ports that SAS servers can use:

PORT

reserves ports for server tasks. The PORT statement specifies only the job names and PROC names that are allowed access to the port.

PORTRANGE

same as PORT parameter but for a range of ports.

RESTRICT

defines a list of user IDs that are prohibited from using TCP/IP.

RESTRICTLOWPORTS

restricts the use of ports 1 to 1023 to specific job names or PROC names that are specified in the PORT or the PORTRANGE statement.

For details about the IBM PROFILE.TCPIP statements, see “References” on page 36.

The search order that is used by the IBM TCP/IP stack to locate PROFILE.TCPIP involves both explicit and dynamic data-set allocation, as follows:

1. //PROFILE DD DSN=
2. jobname.nodename.TCPIP
3. hlq.nodename.TCPIP

4. jobname.PROFILE.TCPIP
5. TCPIP.PROFILE.TCPIP

Note: IBM recommends explicitly specifying the PROFILE DD statement in the TCPIPROC JCL.

When the PROFILE DD statement is specified, no dynamic allocation is performed.

SAS does not access the PROFILE.TCPIP file directly. However, because this file is used to configure the IBM TCP/IP stack, the statements in this file can have an indirect effect on how SAS operates.

IBM TCPIP.DATA File

The TCPIP.DATA file contains the following statements that are used to configure the IBM TCP/IP stack and Communication Server applications.

TCPIPJOBNAME (or TCPIPUSERID)

specifies the member name of the procedure that is used to start the TCPIP address space, which is the TCP/IP stack name.

HOSTNAME

is used by the TCP/IP stack to determine its host name.

DOMAINORIGIN (or DOMAIN)

specifies the name of the domain origin, which is appended to the host name to form the fully-qualified domain name of the host.

DATASETPREFIX

is a high-level qualifier (hlq) for the dynamic allocation of data sets in IBM TCP/IP servers and clients.

For details about the IBM TCPIP.DATA statements, see “References” on page 36.

The IBM TCP/IP stack and the IBM Communication Server applications, including the IBM z/OS Resolver, use the following search order to locate the data set that contains the TCPIP.DATA configuration statements:

1. GLOBALTCPIPDATA value
2. RESOLVER_CONFIG environment variable
3. /etc/resolv.conf
4. SYSTCPD DD
5. userid.TCPIP.DATA
6. SYS1.TCPPARMS(TCPDATA)
7. DEFAULTTCPIPDATA value
8. TCPIP.TCPIP.DATA

5. Host Name Resolution for Systems that Run Multiple TCP/IP Stacks

There is usually a one-to-correspondence between a TCP/IP stack and its host name. The host name is obtained using the `gethostname()` function. However, for systems that run using multiple TCP/IP stacks, the identity of the stack's host name is ambiguous. Here is the process for resolving the host name for a multiple TCP/IP stack system:

- The process that calls `gethostname()` might be bound to a specific TCP/IP stack. The binding is referred to as TCP/IP stack affinity. If affinity to a specific TCP/IP stack has been established, the `gethostname()` function will return the host name for the specific stack.

There are several methods of setting the TCP/IP stack affinity. SAS uses the UNIX System Services call, `pfscctl(BPX1PCT)`. For details, see “Determining SAS TCP/IP Stack Affinity” below.

- Otherwise, the process returns the host name of the default TCP/IP stack. For details about the default stack, see “1. Verify TCP/IP Stack Definition” on page 22.

Determining SAS TCP/IP Stack Affinity

The SAS TCP/IP library uses the UNIX System Services call `pfscctl(BPX1PCT)` to determine whether the z/OS system is running a single TCP/IP stack environment (INET) or a multiple TCP/IP stack environment (CINET). Here is the process:

- If the `pfscctl()` call returns zero, the z/OS System is running an INET environment.
- If the `pfscctl()` call returns the number of CINET TCP/IP stacks and their names, the z/OS System is running a CINET environment.

1. The SAS TCP/IP library searches for the SAS environment variable, `TCPIPMCH`, in the data set that is specified by the `DDNAME`, `TKMVSENV`.

For details about specifying the `TCPIPMCH` environment variable and the `TKMVSENV` data set, see “7. SAS Environment Variables” on page 32.

- If the `TCPIPMCH` environment variable is set to a value of `""`, the SAS TCP/IP library will not attempt to set the TCP/IP stack affinity.
 - If the `TCPIPMCH` environment variable is set to a valid stack name, the SAS TCP/IP library will set the TCP/IP stack affinity to this value.
 - Otherwise the TCP/IP stack affinity will be set to the first TCP/IP stack name that was returned by the previous call to `pfscctl()`.
2. The SAS TCP/IP library resets the stack affinity by making another call to `pfscctl()` using the appropriate flags and specifying the TCP/IP stack name.

The value that is passed to `pfscctl()` must match the value of the `NAME` substatement, which is included in the `SUBFILESYSTYPE` statement, which is defined in the CINET PFS TCP/IP stack in the IBM `BPXPRMxx` parmlib member.

A name resolver is a set of routines that act as a client on behalf of an application to read a local host file or to access one or more domain name servers (DNS) for name-to-address or address-to-name resolution. Name resolution occurs by calling the name resolver functions `gethostbyname()` and `gethostbyaddr()`. A name resolver must be configured for each host.

IP address-to-host name mapping is performed by the Domain Name System if the `NSINTERADDR` statement(s) in the `TCPIP.DATA` file point to the IP address(es) of the Name Servers.

The following keyword/value combinations are involved in resolving host names in the TCP/IP Data file:

<code>NSINTERADDR</code>	value; Required
<code>NSPORTADDR</code>	value; Optional, will use default values
<code>RESOLVEIA</code>	value; Optional, will use default values
<code>RESOLVERTIMEOUT</code>	value; Optional, will use default values
<code>RESOLVERUDPRETRIES</code>	value; Optional, will use default values

Note: *NSINTERADDR is the only one that needs to be verified. The defaults for the others are sufficient for SAS to function properly.*

Note: *If Name Servers are not being used for Host Name Resolution, the NSINTERADDR statement(s) would be commented out.*

Using a Host Table

Using an FQDN with a Host Table

Note: *An administrator must make this change.*

The system host table must be defined such that an FQDN is used for each local network interface so that all clients will be able to use remote browsing.

For IPv4, SAS uses the `ETC.HOSTS` file for address resolution. This file lists IP addresses with names and aliases for each IP address. Only the entries for the addresses of the local network interfaces need to be changed.

`ETC.HOSTS` entries have the syntax (keywords in bold):

```
ipaddress      hostname alias alias
```

For the system to use the FQDN as the hostname, the first entry would be the FQDN, with aliases following as partial hostnames. For example:

```
10.4.86.63  sashost
```

becomes

```
10.4.86.63  sashost.mycompany.com sashost
```

For IPv6, hostnames are contained in the dataset `ETC.IPNODES`, which has the format:

```
Address Hostname1 HostName2 HostName3 ... Hostname35
```

Using the FQDN is similar to IPv4, with the first hostname specified as the FQDN. For example:

```
1:2:3:4:5:6:7:8 sashost
```

becomes

```
1:2:3:4:5:6:7:8 sashost.mycompany.com sashost
```

Refer to the IBM document *z/OS Communications Server: IP Configuration Guide* for more information on configuring hostnames.

If a DNS server is not available for Host Name Resolution, then a Host Table can be used. However, SAS cannot use the IBM TCP/IP file 'tcprefix.HOSTS.LOCAL' or the 'tcprefix.HOST.SITEINFO' due to the format of the data. See the following list for the default SAS search order of finding a HOSTS table:

1. ETC_HOSTS environment variable
2. //hfs:/etc/hosts z/OS (integrated sockets only)
3. tso-prefix.ETC.HOSTS under TSO
4. ETC.HOSTS
5. tcpip-prefix.ETC.HOSTS, if TCPIP_PREFIX is not blank.

To use a Host table not found with the default search order, SAS environment variables must be used. For details on setting up environment variables, see "7. SAS Environment Variables" on page 32.

6. Customizing UNIX System Services (USS) - Open Edition

The SAS TCP/IP library uses the UNIX System Services call `pfsc1(BPX1PCT)` to determine whether the z/OS system is running a single TCP/IP stack environment (INET) or a multiple TCP/IP stack environment (CINET). Here is the process:

- If the `pfsc1()` call returns zero, the z/OS System is running an INET environment.
- If the `pfsc1()` call returns the number of CINET TCP/IP stacks and their names, the z/OS System is running a CINET environment.
 - a. The SAS TCP/IP library searches for the SAS environment variable, `TCPIPMCH`, in the data set that is specified by the `DDNAME`, `TKMVSENV`.

For details about specifying the `TCPIPMCH` environment variable and the `TKMVSENV` data set, see "7. SAS Environment Variables" on page 32.

- If the `TCPIPMCH` environment variable is set to a value of "*", the SAS TCP/IP library will not attempt to set the TCP/IP stack affinity.
 - If the `TCPIPMCH` environment variable is set to a valid stack name, the SAS TCP/IP library will set the TCP/IP stack affinity to this value.
 - Otherwise the TCP/IP stack affinity will be set to the first TCP/IP stack name that was returned by the previous call to `pfsc1()`.
- b. The SAS TCP/IP library resets the stack affinity by making another call to `pfsc1()` using the appropriate flags and specifying the TCP/IP stack name.

The value that is passed to `pfscntl()` must match the value of the `NAME` substatement, which is included in the `SUBFILESYSTYPE` statement, which is defined in the `CINET PFS TCP/IP` stack in the `IBM BPXPRMxx parmlib` member.

7. SAS Environment Variables

TCIPMCH Environment Variable

Environment variables, which are specified in the file to which the `DDNAME TKMVSENV` points, are used to customize TCP/IP for SAS 9.2. SAS 9.2 uses the `TCIPMCH` SAS environment variable to alter default processing for TCP/IP initialization.

The `TCIPMCH` SAS environment variable is useful at sites that simultaneously run multiple TCP/IP packages: either multiple TCP/IP vendor packages or multiple instances of the same vendor's TCP/IP. The `TCIPMCH` environment variable is used to specify the name of the TCP/IP stack name, such as a started task. Setting this environment variable is the equivalent of the `TCPIPJOBNAME/TCPIPUSERID` configuration keywords within the `IBM TCPIP.DATA` file. If the default value for the TCP/IP stack name is not specified, the value is the first TCP/IP stack that is defined to the system.

For information about setting `TCIPMCH`, see “Configuring SAS to Use the IBM z/OS Name Resolver” on page 34.

TKMVSENV Data Set

A SAS data set can be used to specify the SAS environment variables, which is referred to as the `TKMVSENV` data set file. If you use SAS environment variables, you must allocate the `TKMVSENV DD` in the `JCL` or `CLIST` that executes SAS 9.2.

For example, here are the allocation statements for the data set `SAS.DATA.TKMVSENV`, which contains the desired environment variable information:

Note: *Line numbering in the `TKMVSENV` data set must be disabled.*

BATCH statement:

```
//TKMVSENV DD DISP=SHR,DSN=SAS.DATA.TKMVSENV
```

TSO statement:

```
ALLOC F(TKMVSENV) DA('SAS.DATA.TKMVSENV') SHR
```

Each logical record contains an environment variable assignment in this format:

```
SET environment_variable_name=value
```

TCP/IP Name Resolver Configuration

A name resolver is a set of routines that act as a client on behalf of an application to read a local host file or to access one or more domain name servers (DNS) for name-to-address or address-to-name resolution. Name resolution occurs by calling the name resolver functions `getnameinfo()` and `getaddrinfo()`.

A name resolver must be configured for each host. Here are the locations for UNIX configuration files:

- /etc/hosts contains the local host configuration data
- /etc/resolve/conf contains the DNS domain name and the name servers' IP addresses
- /etc/service contains the service configuration data

IBM z/OS Name Resolver

Starting with z/OS V1R4, IBM introduced support for the Internet Protocol Version 6, IPv6, which is the successor to the Internet Protocol, IPv4. In order to support IPv6, new USS BPX calls for the IBM z/OS Name Resolver were introduced to implement the protocol-independent resolver functions that are described in the RFC 3493 specification. Here is a list of the IBM z/OS Name Resolver functions that are supported in IPv6 and IPv4:

IPv6	IPv4
getnameinfo(BPX1GNI)	gethostbyname()
getaddrinfo(BPX1GAI)	gethostbyaddr()

When the IBM z/OS Name Resolver is started, it reads the IBM configuration file that is pointed to by the DD statement SETUP, which can contain the following SETUP directives.

- COMMONSEARCH | NOCOMMONSEARCH
- DEFAULTIPNODES
- DEFAULTTCPIPDATA
- GLOBALIPNODES
- GLOBALTCPIPDATA

Note: The most important SETUP directives are GLOBALTCPIPDATA and DEFAULTTCPIPDATA

GLOBALTCPIPDATA

identifies a global TCPIP.DATA file. Any TCPIP.DATA directive that is specified in this file are system-wide and cannot be overridden by a local TCPIP.DATA file.

DEFAULTTCPIPDATA

identifies a default TCPIP.DATA file, which overrides the TCPIP.DATA file that is named TCPIP.TCPIP.DATA.

If a GLOBALTCPIPDATA statement is located in the resolver setup file, the IBM z/OS Name Resolver will read any name resolver directives that are located in this global TCPIP.DATA file. The IBM z/OS Name Resolver will then search for a local TCPIP.DATA file in this order:

1. RESOLVER_CONFIG environment variable
2. /etc/resolv.conf
3. SYSTCPD DD
4. jobname.TCPIP.DATA

5. SYS1.TCPPARMS (TCPDATA)
6. DEFAULTTCPIPDATA value (if specified in the z/OS Name Resolver setup file)
7. TCPIP.TCPIP.DATA

Some useful IBM z/OS Name Resolver Server directives include:

LOOKUP

changes the order in which name resolution is performed between a DNS name server and a local hosts file. Using the LOOKUP directive, you can specify DNS only, LOCAL only, DNS LOCAL, or LOCAL DNS. By default, a DNS name server is queried first. If DNS fails, then DNS LOCAL is used.

SEARCH

specifies a search of up to six domains, in the specified order. The first domain name that is specified is used as the value for DOMAINORIGIN. If both the SEARCH and DOMAINORIGIN statements are specified, the one that appears last is used.

SORTLIST

specifies up to four IP addresses to use for a specific host. If DNS returns more than one IP address for a host, SORTLIST can use search masks to sort and identify which IP address the resolver returns.

OPTIONS

specifies that for a domain name that contains n or more periods (.), the resolver should look up the name as is before applying the DOMAINORIGIN or SEARCH statement settings. The range of n is 1 to 15. The default is 2.

For complete information about these directives, see the IBM documentation *z/OS IP Configuration Guide and IP Configuration Reference*.

Configuring SAS to Use the IBM z/OS Name Resolver

SAS 9.2 uses the IBM z/OS Name Resolver, by default. No configuration is necessary unless SAS is running under a multiple TCP/IP stack system (CINET). If SAS is running under a multiple TCP/IP stack system, the SAS TCP/IP library will need to set the TCP/IP stack affinity. Specify this SAS environment variable:

```
set TCPIPMCH=stack-affinity
```

TCPIPMCH is a SAS environment variable that is used to specify the name for the TCP/IP stack, which is also known as a started task. TCPIPMCH is equivalent to the TCPIPJOBNAME and TCIPUSERID configuration keywords that are used in the IBM TCPIP.DATA file.

If a value is not specified for TCPIPMCH, the SAS TCP/IP library will use the first TCP/IP stack name that is returned by a call to the UNIX System Service, pfsctl(), which might not be the appropriate TCP/IP stack. For information, see "TCIPMCH Environment Variable" on page 32.

The Services File

The SERVICES file defines port resources that are used when TCP/IP is used to connect client/server sessions. Examples of configured port services include the Telnet port, spawner ports, MP CONNECT pipes, and SAS/SHARE servers. For more information, see *Configuring the SERVICES File*.

Configure SAS Services

A service for each SAS server session (SAS/CONNECT or SAS/SHARE) must be defined in the SERVICES file on each computer that a SAS client session runs on.

Note: Some TCP/IP stacks can restrict the range of ports that can be used. For details, see *TCP/IP stack Configuration Files*.

Sample TCP/IP SERVICES File

Some SAS products (products which utilize a server) require an entry in the TCP/IP SERVICES file. Please see individual product requirements to verify the need for an entry into the TCP/IP SERVICES file.

Entries in the ETC.SERVICES file will follow the sample form below:

```
#   this is a comment in the ETC.SERVICES file
#
telnet  23/telnet
ftp     21/tcp
sassrv1 5010/tcp      #   SAS/SHARE Server number 1
sassrv2 5011/tcp      #   SAS/SHARE Server number 2
appsrv1 5224/tcp      #   SAS/IntrNet Application Server number 1
appsrv2 5225/tcp      #   SAS/IntrNet Application Server number 2
spawner1 5227/tcp     #   z/OS SAS/Connect Spawner
```

The Services File Search Order

The z/OS Resolver searches for the SERVICES file, using this order:

1. Value of the ETC_SERVICES environment variable
2. /etc/services
3. tso-prefix.ETC.SERVICES under TSO or user-ID.ETC.SERVICES under batch execution
4. ETC.SERVICES
5. TCPIP.ETC.SERVICES
6. tcpip-prefix.ETC.SERVICES

8. Diagnosing Configuration Problems

Executing the SAS code “PROC TCPTEST;run;” will provide information on how SAS sees the TCP/IP configuration.

References

- z/OS V1R9.0 Communication Server: IP Configuration Guide, IBM SC31-8775-11
- z/OSV1R9.0 Communication Server: IP Configuration Reference, IBM SC31-8776-12
- z/OS V1R9.0 UNIX System Services Planning, IBM GA22-7800-12
- DNS and BIND, 4th Ed., by Paul Albitz & Cricket Liu, O'Reilly and Associates, Inc.

Customizing Your SAS CLIST, REXX Exec, and Cataloged Procedure

Note: You should complete this task if appropriate for your installation.

There are CLIST, REXX exec, and PROC examples created in the CNTL data set for invoking SAS 9.2 Foundation. You can invoke the encoded SAS 9.2 Foundation using CLSTxx, REXXxx, or PROCxx where xx is the encoding value that can be found in the *Languages, Encodings and Installation Codes* table in the Installation Instructions. (All installations of SAS 9.2 Foundation, whether domestic or foreign, are now encoded images. Encoded images allow SAS 9.2 Foundation to be invoked using different character sets. For example, W0 is English and W3 is German. Other examples of the encoding abbreviations can be found in the *Installation Instructions*.)

The installation job(s) that you ran to complete the installation created tailored versions of the SAS CLIST, REXX exec, and cataloged procedures.

- ❑ The tailored CLISTs were written to the CNTL data set as member CLSTxx for the single-byte image and DLSTxx for the double-byte image, and they were also written to the command procedure library, &prefix.CLIST.
- ❑ The tailored cataloged procedure was written to the CNTL data set as member PROCxx for the single-byte image and DROCxx for the double-byte image. The installation job(s) copied it to the procedure library, &prefix.PROCLIB.
- ❑ The tailored REXX execs were written to the CNTL data set as member REXTxx and copied to the SASRX data set as member REXXxx, and configuration files for the REXX execs were written to the CNTL data set as member REXCxx and copied to the SASRXCFCG data set as member REXXxx. Note that the REXXxx exec is actually just a stub that passes the name of a configuration file to another REXX exec named SASTREXX that contains the greater part of the code. You need only one copy of SASTREXX but it can be shared by multiple instances of the REXXxx code, each of which has its own configuration file in the SASRXCFCG data set.

Note: Only Action A moves the CLIST, REXX execs, and cataloged procedure into the default libraries. Actions B and C require that you manually copy and rename CLSTxx, REXCxx, REXTxx, and PROCxx to the appropriate libraries.

To make the CLIST generally available, you will need to include the &prefix.CLIST data set in your SYSPROC concatenation or else copy the members to a data set that is in the concatenation.

To make the REXX exec generally available, you will need to include the &prefix.SASRX data set in either your SYSPROC or SYSEXEC concatenation, or else copy the members to a data set that is in one of the concatenations.

You may wish to rename the CLIST from CLSTxx to SAS and to rename the REXX exec from REXXxx to SASRX for consistency with names used in SAS 9.2 Companion for z/OS.

Important: If you choose to install the entire SAS 9 load LIBRARY into the LINKLIST or LPALST, a member named SAS will be installed, which will cause a conflict if the name chosen for the CLIST or REXX exec is also SAS. Possible circumventions include:

- Execute the CLIST or REXX exec explicitly (for example, "exec 'tso.clist.lib(SAS)'").
- When the SAS CLIST or REXX exec is being invoked from ISPF, add "SAS" to the ISPF TSO Command Table (ISPTCM) as a CLIST with the appropriate FLAGBYTE setting to indicate it is to be treated as a CLIST. See IBM documentation for details.
- Execute the "SAS" CLIST or REXX exec with prefix of % (for example, "TSO %SAS").
- Use a CLIST or REXX exec name other than "SAS."

You can further customize these tailored versions of the CLIST, REXX exec, and cataloged procedure. For example, the CLIST includes statements that allocate a permanent SASUSER data library for each user. If no permanent SASUSER data library exists for the user, the CLIST creates one. If you do not want each user at your site to maintain an individual permanent SASUSER data library, you can remove these statements from the CLIST. You might also want to make changes as part of selecting a bundled configuration to run or as part of installing SAS 9.2 Foundation in the LPA/ELPA, as previously described. STEP 1 in this section describes some of the changes you might want to make.

Important: *If you are using the REXX exec, most customizations can be made in the corresponding configuration file in the SASRXCFG data set. With the REXX exec, there is less need to modify the code than there is with the CLIST.*

Determine the changes you want to make as described in STEP 1 and apply the changes according to your standard procedures. Make the changes to the CLIST, REXX exec, and cataloged procedure in the libraries to which they were copied.

STEP 1: Determine the customizations you need.

Review the following information that discusses changes you might want to make. You might also have some site-specific issues to address.

Product-specific customizations

The appendices describe product-specific customizations that might be required. Review the appendices for the products you are installing to see what customizations to the SAS CLIST, REXX exec, and PROC you need.

SASUSER Considerations

The SAS 9.2 Foundation CLIST allocates a permanent `SASUSER` data library for each user the first time the user invokes SAS 9.2 Foundation. When there is no `SASUSER` data library allocated to a session, the system by default assigns the `SASUSER libref` to the temporary `WORK` data library. In this case, data written to `SASUSER` disappears when the `WORK` data library is deleted.

Although individual `SASUSER` data libraries are not required, they allow users to take advantage of many interactive features in SAS 9.2 Foundation. The SAS/ASSIST product uses the `SASUSER` library to store all SAS data sets created and to save all work (programs, output, and catalogs) from a SAS/ASSIST session. The SAS windowing environment uses this library for storing various types of information. Users can save profiles in their `SASUSER` data libraries to customize window sizes, function key settings, and other aspects of the SAS full-screen environment. You can use SAS windowing environment `SAVE` and `COPY` commands to transfer data or program statements between windows and catalogs in your `SASUSER` data library. The `FORMS` command stores forms entries used in printing from the windowing environment.

The default CLIST creates a permanent `SASUSER` data library for each user who invokes SAS 9.2 Foundation using the CLIST statements, as shown in the following:

```

IF &SYSPREF EQ THEN +
    SET &USRPREF=&SYSUID
ELSE +
    SET &USRPREF=SYSPREF

IF &STR(&SASUSER) EQ THEN +
    SET &SASUSER = &STR('&USRPREF..SAS9.SASUSER')
SET STATE = &SYSDSN(&SASUSER)
IF &STATE = OK THEN +
    ALLOC F(&DDSASUSR) DA(&SASUSER) OLD REU
ELSE +

IF &STATE = DATASET NOT FOUND THEN DO
    WRITE Warning: SASUSER file does not exist, will be created.
    ALLOC F(&DDSASUSR) DA(&SASUSER) NEW CATALOG +
        SP(30 5) ROUND DSORG(PS) RECFM(F S) +
        BLKSIZE(6144) REU
    END
ELSE DO
    WRITE Warning: SASUSER file: &STATE
    WRITE SASUSER file not allocated to this session
    END

```

You might want to determine a different naming convention for `SASUSER` data libraries or alter the default space allocation.

The REXX exec creates a permanent `SASUSER` data library for each user according to the specification of the `SASUSER` option in the `SASRXCFG` file. The default specification is

```
-SASUSER '&syspref.SAS9.SASUSER'
```

where `&syspref` is replaced with the user's system prefix, or if the prefix is null, then with the `userid`. You can change this default specification in the `SASRXCFG` file to suit your requirements, or replace it with `-NOSASUSER` if you do not want to use `SASUSER` libraries.

Special Cataloged Procedure Parameters

The default cataloged procedure includes two symbolic parameters that allow concatenation of user libraries before your SAS 9.2 Foundation installation libraries:

- **LOAD=**

The `LOAD=` parameter allows you to specify a user load library `DSN` to concatenate before the SAS load library data set.

- **SASAUTO=**

The `SASAUTO=` parameter allows you to specify a user autocall macro library `DSN` to concatenate before the system autocall macro library.

These parameters are intended to provide added flexibility for invoking SAS 9.2 Foundation in batch mode. Please note that any existing JCL you use to invoke earlier releases of SAS 9.2 Foundation might not work as expected with the new default PROC. If large volumes of production JCL at your site contain DD overrides for `//SASAUTOS` and `//STEPLIB`, you might want to customize the `SAS9 PROC` to change the concatenation order of the data sets for these DD statements.

□ **Entry Name Considerations**

If you plan to run a bundled configuration, change the default `ENTRY` name in your CLIST and cataloged procedure. If you are using the REXX exec, make the change in the `SASRXCFG` config file, not in the REXX exec. The default is `SAS`. Valid standard `ENTRY` names are as follows:

- `SAS` for z/OS non-bundled configuration
- `SASB` for z/OS non-LPA bundled configuration
- `SASLPA` for z/OS LPA bundled configuration.

See the sections "Selecting a Bundled Configuration" on page 18 and "Installing SAS 9.2 Foundation into the LPA/ELPA" on page 19 for considerations relating to `ENTRY` name selection.

□ **Running Multiple Versions of SAS 9.2 Foundation Concurrently**

If your users run multiple versions of SAS 9.2 Foundation concurrently in the same TSO session, you might also want to customize the CLIST to avoid `DDname` conflicts.

To do so, determine a naming convention for the SAS file `DDnames` allocated, such as `WORK`, and specify your `DDnames` in the CLIST. The CLIST includes special `DDname` operands for you to use when specifying your SAS file `DDnames`. When you use these operands, the CLIST specifies the appropriate corresponding SAS system options for you. Specify alternate `DDnames` in this manner for any files that would cause conflicts at your site.

If you are using the REXX exec, add the alternate DDname options to the SASRXCFG config file, not in the code of the exec.

❑ Further DDname Considerations

Like the Version 6 SAS System, note that SAS 9.2 Foundation no longer uses the FORTRAN-style DDnames that were used in Version 5 for the SAS log, print, and PARMs data sets. If you want to use the same DDnames in SAS 9.2 Foundation as in Version 5, you need to customize your CLIST, PROC, and system default options.

Customize the CLIST and PROC by changing the DDnames as follows:

- change SASLOG to FT11F001
- change SASLIST to FT12F001
- change SASPARM to FT15F001.

Customize your default options by adding the following option values to your DFLTOPTS table or system configuration file.

- LOG=FT11F001
- PRINT=FT12F001
- PARMCARDS=FT15F001

If you need more information about these options, page 16 of the section “Setting up SAS 9.2 Foundation” contains details on customizing your DFLTOPTS table and system configuration file.

❑ SORT Library Considerations

If your users run PROC SORT and your site does not provide your system sort routine in a linklist library, set the SORTLINK CLIST operand to null and specify the load library that contains your system sort routine in the SORTLDSN operand. If you are using the REXX exec, add these options in the SASRXCFG config file. In the cataloged procedure, concatenate your system sort load library to the STEPLIB DD statement.

STEP 2: Make the CLIST and PROC changes according to standard procedures at your site.

Customizing Your NEWS File

Note: You should complete this task if appropriate for your installation.

The installation process unloads the default NEWS member into your &prefix.NEWS partitioned data set. You can update this member with information appropriate for your site.

Information contained in the NEWS data set is displayed on the SAS log at invocation time when the NEWS= SAS system option is specified. The NEWS= option specifies either a logical or a physical name for the NEWS data set. The default system configuration files loaded into your CNTL data set at installation time contain the NEWS= system option specifying the physical name of the NEWS member in the NEWS data set allocated by the installation jobs. You can modify the NEWS member to contain any information appropriate for your site, or, if you do not want to display standard information at invocation time, you can remove the NEWS= option from your system configuration files.

Installing the SAS 9.2 SVC Routine

Note: This task is optional, but recommended.

STEP 1: Decide whether to install the SAS 9.2 Foundation SVC routine.

In most cases, where the function provided or supported by the SVC routine is not utilized or required, installation of the SAS 9.2 Foundation SVC routine is *not* absolutely necessary.

However, this step should be completed for the following reasons:

- At a later time it might be decided to utilize the otherwise unavailable functions.
- A SAS product might be acquired which requires the SAS 9.2 Foundation SVC routine.

Installation of the SAS 9.2 Foundation SVC is **absolutely required** in any of these six situations:

- The SMF SAS system option is used to write SAS user SMF records.
- SAS/SHARE 9.2 product is installed.
- The SAS/IntrNet product is installed and `PROC APPSRV` is used with the `AUTH=HOST` option.
- The SAS/CONNECT Spawner for z/OS or SAS/CONNECT to the same multi-process machine on z/OS is to be implemented.
- The SAS/Integration Technologies Object Spawner is installed.
- You are installing either the OLAP server or the OMR server in SAS 9.2 Foundation and you will be running these servers with the `SECURITY` option enabled (the default).

Note: For all products requiring the SAS/CONNECT Spawner or the SAS/Integration Technologies Object Spawner, please refer to the System Requirements documentation.

The SAS 9.2 Foundation SVC routine provides all the functions available with the SAS SVC in previous releases and can be used in place of prior releases' SAS SVC routines. However, SAS 9.2 Foundation is not compatible with prior releases of the SAS SVC routine. Therefore, if the SAS SVC routine is to be used with SAS 9.2 Foundation, the SAS 9.2 Foundation SVC routine must be installed.

System Integrity Guidelines

The SAS 9.2 Foundation SVC Routine has been designed, written and tested using IBM guidelines for system integrity. When installed properly, the SAS 9.2 Foundation SVC routine cannot be used to obtain control in an authorized state, nor bypass system security or password protection.

STEP 2: Select the type of SVC to install.

The SAS 9.2 Foundation SVC routine can be installed in one of two ways:

- As a Type 4 Extended Support Router SVC (ESR SVC 109) entry. It is recommended that the SAS 9.2 Foundation SVC routine be installed as a Type 4 ESR SVC (SVC 109). This technique has several advantages. One is that a user SVC reserved exclusively for the SAS 9.2 Foundation SVC is not required. Additionally, if the Type 4 ESR SVC routing code chosen is selected for use by another software vendor, it is relatively easy to change the routing code used by the SAS SVC.

To choose the Type 4 ESR SVC (SVC 109) routing code to use, first determine which routing codes are already installed or in use by the operating system or other software products. To do this, list the names of members beginning with IGX00 in the SYS1.LPALIB and all other libraries listed in the LPALSTxx member of SYS1.PARMLIB. Also check the IEALPAXx member of SYS1.PARMLIB for IGX00nnn modules that can be placed in MLPA. The nnn suffix is the routing code (always in decimal) by which the ESR SVC routine is invoked. For example, the ESR SVC routine IGX00219 would be invoked by loading register 15 with the decimal value 219 and then executing an SVC 109 instruction. IBM has reserved routing codes between 200-255 for customer use under z/OS. SAS recommends that you choose a routing code within this range. However, the default routing code is 4, for compatibility with previous releases of SAS 9.2 Foundation. It is necessary to choose an unused routing code to ensure its integrity.

- As a standard “user” SVC (SVC Routines 200-255) defined in member IEASVCxx of SYS1.PARMLIB. To install the SAS 9.2 Foundation SVC as a “user” SVC routine, ensure that the selected user SVC number is currently unused. Check the IEASVCxx member in SYS1.PARMLIB. For example, to install the SAS 9.2 Foundation SVC as SVC 200, code the following SVCPARM statement in IEASVCxx:

```
SVCPARM 200,REPLACE,TYPE(4)
```

Note that the SAS 9.2 Foundation SVC is installed as a Type 4, preemptive, unauthorized SVC with no locks held.

The first “load” of a Type 4 SVC routine is named according to z/OS convention. That is IGC00nnc, where nnc is the zoned EBCDIC representation of the SVC routine’s number, resulting from the unpacking of the positive, packed decimal value that is the SVC routine number. For example, the first load of a Type 4 SVC routine invoked using SVC 234 would be named IGC0023D. This is because x'234', when unpacked, yields x'F2F3C4' or C'23D'.

STEP 3: Copy and Rename the SAS 9.2 Foundation SVC routine into SYS1.LPALIB or a LNKSTxx library.

Copy and rename the SAS 9.2 Foundation SVC into SYS1.LPALIB or any other LPA library pointed to by the LPALSTxx member of SYS1.PARMLIB. Optionally, the SAS 9.2 Foundation SVC can be installed into a LNKSTxx library and brought into LPA at IPL time by a specification in the IEALPAXx member of SYS1.PARMLIB and an MLPA=xx specification in the IEASYS00 member of SYS1.PARMLIB.

The load module (SVC0MVS) must be installed into SYS1.LPALIB (or other appropriate library) with a valid SVC name. A utility like IEBCOPY could be used to copy the SAS 9.2 Foundation SVC routine into an appropriate LPA library. Refer to member SVC0CPYJ in your &prefix.BAMISC SAS installation library for a sample job.

Alternatively, SMP/E can be used to install the SAS 9.2 Foundation SVC as an SMP/E USERMOD. Refer to member SVC0SMPJ in your &prefix.BAMISC SAS installation library for a sample job.

You must IPL after the SAS 9.2 Foundation SVC routine is actually copied into an appropriate operating system library. Specify the CLPA parameter in response to the IEA101A message.

STEP 4: Verify and update SAS 9.2 Foundation options for the SAS SVC.

If the default values are not used, these options need to be set in the Restricted Options Table. See “Customizing Default Options and System Configuration Files” on page 11 for details on creating a Restricted Options Table.

The following SAS system options are directly related to the SAS 9.2 Foundation SVC routine and the manner in which it is installed. You must set these options as described in the following to invoke the SAS 9.2 Foundation SVC routine correctly.

❑ **SVC0SVC=**

The default is 109 for the ESR SVC 109. If using the “user” SVC instead of the ESR SVC, this option should be set to the SVC number that was defined in STEP 2.

❑ **SVC0R15=**

This option only applies if the SAS 9.2 Foundation SVC was installed as an ESR Type 4 SVC. The default is 4 for compatibility with previous releases of SAS 9.2 Foundation. It should specify the routing code that was chosen when the SAS 9.2 Foundation SVC was installed into your operating system.

STEP 5: Verify and update SAS 9.2 options for the SAS TK SVC.

Note: This section is for the SAS/CONNECT Spawner, the Integration Technologies Object Spawner, and any product that requires these two products.

If the default values are not used, the options need to be set in the ``&prefix.TKMVSENV(TKMVSENV)` data set with the file reference of `TKMVSENV`.

The following SAS TK options are directly related to the SAS 9.2 SVC routine and the manner in which it is installed. You must set these options as described below to invoke the SAS 9.2 SVC routine correctly.

- `set TKOPT_SVCNO=`

The default value is 109 for the ESR SVC 109. If using the “user” SVC instead of the ESR SVC, this option should be set to the SVC number that was defined in STEP 2.

- `set TKOPT_SVCR15=`

This option only applies if the SAS 9.2 SVC was installed as an ESR Type 4 SVC. The default is 4 for compatibility with previous releases of SAS 9.2. It should specify the routing code that was chosen when the SAS 9.2 SVC was installed into your operating system.

Installing the SAS SMF Exit

Note: This procedure is optional.

The `SMF` SAS system option controls whether SMF records formatted by SAS 9.2 Foundation are written to the SMF file at the termination of every SAS Software step. If you intend to enable the SMF option in order to write SMF records, and if you would like to tailor the SMF records that SAS 9.2 Foundation writes, you must install the SMF exit and set the `SMFEXIT=` system option.

STEP 1: Decide whether you need to install the SMF exit.

You can use the SMF exit to examine the SMF record that SAS 9.2 Foundation has formatted, modify fields within the record, write the record to a user file, and suppress the writing of the record by SAS 9.2 Foundation.

Note: If SMF records are to be written to the SMF file, the SAS 9.2 Foundation SVC must be installed. Please see "Installing the SAS 9.2 Foundation SVC Routine" on page 41 for more information.

The use of the SMF exit is entirely optional. Even if it is not installed, records are written to the SMF file if the `SMF` and `STIMER` options are in effect and the SAS 9.2 Foundation SVC is installed. If you install the SMF exit, you can still use the `SMFEXIT=` option to specify whether or not it is to be invoked. If you do not specify the `SMFEXIT=` option, the exit is not invoked. If the value specified is the name of a load module in the search path, the exit is invoked.

STEP 2: Tailor the SMF exit source to meet the requirements of your site.

The sample SMF exit is an assembler source module that must be modified to suit your needs. The source for the sample SMF exit is in the `BAMISC` library member `SMFEXIT`.

Note: In the z/OS environment, the exit is entered in `AMODE 31`. If you are writing to a user file using an access method that requires you to be in `AMODE 24`, change `AMODE` for the access method calls, then return to `AMODE 31` before returning from the exit.

The exit is called at SAS 9.2 Foundation initialization, at SAS Software step termination, and at SAS 9.2 Foundation termination. At entry to the exit, `R15` contains the entry point address, `R14` contains the return address, `R13` points to a standard register save area, and `R1` contains the address of a fullword. If the fullword is `0`, the call is being made after SAS 9.2 Foundation is initialized so that the exit can perform any initialization necessary. If your exit is writing records to a user file, you probably want to open the file on this call. If the fullword is `-1`, the call is being made before SAS 9.2 Foundation is terminated so that the exit can perform any termination necessary. If your exit is writing records to a user file, you probably want to close the file on this call. If the fullword contains neither `0` nor `-1`, it is assumed to be the address of the SMF record to be written.

If the exit returns `0` in `R15`, SAS 9.2 Foundation writes the SMF record pointed to by `R1`. If the exit returns a non-zero value in `R15`, SAS 9.2 Foundation suppresses the writing of the SMF record. The following approaches might be taken in the exit:

- Zero `R15` and return immediately to write the SMF record, as is.
- Return immediately leaving a non-zero value in `R15` to suppress the writing of the record.

- Modify the record pointed to by R1, as desired, including adding data in the user area provided, or perhaps changing the record type. Note, however, that record type must be greater than 127. If it is not, the SAS 9.2 Foundation SVC does not write the record to the SMF file. Note also that the record length in the standard header on input does not include the user area. If data is added in this area, the length field must be incrementally lengthened by the number of bytes added. Return a 0 in R15 to cause SAS 9.2 Foundation to write the modified SMF record.
- Modify the record pointed to by R1, as desired, and write the record to a user file. In this case, record-type checking is up to you. Return a non-zero value in R15 to cause SAS 9.2 Foundation to suppress writing of the SMF record. The format of the record formatted by SAS 9.2 Foundation is as follows:

Hex Offset	SMFRECD	DSECT		
00	SMFRLEN	DS	BL2	Record length
02	SMFSEG	DS	BL2	Segment descriptor
04	SMFFLG	DS	BL1	Header flag
05	SMFRTP	DS	BL1	Record type
06	SMFTIME	DS	BL4	Time given to smf
0A	SMFDATE	DS	PL4	Date given to smf
0E	SMFSID	DS	CL4	System id
12	SMFJOB	DS	CL8	Jobname
1A	SMFRTR	DS	BL4	Reader time
1E	SMFRDTE	DS	PL4	Reader date
22	SMFSTEP	DS	XL1	Step number
23	SMFRSVD	DS	XL1	Reserved
24	SMFPROC	DS	CL8	Proc name
2C	SMFCPU	DS	F	Proc CPU time in timer units
30	SMFEXCP	DS	F	Proc excp count
34	SMFCORE	DS	F	Proc storage used
38	SMFVUSE	DS	F	Vector usage in .01 sec
3C	SMFVAFF	DS	F	Vector affinity time in .01 sec
40	SMFHSP	DS	F	RSM hiperspace time in .01 sec
44	SMFUSER	DS	XL64	User space (Not included in SMFLEN)

STEP 3: Assemble and link the tailored source.

The JCL required to assemble and link the SMF exit is located in member `BASMF` in the `CNTL` data set. You can tailor the link step so that the name supplied on the `ENTRY` statement is the entry point that you want to use when invoking the exit. `SMFEXIT1`, which is the name on the `ENTRY` statement in the JCL, is the entry point in the sample SMF exit that simply zeroes `R15` and returns, causing the SMF record to be written to the SMF file as formatted by SAS 9.2 Foundation.

STEP 4: If your site requirements dictate that SMF always be ON, and that the SMFEXIT= always be set, ensure that SMF, SMFTYPE=, SMFEXIT=, STIMER, SVC0SVC=, and SVC0R15= are set in the Restricted Options Table.

See “Customizing Default Options and System Configuration Files” on page 11 of this document for more information.

Installing UNIX File System Components

Beginning with SAS 9.2, help documentation and products that include additional optional functionality provided through Java require components to be installed into the Unix file system. In SAS 9.2 Foundation, products that provide optional Java functionality include:

- Base SAS
- SAS/Access Interface to R/3
- SAS Enterprise Miner
- SAS/GRAPH support for the `javaimg` device
- SAS IT Resource Management
- SAS/OR
- SAS/SECURE

If you want to use this functionality, you need to install components into the UNIX file system. This section documents the steps necessary to install the components.

The installation of the UNIX file system components of SAS 9.2 Foundation for z/OS requires seven steps:

1. Important: Validate/determine the UNIX File System Components directory structure.
2. Verify or obtain permissions, or enlist z/OS system personnel required to do steps 3, 4, and 7.
3. Allocate the HFS data set to contain the UNIX file system for the SAS components (manually, or run the `HFSCREAT` utility).
4. Mount the data set to enable UNIX file system space for the SAS components (manually or run the `HFSMOUNT` utility).
5. Run the `USSUNTAR` utility to install the SAS components into the UNIX file system.
6. Update system environment variables to access the SAS components.
7. Assign a permanent mount point for the file system in `SYS1 . PARMLIB (BPXPRMxx)`.

STEP 1: Important: Validate or determine the UNIX File System Components directory structure.

The three utilities mentioned above are created by SASNEW or SASIHOLD in actions A, B, or C, at the same time the main install is created, using the same SASEDITP values used to generate the main install. They will be found afterward in the installation control data set, abbreviated &CNTLDSN.

These utilities will use default values for the HFS container data set and for the directory structure.

Important: *It is **highly recommended** that the installer use the default values for the UNIX file system SAS directory root without modification. These values will guarantee the ability to properly apply any possible maintenance or additional product material in the future. If the system personnel accomplish steps 3 and 4 without using the supplied utilities, they should still use the mount path specified in &CNTLDSN(HFSMOUNT). Also, mixed case characters must be preserved in order to maintain the integrity of the HFS directory structure.*

The directory structure root will default to:

```
/usr/lpp/SAS/SAS_9.2/USER.PREFIXyy
```

where

- /usr/lpp/SAS/ is assigned by IBM for SAS applications.
- /usr/lpp/SAS/SAS_9.2/ is the mount point used by default by &CNTLDSN(HFSMOUNT).
- USER.PREFIX is the high-level prefix specified by the installer in &CNTLDSN(SASEDITP), abbreviated as "&prefix" and used by the main install for allocating traditional z/OS data sets, etc.
- yy represents the encoding found in the "Encode (yy) value" column in the "Languages, Encodings, and Installation Codes" table in *Installation Instructions for SAS 9.2 Foundation for z/OS*.

The utilities supplied use the hierarchical file system (HFS). If you are running z/OS and would like to use the zSeries file system (zFS), consult the appropriate IBM z/OS manuals identified in the steps below.

STEP 2: Verify or obtain permissions, or enlist z/OS system personnel required to execute steps 3, 4, and 7.

These steps will likely require the involvement of the z/OS UNIX Systems Services system administrator (sysadmin) as well as of the z/OS Systems Programmer (sysprog).

Showing the generated HFSCREAT and HFSMOUNT JCL jobs to system personnel might suffice to explain what is needed.

System personnel might choose to accomplish steps 3 and/or 4 without using the supplied utilities. They might run the utilities themselves. Or they might provide the installer appropriate permissions and have the installer run the utilities. Step 5 should always be done using the USSUNTAR utility and should be run by the installer to establish correct HFS permissions. Step 6 should be done by the installer or whoever has knowledge of user customizing needs. Step 7 should be completed by a systems programmer.

If the installer has or is given appropriate permissions, then language below referring to z/OS systems personnel should be read as directing the installer.

STEP 3: Allocate the HFS data set to contain the UNIX file system for the SAS components (manually, or run HFSCREAT utility).

This step allocates the z/OS HFS data set that will contain the UNIX style file system directories and files of SAS 9.2 Foundation for z/OS. We recommend that you install these components into their own file system to simplify copy, backup, and recovery. The instructions presume HFS is being used.

As part of SAS 9.2 Foundation for z/OS installation procedure, the SASINew or SASIHOLD job generated the batch jobs to create and mount an HFS data set. The JCL to create the HFS data set is &CNTLDSN(HFSCREAT). The JCL was tailored based on the &prefix high level qualifier and parameters used for allocating traditional data sets for SAS 9.2 Foundation.

System personnel might use this job as an explanation of the work required and accomplish the tasks with local means, or they might verify or adjust the JCL and parameters and submit the job. The *name* of the HFS data set is not critical and should be modified to suit your local needs. The HFSMOUNT utility in step 4 will need to be modified to match the chosen HFS data set name.

If the traditional data sets were allocated using VOLSER allocation, then all the SMS parameters will have to be provided in HFSCREAT. If SMS was used to allocate traditional data sets, then those parms will be present in the generated HFSCREAT. Verify that the SMS parms used in HFSCREAT are proper for the HFS data set.

An excellent reference for further details on creating and mounting an HFS data set is the chapter “Managing the Hierarchical File System” in *UNIX System Services Planning*. For information on using a zSeries file system (zFS), consult *Distributed File Services zFS Administration*.

STEP 4: Mount the data set to enable UNIX file system space for the SAS components (manually or run HFSMOUNT utility).

In this step you mount the z/OS data set that contains the file system based components of SAS 9.2 Foundation for z/OS to the appropriate mount point in the file system hierarchy. The example uses HFS. This job should be run by the z/OS UNIX System Services Administrator (sysadmin) who has root user and group authority (uid=0 and gid=0) and SAF create access to the data set.

As part of SAS 9.2 Foundation for z/OS installation procedure, the SASINew or SASIHOLD job generated the batch jobs to create and mount an HFS data set. The JCL to mount the HFS data set is &CNTLDSN(HFSMOUNT). The JCL was tailored based on the &prefix high level qualifier and parameters used for allocating traditional data sets for SAS 9.2 Foundation.

System personnel might use this job as explanation of the work required and accomplish the task with local means, or they might verify or adjust the JCL and parameters and submit the job.

Note: *If the name of the HFS data set in the HFSCREAT utility was modified in step 3, it will need to be modified to match in HFSMOUNT.*

Use of the default directory structure is strongly recommended. If the job is modified, it is important to retain the default mount point to match the directory path assumed by the &CNTLDSN(USSUNTAR) utility. This will ensure that later installations can find, analyze, and add to this structure without requiring significant manual editing of program language to match non-default directory structures known only to the original installer.

Note that a security problem can occur with mounting if your site has defined several superuser IDs and they have different GIDs. Caching of user information by the security package (for example, RACF) can cause retrieval of the wrong GID for a given UID. IBM recommends that you do not define different user IDs with the same UID.

An excellent reference for further details on mounting an HFS is the chapter “Managing the Hierarchical File System” in *UNIX System Services Planning*. If you want to instead use a zSeries file system (zFS), consult *Distributed File Services zFS Administration*.

STEP 5: Run USSUNTAR utility to install the SAS components into the UNIX file system.

The SAS 9.2 Foundation for z/OS file system components are shipped in a ‘tar file.’ This file contains components written by the UNIX System Services tar (tape archive) utility. This utility, as well as the pax utility, can read the contents of a tar file and install them into a file system. For a description of the tar and pax utilities, refer to their manpages (man tar or man pax from a USS shell prompt) or to the chapter “Shell Command Descriptions” in *UNIX System Services Command*

Reference. For details about the contents of tar files, consult the “File Formats” appendix of the *UNIX System Services Command Reference*.

The SAS 9.2 Foundation for z/OS installation created an MVS partitioned data set (PDS) containing the tar file members in `< USER.PREFIX >.TARFILES`. The `&CNTLDSN(USSUNTAR)` job contains shell scripts and invocations of the pax utility which have been customized to the installed product mix, so that it is unnecessary to separately download the tar file to the HFS or zFS. All that is needed is to run USSUNTAR.

If an installed product has additional functionality enabled by HFS content, the USSUNTAR job will already contain the needed unload and untar language. The content of individual tar files is not provided, and although this work could be done “by hand,” use of USSUNTAR guarantees that all the content for the customer’s products is transferred and is in the default (expected) directories so that later customization is simplified or eliminated. ***Use of the default directory structure is strongly recommended.***

Important: *If the UNIX System Services root directory for your SAS installation cannot use the default directory specified in the `INSTALL_DIR=` line, edit the `INSTALL_DIR=` line in the USSUNTAR member of the installation control dataset AND in any installation control dataset members beginning with “USUT” IF such members exist. For some customers, one or more USUT* members may be placeholder jobs only and not contain an `INSTALL_DIR=` line. This is not an error and such members can be ignored.*

The components stored in the tar files include directories and files. Component attributes such as the owner and permissions are also stored in the tar file. In extracting the components, you will want to preserve the permission bits but reset the owner to the SAS administrator at your site. To accomplish this, the `&CNTLDSN(USSUNTAR)` job should be submitted for execution by the SAS administrator whose user ID was the user of record in the allocation of the traditional z/OS data sets in the main install.

STEP 6: Update system environment variables to access the SAS components.

Once the file system is allocated and its ownership and permissions set, update system environment variables to provide access to the SAS program library, the SAS Java native method DLL library, and SAS Java class libraries. Access the SAS program library through an update to the STEPLIB environment variable, SAS Java native method DLL's and the SAS Java proxy through an update to the LIBPATH environment variable, and SAS Java class libraries through an update to the CLASSPATH variable. You can update `/etc/profile` to make these changes available to all users, or update the `.profile` file in the home directory for specific users requiring access to the file system based functionality. For details on customizing the UNIX System Services environment, see the section “Customizing the UNIX Shells” in the chapter “Customizing the Shells and Utilities” of *UNIX System Services Planning*.

For example:

```
export STEPLIB="$STEPLIB:&prefix.LIBRARY"

export
LIBPATH="$LIBPATH:$JAVA_HOME/bin:$JAVA_HOME/bin/classic:/usr/lpp/SAS/SAS_9.2
/USER.PREFIX/bin"

export
CLASSPATH="$JAVA_HOME/lib:/usr/lpp/SAS/SAS_9.2/USER.PREFIX/hostcm"
```

STEP 7: Assign a permanent mount point for the file system in BPXPRMxx of 'SYS1.PARMLIB'.

Once the file system is allocated and its ownership and permissions set, update the UNIX Systems Services configuration file BPXPRMxx in 'SYS1.PARMLIB' to specify a permanent mount point for the file system components of SAS 9.2 Foundation for z/OS. This will ensure that the components are always available. You might want to mount the file system in read-only mode. This will prevent accidental corruption of the files. In a SYSPLEX environment, this will also improve performance, as cross system communication to maintain the integrity of the file system is not necessary.

Use the MOUNT statement in BPXPRMxx to permanently mount the file system. The xx suffix is set by the OMVS=xx specification in IEASYSxx or by operator reply to the system startup IPL message using the OMVS=xx parameter. For details about the syntax of the MOUNT statement, consult the chapter "BPXPRMxx (z/OS UNIX System Services Parameters" of the *MVS Initialization and Tuning Reference*.

```
MOUNT FILESYSTEM( '&prefix.SAS.VERS9.HFS' )
      TYPE(HFS)
      MOUNTPOINT( '/usr/lpp/SAS/SAS_9.2' )
      MODE(READ)
      NOSETUID
```

Configuring SAS Software for Use with the Java Platform

Java is a programming environment or platform first developed by Sun Microsystems, Inc. You can find background information at Sun's Java Web site: <http://java.sun.com>. IBM and other vendors provide their own implementations of Java for use by their customers.

A small number of features require the use of IBM's Java Software Developer's Kit (SDK). For backward compatibility, no previously existing functionality in SAS 9.2 Foundation requires an SDK. If you do not want to install an SDK, SAS will run, but you will not be able to take advantage of the features that require an SDK. If you decide at a later time that you want to use the features, you can install an appropriate SDK and configure SAS 9.2 Foundation as described in the steps below.

Note: JAVA, JAVAIMG, ACTIVEX, and JAVAMETA all require the SDK (JRE on most systems), although they are **not** new to SAS 9.2 Foundation. Usage of these device drivers is optional, but the JAVAIMG device driver is used by default for graphs created by stored processes on non-Windows systems. If your configuration is part of a BI installation, you should configure for Java, or tell all creators of stored processes to specify a different default device driver such as PNG or GIF.

The following features in SAS 9.2 Foundation require an SDK:

- SAS/GRAPH support for the javaimg device
- SAS REPORT procedure

Complete the following steps to configure SAS for use with the Java Platform:

STEP 1: Determine the version or release of the SDK that you need.

Visit the SAS “Third Party Software Downloads” Web page at <http://support.sas.com/thirdpartysupport> to determine the minimum version or release of the SDK required by SAS 9.2 Foundation. This Web page contains the most accurate and up-to-date information about Java software downloads needed by SAS.

If you already have an SDK installed on your system, identify the version or release of the SDK and verify that it meets the requirements shown on the “Third Party Software Downloads” page.

STEP 2: Install and configure the SDK.

If you have not installed a version or release of an SDK that is specified on the “Third Party Software Downloads” Web page (<http://support.sas.com/resources/thirdpartysupport/>) you need to install and configure an SDK now. Click the appropriate **Download** button on the “Third Party Software Downloads” page and be sure to follow the instructions on the IBM download site.

For more information about IBM’s implementation of Java you can visit <http://www-1.ibm.com/servers/eserver/zseries/software/java>, and click **Links**.

STEP 3: Set TKMVSENV options.

&prefix.TKMVSENV is a fixed block data set that is allocated at install time, where &prefix is the High Level-Qualifier of SAS 9.2 Foundation installation libraries. Its purpose is to contain options that SAS uses to run Java. Before you attempt to use SAS features requiring Java, you must ensure that entries from the following list are provided in the TKMVSENV member located in the &prefix.TKMVSENV data set. Those entries must set the options to appropriate values for your SAS installation.

Options SAS Uses to Start Java

TKMVSENV Command	Status	Meaning
set TKJNI_OPT_LIBPATH=	Required	Specifies the paths of Java shared libraries that SAS uses on z/OS. The most important of these libraries is libjvm.so. libjvm.so uses other libraries, including libjtc.so. Colons delimit the paths in the specification. SAS uses this option like a LIBPATH environment variable in UNIX System Services.
set TKJNI_OPT_PROXYPATH=	Required	Specifies a path that provides an external link to the JPROXY load module. See Step 5 for more information about external links.
set TKJNI_OPT_DISPLAY=	Required	Specifies the name of an X server or X server pool. SAS uses this option like a DISPLAY environment variable in UNIX System Services.
set TKJNI_OPT_LEOPTS=	Required	Specifies that XPLink linkage will be used to call the JVM. This option <u>must</u> specify 'XPLINK(ON)', as shown in the examples below.

For example, the TKMVSENV member might look like the following:

Note: The colon (:) is a delimiter on these lines.

```
Reset                Clear all values
set TKJNI_OPT_LIBPATH=/java/bin/classic:/java/bin
set TKJNI_OPT_PROXYPATH=/usr/lpp/SAS/SAS_9.2/TEST.SAS.V9W0/jproxy
set TKJNI_OPT_DISPLAY='your.x.server:0'
set TKJNI_OPT_LEOPTS='XPLINK (ON)'
```

The TEST.SAS.V9W0 in this example is your install *user.prefix* value.

STEP 4: Customize SAS 9.2 Foundation options.

SAS 9.2 Foundation option JREOPTIONS is used to pass Java options when SAS starts the Java Virtual Machine. The following options are required to configure Java to run SAS applications. They specify the location of the SAS jar files as well as information about how SAS's Java classes should be loaded and executed.

Options	Status	Description
-Dsas.app.class.dirs	Required	Specifies the path where SAS jar files are located.
-Djava.security.policy	Required	Specifies the location of the sas.policy file.

Options	Status	Description
-Djava.class.path	Required	Specifies the location of the sas.launcher.jar file.
-Djava.system.class.loader	Required	Specifies "com.sas.app.AppClassLoader." This option is always the same, regardless of where SAS is installed.
-Dsas.ext.config	Required	Specifies the location of the sas.java.ext.config file.
-DPFS_TEMPLATE	Required	Specifies the location of the qrpfstpt.xml file.

Edit the JREOPTIONS line in the BATnn and TSOnn config files in the &prefix.CNTL data set. Add all of the required options listed above and, where appropriate, assign values to them that reflect the location of the required files in your SAS installation. Any content beyond column 72 must begin in column 1 of the next line, and the method continued for as many lines as needed.

For example the JREOPTIONS= value may look like the following:

```
JREOPTIONS=(-Djava.security.policy=/usr/lpp/SAS/SAS_9.2/TEST.SAS.V9W0/MVS/s
as.policy -Djava.class.path=/usr/lpp/SAS/SAS_9.2/TEST.SAS.V9W0/hostcm/sas.launc
her.jar -Djava.system.class.loader=com.sas.app.AppClassLoader -Dsas.app.
class.dirs=/usr/lpp/SAS/SAS_9.2/TEST.SAS.V9W0/avdobj/:/usr/lpp/SAS/SAS_9
.1/TEST.SAS.V9W0/hostcm/ -Dsas.ext.config=/usr/lpp/SAS/SAS_9.2/TEST.SAS.
V9W0/MVS/sas.java.ext.config -DPFS_TEMPLATE=/usr/lpp/SAS/SAS_9.2/TEST.SAS.V9W
0/hostcm/qrpfstpt.xml)
```

where JREOPTIONS starts in column 2, the first six lines end in column 72, the second through seventh lines start in column one, and the options are enclosed in parentheses. Note that TEST.SAS.V9W0 in this example is your install *user.prefix* value.

STEP 5: Set up the external link for JPROXY.

External links are a special type of symbolic link described in the *z/OS V1R9.0 UNIX System Services User's Guide (SA22-7803-10)*. The TKJNI_OPT_PROXYPATH option (described in Step 3) is used to provide the location of an external link that gives the member name of SAS's JPROXY load module. This external link must be in place so that z/OS can perform a library search to find JPROXY in SAS's load libraries when SAS starts Java.

You can use the following UNIX System Services commands to check if the external link to JPROXY is in place. If, for example, you set the value of TKJNI_OPT_PROXYPATH to `/usr/lpp/SAS/SAS_9.2/user.prefix/myproxy`, you can check for JPROXY's external link by entering:

```
cd /usr/lpp/SAS/SAS_9.2/user.prefix
ls -la
```

You should see an entry like this:

```
myjproxy -> JPROXY
```

If there is no entry for the external link, you need to create one. You can do so by executing the following command from the directory that TKJNI_OPT_PROXYPATH specifies:

```
ln -e JPROXY myjproxy
```

After entering this command you should see the external link entry for JPROXY in the directory TKJNI_OPT_PROXYPATH specifies.

STEP 6: Set the Region Size.

The region size of a typical SAS job needs to be increased when features are used that incorporate Java. The increase will vary depending upon the application. At a minimum, regions for SAS jobs using Java require 256 MB. For a batch job, add either `REGION=256M` or `REGION=262144K` to the JOB card. For a TSO session, specify `SIZE(262144)`.

Locating Hot Fixes

Occasionally, SAS might find it necessary to supply "hot fixes" in order to quickly correct reported problems in a SAS installation. A hot fix is a fix, along with installation instructions and an audit file, used to repair a specific reported problem, or problems, built on specific host(s), tested for accuracy, and immediately delivered via the World Wide Web. An audit file enables Tech Support consultants to determine which hot fixes have been installed. Hot fixes are located at <http://ftp.sas.com/techsup/download/hotfix/hotfix.html> and organized in several ways: by host, by release, and by product.

When hot fixes are released, the information regarding the hosts involved, the products involved, and/or the problems resolved by the hot fix are posted to TSNEWS-L, the SAS Technical Support listserv (information about subscribing to this listserv is available on the SAS Order Information Letter in your installation kit). Once you receive an email from TSNEWS-L that you think might affect your SAS installation, go to the above Web site and click on the "Latest Hot Fixes" link to find more specific information before downloading and installing the appropriate fixes.

Hot fixes address software issues uncovered by customer applications. For some customers, a hot fix provides a quick solution to an immediate problem. For others, it is system maintenance. Hot fixes have been tested and are fully supported.

Chapter 2 — Post-Installation Configuration for National Language Support (NLS)

This chapter contains information on post-installation configuration for Asian and European language support.

Chinese, Japanese, and Korean DBCS Support

This section provides information about Asian font catalogs. It also describes the three SAS 9.2 Foundation options you should use to set the DBCS encoding for Asian DBCS character sets.

- DBCS
- DBCSLANG
- DBCSTYPE

DBCS System Option

The DBCS system option indicates that all text, input, output, and data should be processed as if it is encoded in a double-byte character set. This option is used to process Asian DBCS languages such as Chinese (both Simplified and Traditional), Japanese, and Korean.

- DBCS is a Boolean option whose values are either DBCS or NODBCS.
- DBCS (specifies that SAS 9.2 Foundation process double-byte character sets)
- NODBCS (specifies that SAS 9.2 Foundation not process double-byte character sets)

The default value set by SAS 9.2 Foundation is NODBCS.

DBCSLANG System Option

The DBCSLANG= system option specifies which double-byte character set (DBCS) is in use.

Note: This option does not accept abbreviations for the language value.

The following table provides valid DBCSLANG values for each character set:

Character Set	DBCSLANG Value
Simplified Chinese	CHINESE
Japanese	JAPANESE
Korean	KOREAN
Traditional Chinese	TAIWANESE

When DBCS extensions are in effect, Japanese is the default value.

Note: All values of DBCSLANG require that you specify a value of IBM for the DBCSTYPE system option.

DBCSTYPE System Option

The DBCSTYPE= system option specifies the type of double-byte character set (DBCS) encoding method. The only valid value for DBCSTYPE is IBM (which specifies the IBM encoding method).

Asian Font Catalogs

With the exception of Traditional Chinese fonts, Asian fonts reside in the SASHELP.FONTS catalog. (Note, though, that the configuration file for DBCS extensions does **NOT** contain font definitions.) To use Traditional Chinese fonts, you must specify them in your SAS session.

Specifying the Font Catalog in a SAS Session for Traditional Chinese Fonts

To specify the font catalog in a SAS session, submit the following LIBNAME statement:

```
libname gfontx '<high-level-qualifier>.zt.<font>.GFONT' ;
```

In this statement

- *x* represents a value from 0-9
- *high-level-qualifier* is a user-supplied value
- *font* is the name of the font

European Language Support

The following sections describe different methods for configuring your system for locale, the use of NONLSCOMPATMODE versus NLSCOMPATMODE, and how to set up your local session to transfer data to a remote session. The last section also provides a list of devmap and keymap values that match the locales on your operating system.

z/OS media is available in the following 12 encoded versions, which support multiple locales and regions. (See the table below for the locale and region values).

- 838 (Thai)
- 870 (Central Europe)
- 1025 (Russia)
- 1047 (United States) – the default
- 1141 (Austria and Germany)
- 1142 (Denmark and Norway)
- 1143/1122 (Finland and Sweden)
- 1144 (Italy)
- 1145 (Spain)
- 1146 (United Kingdom)
- 1147 (France)
- 1148/1130 (International)

When you install SAS 9.2 Foundation on z/OS for a given encoding, the installation-generated configuration files set the LOCALE system option to the default value for the encoding installed.

A given SAS 9.2 Foundation installation supports one encoding, and sometimes two encodings, along with a specific list of locales that are compatible with those encodings. Specification of any locale that does not appear in the installed configuration file is unsupported, and it will produce unpredictable results.

To verify the EBCDIC code page for the media you received, refer to the following table, which maps countries to appropriate encodings.

Shipping Map for Countries Based on Encoded Media

Country	Code Page Number	Required Media and Data Set Code	Supported Locales
Austria	cp1141	W3	German_Austria
Belgium	cp1148	WB	Dutch_Belgium; French_Belgium
Croatia	cp870	C0	Croatian_Croatia
Czech Republic	cp870	C0	Czech_CzechRepublic
Denmark	cp1142	W5	Danish_Denmark
Finland	cp1143/cp1122	W6	Finnish_Finland
France	cp1147	WA	French_France
Germany	cp1141	W3	German_Germany
Hungary	cp870	C0	Hungarian_Hungary
Italy	cp1144	W7	Italian_Italy
North and South America	cp1047	W0	English_Canada; French_Canada; English_UnitedStates; Portuguese_Brazil; Spanish_Argentina; Spanish_Bolivia; Spanish_Chile; Spanish_Colombia; Spanish_Ecuador; Spanish_ElSalvador; Spanish_Mexico; Spanish_Nicaragua; Spanish_Panama; Spanish_Paraguay; Spanish_Peru; Spanish_PuertoRico; Spanish_UnitedStates; Spanish_Uruguay; Spanish_Venezuela
Norway	cp1142	W5	Norwegian_Norway
Poland	cp870	C0	Polish_Poland
Portugal	cp1047	W0	Portuguese_Portugal
Romania	cp870	C0	Romanian_Romania
Russia	cp1025	R0	Russian_Russia
Slovakia	cp870	C0	Slovak_Slovakia
Slovenia	cp870	C0	Slovenian_Slovenia
Spain	cp1145	W8	Spanish_Spain
Sweden	cp1143/cp1122	W6	Swedish_Sweden
Switzerland	cp1148/cp1130	WB	French_Switzerland German_Switzerland Italian_Switzerland
Thailand	cp838	F0	Thai_Thailand

Country	Code Page Number	Required Media and Data Set Code	Supported Locales
United Kingdom	cp1146	W9	English_UnitedKingdom
United States	cp1047	W0	English_UnitedStates
Vietnam	cp1148/cp1130	WB	Vietnamese_Vietnam

Note: SASHELP has been built specifically for your encoding. Therefore, valid values for the LOCALE system option are limited to those values that are consistent with the encoding.

If you do plan to select a locale other than the default, you might also benefit from the section “Additional Information,” which appears later in this chapter. If you will be running SAS as a server on your platform serving a SAS client on an EBCDIC platform, refer to the section “Locale Setup on the Remote Server,” which appears later in this chapter. Following that section, SAS/GRAPH users will find instructions for setting up the correct devmaps and keymaps in the section “Devmaps and Keymaps for SAS/GRAPH Software.”

Using NONLSCOMPATMODE Versus NLSCOMPATMODE

The default for the NLSCOMPATMODE option has changed to NONLSCOMPATMODE. The NONLSCOMPATMODE option specifies that data is processed in the session encoding, including reading and writing external files as well as processing SAS syntax and user data. The session encoding is the encoding set in the ENCODING= system option.

On z/OS, some existing programs that ran in previous releases of SAS will no longer run when NONLSCOMPATMODE is in effect. If you have made character substitutions in SAS syntax, you will have to modify your programs to use national characters. For example, a Danish customer who has substituted the Å character for the ‘\$’ character in existing SAS syntax will have to update the program to use the ‘\$’ in the Danish environment.

Setting the NLSCOMPATMODE system option provides compatibility with previous releases of SAS. Programs that were run in previous releases of SAS will continue to work when NLSCOMPATMODE is specified.

An additional change you will find with NONLSCOMPATMODE is that the appropriate Open Edition version of the EBCDIC encoding will be used as the session encoding, which uses the new-line character as the end-of-line character. NLSCOMPATMODE used the traditional EBCDIC encodings. See the section below titled “New-Line Character and Line-Feed Character” for more information about traditional and Open Edition EBCDIC encodings.

For more information about the NLSCOMPATMODE and NONLSCOMPATMODE system options, refer to the system option documentation for your operating environment.

New-Line Character and Line-Feed Character

Both the line-feed character and the new-line character appear in EBCDIC encodings, but only the line-feed character appears in ASCII encodings. Much of SAS 9.2 Foundation uses the new-line character in EBCDIC to indicate the end of the line.

Because ASCII encodings do not support the new-line character, software running on ASCII platforms always expects the line-feed character to indicate the end of the line. When data is transferred from z/OS to a machine that supports ASCII encodings, formatting problems can occur (particularly in HTML output) because the EBCDIC line-feed is not written in the data stream as an end-of-line character.

SAS supports two sets of EBCDIC-based encodings for z/OS. The encodings that have EBCDIC in their names use the traditional mapping of the EBCDIC line-feed character to the ASCII line-feed character. This mapping can cause data to appear as one stream when the data is transferred to an ASCII platform.

The encodings that use Open Edition in their names use the line-feed character as the end-of-line character. When the data are transferred to an ASCII platform, the EBCDIC new-line character maps to an ASCII line-feed character. This mapping enables ASCII applications to interpret the end-of-line correctly, resulting in better formatting.

Configuring Your System for Locale

If you want to configure your SAS session for a locale other than the default locale, you have a couple of methods you can use to achieve that goal. This section explains those methods.

Changing the Default LOCALE Option Setting

When you install SAS 9.2 Foundation and you choose to load NLS language translations, the installation automatically sets the LOCALE system option to the default value for the language installed. The LOCALE option is set in the system configuration file for each language installed.

For example, CNTL(TSOWA) sets LOCALE to French by default.

Note: *The English_UnitedStates version sets LOCALE by default.*

If you want to change the default locale setting for SAS, you can set the LOCALE system option to the appropriate language in your system configuration file.

For example, you can edit CNTL(TSOWA) and change `locale=French_France` to `locale=French_Canada`. To change the locale in the file, place an asterisk (*) in front of the comment line for `locale=French_France` to comment it out. Then, to use the value `French_France`, remove the asterisk in front of the line for `locale=French_France`.

Running SAS in a Different Locale

To set the locale for SAS 9.2 Foundation at your site, add the LOCALE system option to your configuration file. You can find a list of locale values in the *SAS 9.2 National Language Support (NLS) User's Guide*.

When you read or write a file, SAS 9.2 Foundation expects the data in the external files to be in the session encoding. To specify a different encoding, refer to the documentation for the ENCODING system option in the FILENAME, INFILE, or FILE statement in the *SAS 9.2 National Language Support (NLS) User's Guide*.

When LOCALE is set, the ENCODING system option will be set to an encoding that supports the language for the locale. SAS 9.2 Foundation expects user data to be in the encoding that matches

the ENCODING option. If you prefer an encoding other than the most common encoding for the locale, you can also set the ENCODING system option in the configuration file.

If you are running on an EBCDIC platform, the encoding will be an Open Edition encoding rather than the corresponding EBCDIC encoding. EBCDIC and Open Edition encodings are based on the same encoding. However, EBCDIC encodings use a different new-line character. Refer to the earlier section “New-Line Character and Line-Feed Character.”

The encoding set by the ENCODING system option will also be used by applications that create output in or that establish communications with applications whose syntax and protocols are not determined by SAS. For example, when ODS generates HTML, RTF, or JavaScript, the output will use, by default, the encoding set by the ENCODING system option. If you want your output to be created using a different encoding, refer to the documentation for the Output Delivery System (ODS).

When the ENCODING option is set, the TRANTAB option will always be set to match the ENCODING system option. The transport format trantabs (translation tables), set by the TRANTAB option, are used by the CPORT and CIMPORT procedures to transfer SAS data files. These trantabs are also used by the UPLOAD and DOWNLOAD procedures for transferring files and catalogs, remotely submitting code to the server, and returning logs and listings to the client. The Output Delivery System (ODS) creates output using the encoding that matches the ENCODING system option. If you want your output to be created using a different encoding, please refer to the documentation for the Output Delivery System.

For more information, refer to the *Base SAS 9.2 Procedures Guide* in base SAS for documentation about PROC CPORT and PROC CIMPORT. Refer to the *SAS/CONNECT 9.2 User's Guide* for documentation about PROC UPLOAD and PROC DOWNLOAD.

Additional Information

Depending on the applications you run, additional setup might be required for your system. Refer to the following sections for more information about configuring your system to run with alternate locales.

Locale Setup on the Remote Server

Note: *The %LS() macro is new in SAS 9.2 Foundation. This macro replaces the functionality of the Locale Setup Window that was used in previous releases. References to "SAS 9" in the following section refer to all releases within SAS 9.*

If you are running SAS 9 as both your client and server sessions, it is not usually necessary to run the %LS() macro to do any further locale setup. The locale of a server should be compatible with the locale of your client session; otherwise, your data might be corrupted.

If your SAS 9 client is connecting to a session running a release of SAS prior to SAS 9, you can use the %LS() macro to set up the remote SAS environment for data transfer. As the Locale Setup Window did in previous releases, the %LS() macro copies the host-to-host translation tables from the LOCALE catalog into SASUSER.PROFILE. The %LS() macro does not set the encoding for the SAS session.

If you use SAS/CONNECT to connect to a remote SAS server, you will need to set up the server session for the locale that the SAS client is using. You must set up the server *after* signing on to the remote session from the client.

The following examples show how to set locale for remote connections:

- **Connecting SAS 9-to-SAS 9:** Use the LOCALE option at startup. The LOCALE option value of the SAS client and server sessions should be the same. For example,

```
sas o('locale=Danish_Denmark')
```

- Connecting SAS 9 and a previous release of SAS:

SAS 9 receives the data: Use the LOCALE option on the Version 9 side at start up.

Example:

```
sas o('locale=Spanish_Spain')
```

Previous release receives the data: Start SAS 9 with the LOCALE option at start up.

Example:

```
sas o('locale=Spanish_Mexico')
```

Then use the %LS () macro in SAS 9 to set up the host-to-host translation tables on the previous release after connection is established.

Example: Submit the following code from the Program Editor:

```
%ls (locale=Spanish_Mexico, remote=on);
```

Devmaps and Keymaps for SAS/GRAPH Software

If you are running SAS/GRAPH software and you want to display non-ASCII characters, you will need to set the appropriate devmaps and keymaps to match your current encoding. The devmap and keymap entries are located in

the SASHELP.FONTS catalog. To get the correct devmaps and keymaps for your encoding, you should use the %LSGRAPH macro. %LSGRAPH automatically sets up your environment for you by

- copying the devmap and keymap entries that match your encoding to the GFONT0.FONTS catalog.
- changing the name of the entry to the name DEFAULT so the devmaps and keymaps will be loaded for you.

The following example uses %LSGRAPH to set the correct devmap and keymap (E142) for a Polish user on the z/OS operating system:

```
libname gfont0 'your-font-library';
%lsgraph(e142);
```

The following tables—one for running NLSCOMPATMODE; the other for running NONNLSCOMPATMODE—list devmaps and keymaps that match the locales on your platform when you are running in NLSCOMPATMODE or in NONNLSCOMPATMODE.

Devmaps and Keymaps for NLSCOMPATMODE

Locale	Devmap and Keymap Name
Arabic_Algeria	e425
Arabic_Bahrain	e425
Arabic_Egypt	e425
Arabic_Jordan	e425
Arabic_Kuwait	e425
Arabic_Lebanon	e425
Arabic_Morocco	e425
Arabic_Oman	e425
Arabic_Qatar	e425
Arabic_SaudiArabia	e425
Arabic_Tunisia	e425
Arabic_UnitedArabEmirates	e425
Bulgarian_Bulgaria	ecyr
Byelorussian_Belarus	ecyr
Croatian_Croatia	e870
Czech_CzechRepublic	e870
Danish_Denmark	e142
Dutch_Belgium	elat
Dutch_Netherlands	elat
English_Australia	elat
English_Canada	elat
English_HongKong	e146
English_India	e146
English_Ireland	e146
English_Jamaica	elat
English_NewZealand	elat
English_Singapore	e146
English_SouthAfrica	e1at
English_UnitedKingdom	e146
English_UnitedStates	e1at
Estonian_Estonia	eest

Locale	Devmap and Keymap Name
Finnish_Finland	e143
French_Belgium	e148
French_Canada	elat
French_France	e147
French_Luxembourg	e147
French_Switzerland	e148
German_Austria	e141
German_Germany	e141
German_Liechtenstein	e141
German_Luxembourg	e141
German_Switzerland	e148
Greek_Greece	e875
Hebrew_Israel	e424
Hungarian_Hungary	e870
Icelandic_Iceland	elat
Italian_Italy	e144
Italian_Switzerland	e148
Latvian_Latvia	ebal
Lithuanian_Lithuania	ebal
Norwegian_Norway	e142
Polish_Poland	e870
Portuguese_Brazil	e275
Portuguese_Portugal	elat
Romanian_Romania	e870
Russian_Russia	ecyr
Serbian_Yugoslavia	ecyr
Slovak_Slovakia	e870
Slovenian_Slovenia	e870
Spanish_Argentina	elat
Spanish_Bolivia	elat
Spanish_Chile	elat

Locale	Devmap and Keymap Name
Spanish_Colombia	elat
Spanish_CostaRica	elat
Spanish_DominicanRepublic	elat
Spanish_Ecuador	elat
Spanish_ElSalvador	elat
Spanish_Guatemala	elat
Spanish_Honduras	elat
Spanish_Mexico	elat
Spanish_Nicaragua	elat
Spanish_Panama	elat
Spanish_Paraguay	elat
Spanish_Peru	elat
Spanish_PuertoRico	elat
Spanish_Spain	e145
Spanish_UnitedStates	elat
Spanish_Urugay	elat
Spanish_Venezuela	elat
Swedish_Sweden	e143
Turkish_Turkey	etur
Ukrainian_Ukraine	Ecyr

Devmaps and Keymaps for NONLSCOMPATMODE

Locale	Devmap and Keymap Name
Arabic_Algeria	eo2
Arabic_Bahrain	eo2
Arabic_Egypt	eo2
Arabic_Jordan	eo2
Arabic_Kuwait	eo2
Arabic_Lebanon	eo2
Arabic_Morocco	eo2
Arabic_Oman	eo2
Arabic_Qatar	eo2
Arabic_SaudiArabia	eo2
Arabic_Tunisia	eo2
Arabic_UnitedArabEmirates	eo2
Bulgarian_Bulgaria	eocy
Byelorussian_Belarus	eocy
Croatian_Croatia	eol2
Czech_CzechRepublic	eol2
Danish_Denmark	eo42
Dutch_Belgium	eol1
Dutch_Netherlands	eol1
English_Australia	eol1
English_Canada	eol1
English_HongKong	eo46
English_India	eo46
English_Ireland	eo46
English_Jamaica	eol1
English_NewZealand	eol1
English_Singapore	eo46
English_SouthAfrica	eol1
English_UnitedKingdom	eo46
English_UnitedStates	eol1

Locale	Devmap and Keymap Name
Estonian_Estonia	eoet
Finnish_Finland	eo43
French_Belgium	eo48
French_Canada	eol1
French_France	eo47
French_Luxembourg	eo47
French_Switzerland	eo48
German_Austria	eo41
German_Germany	eo41
German_Liechtenstein	eo41
German_Luxembourg	eo41
German_Switzerland	eo48
Greek_Greece	eoel
Hebrew_Israel	eoiw
Hungarian_Hungary	eol2
Icelandic_Iceland	eol1
Italian_Italy	eo44
Italian_Switzerland	eo48
Latvian_Latvia	eobl
Lithuanian_Lithuania	eobl
Norwegian_Norway	eo42
Polish_Poland	eol2
Portuguese_Brazil	eobr
Portuguese_Portugal	eol1
Romanian_Romania	eol2
Russian_Russia	eocy
Serbian_Yugoslavia	eocy
Slovak_Slovakia	eol2
Slovenian_Slovenia	eol2
Spanish_Argentina	eol1

Locale	Devmap and Keymap Name
Spanish_Bolivia	eol1
Spanish_Chile	eol1
Spanish_Columbia	eol1
Spanish_CostaRica	eol1
Spanish_DominicanRepublic	eol1
Spanish_Ecuador	eol1
Spanish_ElSalvador	eol1
Spanish_Guatemala	eol1
Spanish_Honduras	eol1
Spanish_Mexico	eol1
Spanish_Nicaragua	eol1
Spanish_Panama	eol1
Spanish_Paraguay	eol1
Spanish_Peru	eol1
Spanish_PuertoRico	eol1
Spanish_Spain	eo45
Spanish_UnitedStates	eol1
Spanish_Urugay	eol1
Spanish_Venezuela	eol1
Swedish_Sweden	eo43
Turkish_Turkey	eotr
Ukrainian_Ukraine	eocy

Appendix A — Implementing the SAS/ACCESS Interface to ADABAS

Customizing the SAS CLIST, REXX exec, and Cataloged Procedure

Note: This task is required.

If the ADABAS interface is used extensively at your site, and if your ADABAS system load library is not a link list library, you might want to modify the SAS CLIST and cataloged procedure to allocate the necessary files by default. Alternatively, users must specify the `SASLOAD` operand shown below when invoking the CLIST and must override `STEPLIB` when running batch to concatenate the library that contains your site's Software AG load modules. In either case, they might also be required to allocate `DDCARD`, as appropriate.

Note: The concatenation order of the `SAS LOAD LIBRARY` and the load library for database access can be interchanged. Generally, if more database access activities will occur in the SAS session or batch job, place the database load library first in the concatenation (or ahead of the `SAS LOAD LIBRARY`). The opposite is true when SAS processing dominates the session or JOB.

- Make the following changes to the CLIST:
 - Replace the following line in the installation-supplied CLIST:
`SASLOAD(' '&prefix.LIBRARY' '')`
with the line
`SASLOAD(' 'your.ADABAS.loadlib' ' '&prefix.LIBRARY' '')`
 - Optionally, add an appropriate `ALLOCATE` statement for the fileref `DDCARD`, unless appropriate parameters are included in the `ADARUN` module.
- Make the following changes for the REXX exec:
 - If you need to concatenate the ADABAS load library in the SAS tasklib, insert the following option specification into the installation-supplied `SASRXCFG` config file or on the REXX exec command line:
`-DBMSLIBS 'your.ADABAS.loadlib'`
 - If you want the ADABAS load library concatenated before, rather than after, the SAS load library, insert the following option specification into the installation-supplied `SASRXCFG` config file or on the REXX exec command line:
`-DBMSCONCAT FIRST`
- Make the following changes to the cataloged procedure:
 - Modify the `STEPLIB DD` statement to concatenate your Software AG load library as follows:

```
//STEPLIB DD DISP=SHR,DSN=&LOAD  
// DD DISP=SHR,DSN=&prefix.LIBRARY  
// DD DISP=SHR,DSN=your.ADABAS.loadlib
```

- Optionally, add an appropriate DDCARD DD statement, unless appropriate parameters are included in the ADARUN module. The following example illustrates the DDCARD parameters required by this interface:

```
ADARUN DATABASE=001 /*site-specific value*/
ADARUN DEVICE=3380 /*site-specific value*/
ADARUN MODE=MULTI /*multi (default) or single*/
ADARUN SVC=253 /*site-specific value*/
ADARUN PROGRAM=USER /*required*/
```

Using the NATURAL Security Interface

Note: This task is optional.

The modules NSCDDM and NSCDDM22 unloaded to your SAS LIBRARY data set are the NATURAL Security Interface modules supplied by Software AG. If you do not have the NATURAL Security Software, or do not want to use it, delete or rename the load module NSCDDM.

The module NSCDDM is necessary for support of NATURAL Release 2.3.1 or later or ADABAS Version 6 when using the NATURAL Security Interface. If you are using NATURAL Release 2.2 and you want to use the NATURAL Security Interface, delete or rename the existing NSCDDM module and then rename the module NSCDDM22 to NSCDDM.

Using the Samples

Note: This task is optional.

Three samples for the ADABAS interface are unloaded into your &prefix.SAMPLE library at install time:

- ADBDOC, which is used to create sample data sets. These sample data sets are referenced in *SAS/ACCESS Interface to ADABAS Software, Version 8 and later*.
- ADBUTL contains input to the ADABAS Utilities that are used to create the four ADABAS files referenced in *SAS/ACCESS Interface to ADABAS Software, Version 8 and later*. You might want to create NATURAL DDMs to match these files using the NATURAL software.
- ADBSTMT contains line-mode statements used to create sample access and view descriptors. These access and view descriptors are referenced in *SAS/ACCESS Interface to ADABAS Software, Version 8 and later*.

SAS 9.2 Foundation Options for this Interface

Note: This task is optional.

To see a list of SAS 9.2 Foundation options for this interface, invoke SAS 9.2 Foundation and submit the following statements:

```
proc options group = adabas; run;
```

Appendix 1 in the *SAS/ACCESS Interface to ADABAS Software* manual also lists SAS 9.2 Foundation options for this interface. You might want to review these during installation. All options have defaults.

The options applicable to this interface are all invocation options. That means you can change them when you invoke SAS 9.2 Foundation but not during a SAS session. Some system options can be overridden by using the corresponding data set options. You can supply data set options when a data set is referenced in a SAS procedure or DATA step.

You are allowed to restrict changes to invocation options by placing them in the Restricted Options Table, `SASOPTRS`. This process is described in “Customizing Default Options and System Configuration Files” on page 11 of this document.

One option, `ADBUPD=`, is suggested as a restricted option. The `ADBUPD=` option determines whether the SAS/ACCESS Interface to ADABAS engine can perform both read and update operations or is restricted to read-only. The default option value, `Y`, specifies that both reads and updates are allowed. The option value, `N`, specifies that the engine can only read ADABAS data; any attempt to update an ADABAS file results in an error.

Note: The `ADBUPD=` option replaces the `ADBENGMD=` option available in earlier releases.

Other options you might want to examine first are those listed in Appendix 1 of the *SAS/ACCESS Interface to ADABAS Software, Version 8 and later*, under the heading ADABAS System Options.

NATURAL Date and Time Support

The SAS/ACCESS Interface to ADABAS now supports NATURAL Date (D) and Time (T) datatypes. Such fields will have their data values translated into the equivalent SAS dates and times respectively.

When an access descriptor is created on a NATURAL or PREDICT DDM containing a date field, the default SAS format and informat is set to `DATE9`. As with other default formats and informats displayed with the LIST commands, this can be changed to another compatible SAS format/informat.

A NATURAL time field has a default SAS format and informat of `TIME8`. This can be changed to another SAS format/informat.

If the NATURAL time field is an extended time field, this can be interpreted as a SAS datetime value by specifying an `E` in the `DB Content` field. In this case, the default SAS format and informat is changed from `TIME8` to `DATETIME18`. This can be changed to another compatible SAS format/informat.

While it is not possible to display these values as simple numeric values (date or time values), NATURAL date and time values are stored differently than SAS date and time values. Thus the values displayed in this manner will not be the same as if they had been displayed through a NATURAL application. It is not recommended that these values be used as other than their corresponding date or time values.

Reentrancy

Load module `SASIOADB` is non-reentrant because of the non-reentrant program `ADAUSER` provided by Software AG, which must be linked with `SASIOADB` in order to communicate with the ADABAS DBMS.

Appendix B — Implementing the SAS/ACCESS Interface to CA-DATACOM/DB Software

Customizing the SAS CLIST, REXX exec, and Cataloged Procedure

Note: This task is required.

If the CA-DATACOM/DB interface is used extensively at your site, and your DATACOM system load library is not a link list library, you might want to modify the SAS CLIST and cataloged procedure to allocate the necessary files by default. If you do not, users will be required to specify the `SASLOAD` operand shown below when invoking the CLIST, and to override `STEPLIB` when running batch to concatenate the library that contains your site's CA-provided load modules.

Note: The concatenation order of the `SAS LOAD LIBRARY` and the load library for database access can be interchanged. Generally, if more database access activities will occur in the SAS session or batch job, place the database load library first in the concatenation (or ahead of the `SAS LOAD LIBRARY`). The opposite is true when SAS processing dominates the session or JOB.

- Make the following change to the CLIST:

Replace the following line in the installation-supplied CLIST:

```
SASLOAD(' '&prefix.LIBRARY''')
```

with the line:

```
SASLOAD(' 'your.DATACOM.loadlib' ' '&prefix.LIBRARY''')
```

- Make the following changes for the REXX exec:
 - If you need to concatenate the DATACOM load library in the SAS tasklib, insert the following option specification into the installation-supplied SASRXCFG config file or on the REXX exec command line:

```
-DBMSLIBS 'your.DATACOM.loadlib'
```
 - If you want the DATACOM load library concatenated before, rather than after, the SAS load library, insert the following option specification into the installation-supplied SASRXCFG config file or on the REXX exec command line:

```
-DBMSCONCAT FIRST
```
- Make the following change to the cataloged procedure:

Modify the `STEPLIB DD` statement to concatenate your CA-provided load module library as follows:

```
//STEPLIB DD DISP=SHR,DSN=&LOAD  
// DD DISP=SHR,DSN=&prefix.LIBRARY  
// DD DISP=SHR,DSN=your.DATACOM.loadlib
```

Using the Samples

Note: This task is optional.

Three samples for the CA-DATACOM/DB interface are unloaded into your `&prefix.SAMPLE` library at install time:

- `DDBDOC` is used to create sample data sets referred to in *SAS/ACCESS Interface to CA-DATACOM/DB: Reference, Version 8 and later, First Edition*.
- `DDBUTL` contains input to the CA-DATADICIONARY Batch Utility that creates the database referred to in *SAS/ACCESS Interface to CA-DATACOM/DB: Reference, Version 8 and later, First Edition*. `DDBUTL` creates a new CA-DATACOM/DB database with four tables.
- `DDBDESC` contains code to create the access and view descriptors to be used in recreating the examples in *SAS/ACCESS Interface to CA-DATACOM/DB: Reference, Version 8 and later, First Edition*.

SAS 9.2 Foundation Options for this Interface

Note: This task is optional.

To see a list of SAS 9.2 Foundation options for this interface, invoke V9 SAS and submit the following statements:

```
proc options group=datacom; run;
```

You can also refer to Appendix 1 in *SAS/ACCESS Interface to CA-DATACOM/DB: Reference, Version 8 and later, First Edition* for a list of SAS 9.2 Foundation options for this interface. You might want to review this information during installation. All options have defaults.

The options applicable to this interface are invocation options. That means you can change them when you invoke SAS 9.2 Foundation, but not during a SAS session. Some system options have corresponding data set options; these can be set during a SAS procedure or DATA Step. Refer to Appendix 2 in *SAS/ACCESS Interface to CA-DATACOM/DB: Reference, Version 8 and later, First Edition* for information on these data set options.

One invocation option, `DDBMISS=`, specifies a value to be used for representing null values when the SAS/ACCESS Interface engine for CA-DATACOM/DB inserts or updates records in a database table. Valid values for the `DDBMISS=` option are blank (0X40) which is the default, and 0X00.

You are allowed to restrict changes to invocation options by placing them in the Restricted Options Table, `SASOPTRS`. This process is described in “Customizing Default Options and System Configuration Files,” on page 11 of this document.

One option, `DDBUPD=`, is suggested as a restricted option. The `DDBUPD=` system option determines whether the SAS/ACCESS Interface for CA-DATACOM/DB engine can perform both read and update operations, or is restricted to read-only. The default option value, `Y`, specifies that both reads and updates are allowed. The option value, `N`, specifies that the engine can only read CA-DATACOM/DB tables; any attempt to update a CA-DATACOM/DB table results in an error.

Appendix C — Implementing the SAS/ACCESS Interface to CA-IDMS

Customizing the SAS CLIST, REXX exec, and Cataloged Procedure

Note: This task is required.

If the CA-IDMS interface is used extensively at your site, and your IDMS system load library is not a link list library, you might want to modify the SAS CLIST and cataloged procedure to allocate the necessary files by default. Refer to Steps 1, 2 and 3 below for instructions on making these changes.

If you do not make the changes described in Steps 1, 2 and 3, you are required to specify the `LOAD` operand shown below when invoking the CLIST and/or the `LOAD=` parameter when executing the cataloged procedure to concatenate the library that contains your site's CA-IDMS load modules.

If you are accessing the IDMS databases using central version, you might also be required to allocate the `SYSCTL` file. If you are accessing the IDMS databases using local mode, you might also be required to allocate the database files.

- Enter the following command under TSO to start a SAS session in which you will execute the IDMS interface:

```
SASname LOAD("your.IDMS.loadlib")
```

or

```
SASRXname -DBMSLIB 'your.IDMS.loadlib'
```

Where `your.IDMS.loadlib` is the IDMS system load library and the `SASname` is the name of the CLIST or `SASRXname` is the name of the REXX exec you use to invoke SAS 9.2 Foundation.

- To use the IDMS interface with the SAS cataloged procedure, use the `LOAD=` parameter of the PROC to specify your site's load library. For example, you can use the following code in your JCL:

```
//SASIDM EXEC SASname,LOAD='your.IDMS.loadlib'
```

Note: The name the user supplies must consist of all capital letters.

Step 1: Update your SAS CLIST (optional)

If the IDMS interface is used extensively at your site, you might want to alter the CLIST so that your IDMS load library is allocated and concatenated before the SAS load library in the `TASKLIB` symbol. Replace the following line in the installation-supplied CLIST (where `SASLOAD` is the replacement parameter in the CLIST).

- Replace the following line in the installation-supplied CLIST:

```
SASLOAD("&prefix.LIBRARY")
```

with the line:

```
SASLOAD("your.IDMS.loadlib" "&prefix.LIBRARY")
```

- Add an appropriate `ALLOCATE` statement for the fileref `SYSCTL`, if you are accessing your IDMS databases using central version. Add the appropriate `ALLOCATE` statements for the IDMS databases and dictionary you have the authority to access, if you are accessing your IDMS databases using local mode.

Step 2: Update your SAS REXX exec (optional)

- If you need to concatenate the IDMS load library in the SAS tasklib, insert the following option specification into the installation-supplied `SASRXCFG` config file or on the REXX exec command line:


```
-DBMSLIBS 'your.IDMS.loadlib'
```
- If you want the IDMS load library concatenated before, rather than after, the SAS load library, insert the following option specification into the installation-supplied `SASRXCFG` config file or on the REXX exec command line:


```
-DBMSCONCAT FIRST
```
- In the `SASRX` exec, before the "sastrexx" command, add an appropriate `ALLOCATE` command for the fileref `SYSCTL`, if you are accessing your IDMS databases using central version. Add the appropriate `ALLOCATE` commands for the IDMS databases and dictionary you have the authority to access, if you are accessing your IDMS databases using local mode.

Step 3: Update your SAS cataloged procedure (optional)

- If the IDMS interface is used extensively at your site with the SAS cataloged procedure, you might want to make the `STEPLIB DD` statement change permanent to your cataloged procedure. To always execute SAS 9.2 Foundation using the IDMS interface, modify the `STEPLIB DD` statement in your cataloged procedure to reflect the following lines:

```
//STEPLIB DD DISP=SHR,DSN=&LOAD
// DD DISP=SHR,DSN=&prefix.LIBRARY
// DD DISP=SHR,DSN=your.IDMS.loadlib
```

- Add the appropriate `DD` statements for the fileref `SYSCTL`, if you are accessing your IDMS databases using central version.
- Add the appropriate `DD` statements for the IDMS databases and dictionary you have authority to access, if you are accessing your IDMS databases using local mode.

SAS/ACCESS DATA Step Interface Enhancement

The `DATA` step component of the SAS/ACCESS Interface to CA-IDMS is the only component available with this release. This component allows you to directly access network data using special SAS system extensions for the standard SAS `INFILE` statement, along with `DATA` step programming statements. The `INFILE` statement extensions, along with the `DATA` step programming statements, allow you to generate DML calls to the database.

Appendix D — Implementing the SAS/ACCESS Interface to DB2

Defining the Interface to DB2 and DB2 Users

Note: This task is required.

The DB2 database administrator at your site must perform the following steps.

Even if you installed and implemented any previous release of SAS 9.2 Foundation, and SAS/ACCESS Interface to DB2, and completed the `BIND` and `GRANT` steps at that time, you will need to repeat them for this installation.

STEP 1: Bind the DBRMs into an application plan (required).

You can bind the application plan by using the `BIND` option from the DB2I panels under ISPF. Alternatively, you can issue the `BIND` command from TSO through the DSN command processor running in either foreground or background. A new plan will be created. If a plan with this name already exists on your system, it will be replaced.

- The syntax of the `BIND` command is:

```
BIND PLAN(SAS92) ACTION(REPLACE) ISOLATION(CS)
DEGREE(ANY) VALIDATE(RUN) RELEASE(COMMIT) ACQUIRE(USE)
KEEPDYNAMIC(NO) NODEFER(PREPARE) NOREOPT(VAR)
CURRENTDATA(YES) SQLRULES(DB2) DISCONNECT(EXPLICIT)
PKLIST(*.COLLID.SAS92, DB2V7R1.DSNUTILS.*)
```

- If you plan to use DRDA support, you must regenerate plan SAS92 to include the following:

Bind packages for each server you plan to access

Bind for the plan which must include all bind packages

Some statements in this plan that will use a DB2 stored procedure (to invoke the `LOAD UTILITY`), which do not exist on other DBMSs, or in earlier releases of DB2. If you encounter errors regarding unknown SQL statements when binding a package against a particular server, you might want to use the `SQLERROR(CONTINUE)` option to allow the package to be bound without those statements. These statements are not required, since the optional `LOAD UTILITY` will not exist on that DBMS. See the `BIND PACKAGE` statement below for DB2VM as an example.

For some DBMS's, `SQLERROR(CONTINUE)` will still not allow the DBRM to bind. For these DBMS's, there is a second DBRM that can be used which does not include any DB2 z/OS specific SQL statements. This DBRM can be used for the `BIND PACKAGE` on those DBMSs. The name of this DBRM is SAS92X. In order to use this DBRM in the `BIND PACKAGE`, it's name must be changed to SAS92. You can place this DBRM in a different `dbrmlib` than the 'real' SAS92 DBRM, and specify this `dbrmlib` on the `BIND PACKAGE` statements for those DBMS's. This way you can run all of the `BINDs` at one time. Alternately, you can temporarily rename the two DBRMs and run the `BIND PACKAGE` statements individually for the DBMSs that require this DBRM, and then `BIND` the `PLAN` for all of the packages. See the `BIND PACKAGE` statement below for DB2AIX as an example.

- The syntax for binding packages and the plan is as follows:

Note: *These are the suggested option values that have been run and tested. Other values can cause different behaviors. Please consult the DB2 manuals for more information about bind options.*

```

BIND PACKAGE(DB2V7R1.COLLID) OWNER(USERID) QUALIFIER(USERID) MEMBER(SAS92)
ACTION(REPLACE) ISOLATION(CS)
DEGREE(ANY) VALIDATE(RUN) RELEASE(COMMIT) KEEPDDYNAMIC(NO) NODEFER(PREPARE)
NOREOPT(VAR) CURRENTDATA(YES) SQLERROR(NOPACKAGE)
LIBRARY('DB2V7R1.DBRM.DATA')

```

```

BIND PACKAGE(DB2V6R1.COLLID) OWNER(USERID) QUALIFIER(USERID) MEMBER(SAS92)
ACTION(REPLACE) ISOLATION(CS)
DEGREE(ANY) VALIDATE(RUN) RELEASE(COMMIT) KEEPDDYNAMIC(NO) NODEFER(PREPARE)
NOREOPT(VAR) CURRENTDATA(YES) SQLERROR(NOPACKAGE)
LIBRARY('DB2V7R1.DBRM.DATA')

```

```

BIND PACKAGE(DB2VM.COLLID) OWNER(USERID) QUALIFIER(USERID) MEMBER(SAS92)
ACTION(REPLACE) ISOLATION(CS)
DEGREE(ANY) VALIDATE(RUN) RELEASE(COMMIT)
CURRENTDATA(YES) SQLERROR(CONTINUE)
LIBRARY('DB2V7R1.DBRM.DATA')

```

```

BIND PACKAGE(DB2AIX.COLLID) OWNER(USERID) QUALIFIER(USERID) MEMBER(SAS92)
ACTION(REPLACE) ISOLATION(CS) DEGREE(ANY)
VALIDATE(BIND) RELEASE(COMMIT) LIBRARY('DB2V7R1.DBRM.SAS92X.DATA')

```

```

BIND PLAN(SAS92) ACTION(REPLACE) ISOLATION(CS) DEGREE(ANY)
VALIDATE(RUN) RELEASE(COMMIT) ACQUIRE(USE) KEEPDDYNAMIC(NO) NODEFER(PREPARE)
NOREOPT(VAR) CURRENTDATA(YES) SQLRULES(DB2) DISCONNECT(EXPLICIT)
PKLIST(*.COLLID.SAS92, DB2V7R1.DSNUTILS.*)

```

STEP 2: Grant EXECUTE authority to users (required).

Grant EXECUTE authority for the plan created by the BIND command to all users of the plan or to PUBLIC for general use. Issue the GRANT command as follows:

```
GRANT EXECUTE ON PLAN SAS92 TO userid
```

Customizing the SAS CLIST, REXX exec, and Cataloged Procedure

Note: *This task is required.*

If the DB2 interface is used extensively at your site, and if your DB2 system load library is not a link list library, you might want to modify the SAS CLIST, REXX exec, and cataloged procedure to allocate your DB2 system load library by default. See Steps 1, 2 and 3, which follow, for the changes to make.

If you do not make the changes described in Steps 1 and 2, users must specify the LOAD operand shown in the following example when invoking the CLIST. The user must also specify the LOAD= parameter when executing the cataloged procedure to concatenate your DB2 system load library.

Note: The concatenation order of the SAS LOAD LIBRARY and the load library for database access can be interchanged. Generally, if more database access activities will occur in the SAS session or batch job, place the database load library first in the concatenation (or ahead of the SAS LOAD LIBRARY). The opposite is true when SAS processing dominates the session or JOB.

- Enter the following command under TSO to start a SAS session in which you will execute the DB2 interface:

```
SASname LOAD(''your.db2.loadlib'')
```

or

```
SASRXname -DBMSLIB 'your.db2.loadlib'
```

where `your.db2.loadlib` is the DB2 system load library and `SASname` is the name of the CLIST or `SASRXname` is the name of the REXX exec you use to invoke SAS 9.2 Foundation.

- To use the DB2 interface with the SAS cataloged procedure, use the `LOAD=` parameter of the PROC to specify your site's DB2 load library. For example, you can use the following code in your JCL:

```
//SASDB2 EXEC SASname,LOAD='your.DB2.loadlib'
```

STEP 1: Update your SAS CLIST (optional).

If the DB2 interface is used extensively at your site, you might want to alter the CLIST so that your DB2 load library is allocated and concatenated before the SAS load library in the `TASKLIB` symbol. Replace the following line in the installation-supplied CLIST (where `SASLOAD` is a replacement parameter in the CLIST):

```
SASLOAD("&prefix.LIBRARY")+
```

with this line:

```
SASLOAD(''your.db2.loadlib'' '&prefix.LIBRARY'') +
```

STEP 2: Update your SAS REXX exec (optional).

If you need to concatenate the DB2 load library in the SAS tasklib, insert the following option specification into the installation-supplied `SASRXCFG` config file or on the REXX exec command line:

```
-DBMSLIBS 'your.DB2.loadlib'
```

If you want the DB2 load library concatenated before, rather than after, the SAS load library, insert the following option specification into the installation-supplied `SASRXCFG` config file or on the REXX exec command line:

```
-DBMSCONCAT FIRST
```

STEP 3: Update your SAS cataloged procedure (optional).

If the DB2 interface is used extensively at your site with the SAS cataloged procedure, you might want to permanently change the STEPLIB DD statement in your cataloged procedure. To always execute SAS 9.2 Foundation using the DB2 interface, modify the STEPLIB DD statement in your cataloged procedure to reflect the following lines.

```
//STEPLIB DD DISP=SHR,DSN=&LOAD
// DD DISP=SHR,DSN=&prefix.LIBRARY
// DD DISP=SHR,DSN=your.db2.loadlib
```

where *&prefix.LIBRARY* is SAS 9.2 Foundation load library and *your.db2.loadlib* is the DB2 System load library.

Creating and Loading the Sample Tables

Note: This task is optional.

Refer to *SAS/ACCESS 9.2 for Relational Databases: Reference (DB2® under z/OS Chapter)* for a number of coding examples based on sample DB2 tables that can be created at your site. Creating these tables will assist the users at your site in learning how to use the SAS/ACCESS Interface to DB2 product.

The program to create these sample tables is in the SAMPLE library member ACCDATA. The program to run the SAS code using the sample tables is in the SAMPLE library member, ACCRUN. Before running the ACCDATA job, you must first edit the autoexec file called ACCAUTO, also found in the SAMPLE library, to assign two libname statements, one for a SAS library and one for a SAS library of DB2 tables. To execute this program, use the JCL as described in "Customizing the SAS CLIST, REXX exec, and Cataloged Procedure" on page 78, adding the following to allocate the autoexec file:

```
//SASEXEC DD DISP=SHR,DSN=&prefix.SAMPLE(ACCAUTO)
```

Then you can assign the SYSIN DD card referring first to the ACCDATA member in the SAMPLE library as follows:

```
//SYSIN DD DISP=SHR,DSN=&prefix.SAMPLE(ACCDATA)
```

Submit the JCL to run the ACCDATA job; once it completes, you can edit the SYSIN DD statement:

```
//SYSIN DD DISP=SHR,DSN=&prefix.SAMPLE(ACCRUN)
```

and run the ACCRUN job to make use of the Sample tables.

Note: Some customization of this job will be required.

SAS 9.2 Foundation Options for this Interface

Note: This task is optional.

To see a list of SAS 9.2 Foundation options for this interface, invoke SAS 9.2 Foundation and submit the following statements:

```
proc options group = db2; run;
```

Alternatively, you can refer to online help for the current information. You might want to review these system options during installation. All options have defaults.

Most of the options applicable to this interface are invocation options. That means you can change them when you invoke SAS 9.2 Foundation, but not during a SAS session. Some system options have corresponding data set options; these can be set during a SAS procedure or DATA step.

Other system options such as `DB2SSID=` can be set at invocation time or within a SAS session. You are allowed to restrict changes to invocation options by placing them in the Restricted Options Table. This process is described in “Customizing Default Options and System Configuration Files,” on page 11 of this document.

One option, `DB2UPD=`, is suggested as a restricted option. The `DB2UPD=` option determines if, in certain circumstances, the SAS/ACCESS Interface for DB2 engine can perform both read and update operations, or is restricted to read-only. The default option value, `Y`, specifies that both reads and updates are allowed. The option value, `N`, specifies that the engine can only read DB2 data; users who are using access views or libname engine and attempt to update a DB2 file will receive an error message.

Note: Even with the `DB2UPD=` set to `N`, DB2 Tables can be updated by users with PROC SQL Pass-Through Facility, PROC DBLOAD and the Version 5 Compatibility procs.

Special Consideration for Using the RRS Attachment Facility.

The V9 DB2 Access Product supports both RRSAF and CAF as the attachment facility. For usage details, please see the sections “SAS 9.2 Options and Settings for DB2” and “Information for the Database Administrator” in “DB2 under z/OS Chapter, First Edition” of “Part 5, SAS/ACCESS Software: DBMS-Specific Information” in the OnlineDoc *SAS/ACCESS Software for Relational Databases: Reference*.

There are three installation requirements that must be met before RRSAF can be used as the attachment facility:

The SAS 9.2 Foundation SVC Routine must be installed, and must be at Level 7 or greater. See “Installing the SAS 9.2 Foundation SVC Routine” on page 41 for details on the SAS SVC.

Note: The SVC routine shipped with V9 SAS is Level 8.

The load module named DSNRLI is part DB2 and should be found in DB2’s SDSNLOAD load library. DSNRLI must be in an APF authorized load library that is included in the LINKLIST.

The load module named SASD2RUB is part of the SAS DB2 Access Product and should be found in your SAS load library. SASD2RUB must be in an APF authorized load library, which is recommended to be included in the LINKLIST.

With these requirements met, the full functionality of the RRSAF will be supported.

Note: If these requirements are not met, RRSAF can still be used by a non-server SAS 9.2 session, but the Authorized Signon support will not work. Since Authorized Signon support is the main reason for having SAS 9.2 use this facility, these requirements are not considered optional.

Appendix E — Implementing the SAS/ACCESS Interface to IMS-DLI

Customizing the SAS CLIST and Cataloged Procedure

Note: This task is required.

If the IMS interface is used extensively at your site, you might want to alter the CLIST and cataloged procedure so that your IMS libraries are allocated and concatenated.

Note: The concatenation order of the SAS LOAD LIBRARY and the load library for database access can be interchanged. Generally, if more database access activities will occur in the SAS session or batch job, place the database load library first in the concatenation (or ahead of the SAS LOAD LIBRARY). The opposite is true when SAS processing dominates the session or JOB.

STEP 1: Determine the IMS library data set names and identify the IMS libraries you need to allocate at your site.

In order to access your IMS databases, you need to add allocations for the following IMS data sets to your SAS CLIST and cataloged procedure:

- DFSRESLB
- DFSVSAMP (only required for VSAM and OSAM access methods, as defined in the DBD)
- IEFRDER
- IMS
- DD statements for your IMS databases if using DLI or DBB region.

Depending on how your site runs IMS, you might also want to add allocations for these additional data sets:

- IMSACB
- IMSERR
- RECON1
- RECON2.

STEP 2: Add allocations to your SAS CLIST, specifying your IMS library data set names as determined in STEP 1.

- Replace the following line in the installation-supplied CLIST (where SASLOAD is a replacement parameter in the CLIST) so that your IMS RESLIB is concatenated before the SAS Load library in the TASKLIB symbol.

Replace the following:

```
SASLOAD("&prefix.LIBRARY")
```

with the following:

```
SASLOAD("your.ims.reslib" "&prefix.LIBRARY")
```

- Add the following parameter at the top of the default CLIST to allow optional user input of the IMS LOG IEFRDER data set name:

```
/* -----Allow IMS LOG DSN input ----- */ +  
IMSLOG(NULLFILE) /* IMS LOG DSN */ +
```

Add lines like the following in the default SAS CLIST to allocate required IMS libraries:

```
ALLOC F(DFSRESLB) DA('your.ims.reslib') SHR REU
ALLOC F(IMS) DA('your.ims.psblib' 'your.ims.dbdlib') SHR REU
IF &STR(IMSLOG) NE THEN ALLOC F(IEFRDER) DA('&IMSLOG.') OLD
ALLOC F(DFSVSAMP) DA('your.parmlib(DFSVSAMP)') SHR REU
ALLOC F(database) DA('your.ims.database') OLD
```

Note: You only need to allocate your database data sets with disposition OLD if you will be writing to them with the SAS/ACCESS software. Data base data set allocations are not required for a region type of BMP.

- If needed, add lines like the following immediately after the statements you added for required IMS libraries:

```
ALLOC F(IMSACB) DA('your.ims.acblib') SHR REU
ALLOC F(RECON1) DA('your.recon1') SHR REU
ALLOC F(RECON2) DA('your.recon2') SHR REU
ALLOC F(IMSERR) DA('your.dump.data set') SHR REU
```

STEP 3: Add allocations to your SAS cataloged procedure, specifying your IMS library data set names as determined in STEP 1.

Concatenate your IMS RESLIB to your STEPLIB statement in your default SAS cataloged procedure as follows:

```
//STEPLIB DD DISP=SHR,DSN=&LOAD
// DD DISP=SHR,DSN=&prefix.LIBRARY
// DD DISP=SHR,DSN=your.ims.reslib
```

where &prefix.LIBRARY is SAS 9.2 Foundation load library and your.ims.reslib is the IMS RESLIB library.

Add lines like the following in your default SAS cataloged procedure. If you want to use the IMS LOG facility, be sure to include appropriate DD specifications for your site in the IEFRDER DD statement.

```
//DFSRESLB DD DISP=SHR,DSN=your.ims.reslib
//IMS DD DISP=SHR,DSN=your.ims.psblib
// DD DISP=SHR,DSN=your.ims.dbdlib
//IEFRDER DD DSN=NULLFILE,DISP=(,KEEP),
// UNIT=(TAPE,,DEFER),VOLSER=xxxxxxx,
// DCB=(RECFM=VB,BLKSIZE=1920,LRECL=1916,BUFNO=2)
//database DD DISP=OLD,DSN=your.ims.database
```

Note: You only need to allocate your database data sets with disposition OLD if you will be writing to them with the SAS/ACCESS software. Database data set allocations are not required for a region type of BMP.

If needed at your site, also add lines like the following in your default SAS cataloged procedure:

```
//IMSACB DD DISP=SHR,DSN=your.ims.acbllib
//RECON1 DD DISP=SHR,DSN=your.recon1
//RECON2 DD DISP=SHR,DSN=your.recon2
//IMSERR DD DISP=SHR,DSN=your.dump.dataset
```

Verifying Installation of the SAS/ACCESS Interface to IMS

Note: This task is required.

STEP 1: Set up JCL to invoke the standard SAS cataloged procedure at your site.

The additional allocations described in the preceding STEP 3 are not required because TESTIMS does not attach to the IMS DBMS.

Before executing the TESTIMS verification job, you must edit either the LIBNAME or CREATE statement contained in that member of your &prefix.TESTS PDS. You must either supply a valid permanent SAS data library name in place of 'your.sas.library' in the LIBNAME statement, or you might choose to delete the LIBNAME statement and write the test access descriptor to the WORK library by changing the libref in the CREATE statement from 'mylib' to 'work'.

Include a SYSIN DD statement like the following to run the installation verification program for this product:

```
//SYSIN DD DISP=SHR,DSN=&prefix.TESTS(TESTIMS)
```

STEP 2: Submit the job and verify the results.

This job should complete with return code 0.

When this job completes successfully, you have verified the ability to create descriptors in SAS/ACCESS. Since this job does not attach to the DBMS, it does not test the actual interface.

SAS 9.2 Foundation Options for this Interface

Note: This task is required.

To see a list of SAS 9.2 Foundation options for this interface, invoke V9 SAS and submit the following statements:

```
proc options group=ims; run;
```

You can also refer to Appendix 1 in *SAS/ACCESS Interface to IMS-DL/I Software, Version 8 and later, First Edition* for a list of SAS 9.2 Foundation options for this interface. You might want to review the default option settings during installation, as they determine whether the interface can attach successfully to the IMS DBMS.

The BMPREAD=, DLIREAD=, IMSBPUPD=, IMSDLUPD=, IMSID=, IMSREGTP=, and IMSWHST= options for this interface are invocation options. That means you can change them when you invoke SAS 9.2 Foundation, but not during a SAS session. All other options are classified as session options.

You are allowed to restrict changes to invocation options by placing them in the Restricted Options Table. This process is described in “Customizing Default Options and System Configuration Files” on page 11 of this document.

BMPREAD=, DLIREAD=, IMSBPUPD=, and IMSDLUPD= are suggested as restricted options. The BMPREAD= and DLIREAD= options determine whether the SAS/ACCESS Interface to IMS DATA step can perform both read and update operations, or is restricted to read-only. The default value, N, allows update processing. The option value, Y, causes SAS 9.2 Foundation to return a status code of SE and set _ERROR_=1 if a DL/I update call is issued. The IMSBPUPD= and IMSDLUPD= options determine whether the SAS/ACCESS Interface for IMS engine can perform both read and update operations, or is restricted to read-only. The default option value, Y, specifies that both reads and updates are allowed. The option value, N, specifies that the engine can only read IMS data; any attempt to update an IMS file results in an error. For more information about these options, refer to *SAS/ACCESS Interface to IMS-DL/I Software, Version 8 and later*.

Appendix F — Implementing the SAS/ACCESS Interface to Oracle

Customizing the SAS CLIST, REXX Exec, and Cataloged Procedure (required)

If the Oracle interface is used extensively at your site, and if your Oracle system load library is not a link list library, you might want to modify the SAS CLIST, REXX exec, and cataloged procedure to allocate your Oracle system load library by default. See Steps 1 and 2 below for the changes to make.

If you do not make the changes described in Steps 1 and 2, users must specify the `LOAD` operand shown below when invoking the CLIST and the `LOAD=` parameter when executing the cataloged procedure to concatenate your Oracle system load library.

- Enter the following command under TSO to start a SAS session in which you will execute the Oracle interface:

```
SASname LOAD(''your.ORACLE.cmdload'')
```

where `your.ORACLE.cmdload` is the Oracle system load library and `SASname` is the name of the CLIST or REXX exec you use to invoke SAS 9.2 Foundation.

To use the Oracle interface with the SAS cataloged procedure, use the `LOAD=` parameter of the PROC to specify your site's load library. For example, you can use the following code in your JCL:

```
//SASORA EXEC SASname,LOAD='your.ORACLE.cmdload'
```

- If you are running SQL*Net Version 2 or above and you will be accessing the Oracle server as an z/OS client, you might also need to allocate the TNSNAMES file to your current SAS session. Check with your DBA to see if the TNSNAMES file is centrally maintained. If it is not, get the correct data set name for your site's TNSNAMES file.

If you do not have the TNSNAMES file allocated, you will receive the following error from Oracle:

```
ORACLE connection error: ORA-12154 TNSNAMES: could not resolve service name.
```

- Optionally, you might want to allocate the data set that contains the default pathname for connecting to the Oracle database server. This is especially convenient if you are always connecting to the same server. Check with your DBA for the correct `DDname` to the data set name.

STEP 1: Update your SAS CLIST (Optional)

If the Oracle interface is used extensively at your site, you might want to alter the CLIST so that your Oracle load library is allocated and concatenated before the SAS load library in the `TASKLIB` symbol. Replace the following line in the installation-supplied CLIST (where `SASLOAD` is a replacement parameter in the CLIST):

```
SASLOAD(''&prefix.LIBRARY'')+
```

with this line:

```
SASLOAD(''your.ORACLE.cmdload'' ''&prefix.LIBRARY'')+
```

If you are running SQL*Net Version 2 or above and you will be accessing the Oracle server as an z/OS client, you might also need to allocate the TNSNAMES file to your current SAS session. Check with your DBA to see if the TNSNAMES file is centrally maintained. If it is not, get the correct data set name for your site's TNSNAMES file:

```
ALLOC F(TNSNAMES) DA('Your.TNSNAMES.dataset') SHR
```

If you do not have the TNSNAMES file allocated, you will receive the following error from Oracle:

```
ORACLE connection error: ORA-12154: TNSNAMES: could not resolve service name.
```

Optionally, you might want to allocate the data set that contains the default pathname for connecting to the Oracle database server. This is especially convenient if you are always connecting to the same server. Check with your DBA for the correct DDname to the data set name.

```
ALLOC F(Your site's DDname) DA('Your.data.setname') SHR
```

STEP 2: Update your SAS REXX exec (Optional)

If the Oracle interface is used extensively at your site, you might want to alter the REXX exec so that your Oracle load library is allocated and concatenated before the SAS load library in the TASKLIB symbol.

Replace the following lines in the installation-supplied SASRXCFG config file:

```
-SASLOAD(' &prefix.LIBRARY'
        ' &prefix.LIBE' )
```

with the lines

```
-SASLOAD('your.ORACLE.cmdload'
        ' &prefix.LIBRARY'
        ' &prefix.LIBE' )
```

If you are running SQL*Net Version 2 or above and you will be accessing the Oracle server as an z/OS client, you might also need to allocate the TNSNAMES file to your current SAS session. Check with your DBA to see if the TNSNAMES file is centrally maintained. If it is not, get the correct data set name for your site's TNSNAMES file, and in the SASRX exec, insert the following command before the "sastrex" command:

```
"ALLOC F(TNSNAMES) DA('Your.TNSNAMES.dataset') SHR"
```

If you do not have the TNSNAMES file allocated, you will receive the following error from Oracle:

```
ORACLE connection error: ORA-12154: TNSNAMES: could not resolve service name.
```

Optionally, you might want to allocate the data set that contains the default pathname for connecting to the Oracle database server. This is especially convenient if you are always connecting to the same server. Check with your DBA for the correct DDname to the data set name, and in the SASRX exec, insert the following command before the "sastrex" command:

```
"ALLOC F(Your site's DDname) DA('Your.data.setname') SHR"
```

STEP 3: Update your SAS cataloged procedure (optional)

If the Oracle interface is used extensively at your site with the SAS cataloged procedure, you might want to make the STEPLIB DD statement change permanent to your cataloged procedure. To always execute SAS 9.2 Foundation using the Oracle interface, modify the STEPLIB DD statement in your cataloged procedure to reflect the following lines.

```
//STEPLIB DD DISP=SHR,DSN=&LOAD
// DD DISP=SHR,DSN=&prefix.LIBRARY
// DD DISP=SHR,DSN=your.ORACLE.cmdload
```

where &prefix.LIBRARY is SAS 9.2 Foundation load library and your.ORACLE.cmdload is the Oracle System load library.

If you are running SQL*Net Version 2 or above and you will be accessing the Oracle server as a z/OS client, you might also need to allocate the TNSNAMES file to your current SAS session. Check with your DBA to see if the TNSNAMES file is centrally maintained. If it is not, get the correct data set name for your site's TNSNAMES file.

```
//TNSNAMES DD DISP=SHR,DSN=your.TNSNAMES.dataset
```

If you do not have the TNSNAMES file allocated, you will receive the following error from Oracle:

```
ORACLE connection error: ORA-12154: TNSNAMES:could not resolve service name.
```

Optionally, you might want to allocate the data set that contains the default pathname for connecting to the Oracle database server. This is especially convenient if you are always connecting to the same server. Check with your DBA for the correct DDname to the data set name.

```
//yourDDname DD DISP=SHR,DSN=your.dataset.name
```

Creating and Loading the Sample Tables (optional)

Refer to *SAS/ACCESS 9.2 for Relational Databases: Reference, (SAS/ACCESS Software: Appendices, Sample Data)* for coding examples based on sample Oracle tables that can be created at your site. Creating these tables will assist the users at your site in learning how to use the SAS/ACCESS Interface to Oracle product.

The program to create these sample tables is in the SAMPLE library member ACCDATA. The program to run the SAS code using the sample tables is in the SAMPLE library member ACCRUN. Before running the ACCDATA job, you must first edit the autoexec file called ACCAUTO, also found in the SAMPLE library, to assign two libname statements, one for a SAS library and one for a SAS library of Oracle tables. To execute this program, use the JCL as described in "Customizing the SAS CLIST, REXX Exec, and Cataloged Procedure (required)" on page 87, adding the following to allocate the autoexec file:

```
//SASEXEC DD DISP=SHR,DSN=&prefix.SAMPLE(ACCAUTO)
```

Then you can assign the SYSIN DD card referring first to the ACCDATA member in the SAMPLE library as follows:

```
//SYSIN DD DISP=SHR,DSN=&prefix.SAMPLE(ACCDATA)
```

Submit the JCL to run the ACCDATA job; once it completes, you can edit the SYSIN DD statement:

```
//SYSIN DD DISP=SHR,DSN=&prefix.SAMPLE(ACCRUN)
```

and run the ACCRUN job to make use of the Sample tables.

Note: *Some customization of this job will be required.*

Appendix G — Implementing the SAS/ACCESS Interface to R/3

SAS/ACCESS Interface to R/3 software requires extensive installation setup before it can be used. Refer to the *Installation Instructions for SAS/ACCESS 4.2 Interface to R/3 Software on SAS 9.2* included in your SAS order for detailed installation instructions and configuration information.

Appendix H — Implementing the SAS/ACCESS Interface to SYSTEM 2000

Customizing the SAS CLIST, REXX Exec, and Cataloged Procedure

Note: This task is required.

If the SYSTEM 2000 Interface is used extensively at your site, and if your SYSTEM 2000 load library is not a link list library, you might want to modify the SAS CLIST and cataloged procedure to allocate your SYSTEM 2000 load library by default. The file S2KCOM is required for SYSTEM 2000 Multi-User, the file S2KPARMS is required for SYSTEM 2000 single-user, and the file S2KDEFC is required for both Multi-User and single-user. These must also be allocated to your CLIST and procedure. See Steps 1 and 2 below for the changes to make.

Note: The concatenation order of the SAS LOAD LIBRARY and the load library for database access can be interchanged. Generally, if more database access activities will occur in the SAS session or batch job, place the database load library first in the concatenation (or ahead of the SAS LOAD LIBRARY). The opposite is true when SAS processing dominates the session or JOB.

STEP 1: Add allocations to your SAS CLIST (Optional)

Replace the following line in the installation-supplied CLIST (where SASLOAD is a replacement parameter in the CLIST):

```
SASLOAD('' &prefix.LIBRARY''')+
```

with this line:

```
SASLOAD(''your.SYSTEM2000.load'' '' &prefix.LIBRARY''')+
```

In addition, you need to allocate three SYSTEM 2000 specific files in your SAS CLIST by adding the following:

```
/* Allocate SYSTEM 2000 file for DBLOAD  
ALLOC F(S2KDEFC) SP(1 1) CYL +  
LRECL(100) BLKSIZE(100) RECFM(F) REU
```

```
/* Allocate SYSTEM 2000 Multi-User specific file  
ALLOC F(S2KCOM) DA('your.SYSTEM2000.s2kcom.file') SHR REU
```

```
/* Allocate SYSTEM 2000 single-user specific file
ALLOC F(S2KPARMS) DA('your.SYSTEM2000.CNTL(NLPARM)) SHR REU
```

STEP 2: Update your SAS REXX exec (Optional)

If you need to concatenate the System 2000 load library in the SAS tasklib, insert the following option specification into the installation-supplied SASRXCFG config file or on the REXX exec command line:

```
-DBMSLIBS 'your.SYSTEM2000.load'
```

If you want the System 2000 load library concatenated before, rather than after, the SAS load library, insert the following option specification into the installation-supplied SASRXCFG config file or on the REXX exec command line:

```
-DBMSCONCAT FIRST
```

In the SASRX exec, insert the following before the "sastrex" command:

```
/* Allocate SYSTEM 2000 file for DBLOAD */
"ALLOC F(S2KDEFC) SP(1 1) CYL LRECL(100) BLKSIZE(100) RECFM(F) REU"

/* Allocate SYSTEM 2000 Multi-User specific file */
"ALLOC F(S2KCOM) DA('your.SYSTEM2000.s2kcom.file') SHR REU"

/* Allocate SYSTEM 2000 single-user specific file */
"ALLOC F(S2KPARMS) DA('your.SYSTEM2000.CNTL(NLPARM)') SHR REU"
```

STEP 3: Add allocations to your SAS cataloged procedure (optional)

Concatenate your SYSTEM 2000 LOAD library to your STEPLIB statement in your default SAS cataloged procedure as follows:

```
//STEPLIB DD DISP=SHR,DSN=&LOAD
// DD DISP=SHR,DSN=&prefix.LIBRARY
// DD DISP=SHR,DSN=your.SYSTEM2000.load
```

Allocate the following SYSTEM 2000 Multi-User file in your SAS procedure for Multi-User access:

```
//S2KCOM DD DSN=your.SYSTEM2000.s2kcom.file,DISP=SHR
```

Allocate the following SYSTEM 2000 single-user specific file in your SAS procedure for single-user access:

```
//S2KPARMS DD,DSN=your.SYSTEM2000.CNTL(NLPARM),DISP=SHR
```

Allocate the following SYSTEM 2000 file in your SAS procedure for both Multi-User and single-user access:

```
//S2KDEFC DD UNIT=SYSDA,SPACE=(CYL,(1,1)),
// DCB=(RECFM=FB,LRECL=100,BLKSIZE=100)
```

With all three S2KCOM, S2KDEFC, and S2KPARMS files present, your SAS procedure can access both Multi-User and single-user SYSTEM 2000.

Appendix I — Implementing the SAS/ACCESS Interface to Teradata

Defining the Interface to Teradata

Teradata CLIV2 and TDP software for z/OS must be installed before using SAS/ACCESS Interface to Teradata. The procedure to install this software is described in the *Teradata® Tools and Utilities Installation Guide for MVS* supplied by Teradata. Once the Teradata software is installed, connectivity to the Teradata RDBMS must be verified. Connectivity can be tested with SAS/ACCESS, or with BTEQ, FastLoad, or other Teradata utilities on z/OS. If you are already running any Teradata utility, then no setup is necessary — your Teradata software is already correctly configured for use by SAS/ACCESS Interface to Teradata.

Along with the customer's installation of Teradata client software, the system administrator must have started and initialized the Teradata Director Program (TDP). This communication task must be running before the SAS/ACCESS Interface to Teradata product can communicate with the Teradata server. Names of the TDPs must be communicated to the SAS user community if a default TDP (see below) is not established.

Note: *A typical installation might include two TDPs; one named "TDP0 - production TDP" and another "TDP1 - test TDP." The test TDP could be used for the testing of new versions of Teradata, new applications, etc. The production TDP's role would be to communicate with the production database server.*

The system administrator can set up a default TDP to be used by Teradata client applications, including SAS sessions. To do this, the System Parameter Block (HSISPB and HSHSPB) must be modified to indicate the default name.

Note: *This is a Teradata parameter block and not a SAS parameter block.*

Once the default TDP is defined, then SAS will use this TDP by default when connecting to the Teradata server.

Customizing the SAS CLIST, REXX exec, and Cataloged Procedure

After the Teradata components are in place, tested and defined, the system administrator needs to configure SAS 9.2 Foundation to use the Teradata load libraries. SAS/ACCESS Interface to Teradata under z/OS uses the APPLOAD load library supplied by Teradata. This library must be accessible to SAS 9.2 Foundation in order for SAS/ACCESS Interface to Teradata to function. There are several methods that can be used to accomplish this task:

- Enter the following command under TSO to start a SAS session in which you will execute the SAS/ACCESS Interface to Teradata under z/OS interface:

```
SASname LOAD(' 'your.ncr.appload' '')
```

or

```
SASRXname -DBMSLIB 'your.ncr.appload'
```

where `your.ncr.appload` is the Teradata system load library and `SASname` is the name of the CLIST or `SASRXname` is the name of the REXX exec you use to invoke SAS 9.2 Foundation.

- To use the Teradata interface with the SAS cataloged procedure, use the LOAD= parameter of the PROC to specify your site's Teradata load libraries. For example, you can use the following code in your JCL:

```
//SASTRA EXEC SASname,LOAD='your.ncr.appload'
```

where `your.ncr.appload` is the Teradata system load library and `SASname` is the name of the cataloged procedure you use to invoke SAS 9.2 Foundation.

- You can modify your SAS CLIST to make the necessary Teradata libraries accessible when you invoke SAS 9.2 Foundation. Follow these steps to do this:

1. Edit your SAS CLIST.

2. Replace `SASLOAD(' '&prefix.LIBRARY' ')+` with this line:

```
SASLOAD(' 'your.ncr.appload' ' '&prefix.LIBRARY' ')+
```

- If you need to concatenate the Teradata load library in the SAS tasklib, insert the following option specification into the installation-supplied SASRXCFG config file or on the REXX exec command line:

```
-DBMSLIBS 'your.NCR.appload'
```

If you want the Teradata loadlib concatenated before, rather than after, the SAS loadlib, insert the following option specification into the installation-supplied SASRXCFG config file or on the REXX exec command line:

```
-DBMSCONCAT FIRST
```

- You can modify your SAS cataloged procedure by changing the STEPLIB DD statement as shown below.

```
//STEPLIB DD DISP=SHR,DSN=&LOAD
// DD DISP=SHR,DSN=&prefix.LIBRARY
// DD DISP=SHR,DSN=your.ncr.appload
```

Creating and Loading the Sample Tables (optional)

Refer to *SAS/ACCESS 9.2 for Relational Databases: Reference, (SAS/ACCESS Software: Appendices, Sample Data)* for coding examples based on sample Teradata tables that can be created at your site. Creating these tables will assist the users at your site in learning how to use the SAS/ACCESS Interface to Teradata product.

The program to create these sample tables is in the SAMPLE library member ACCDATA. The program to run the SAS code using the sample tables is in the SAMPLE library member ACCRUN. Before running the ACCDATA job, you must first edit the autoexec file called ACCAUTO, also found in the SAMPLE library, to assign two libname statements, one for a SAS library and one for a SAS library of Teradata tables. To execute this program, use the JCL as described in "Customizing the SAS CLIST, REXX exec, and Cataloged Procedure" on page 95, adding the following to allocate the autoexec file:

```
//SASEXEC DD DISP=SHR,DSN=&prefix.SAMPLE(ACCAUTO)
```

Then you can assign the SYSIN DD card referring first to the ACCDATA member in the SAMPLE library as follows:

```
//SYSIN DD DISP=SHR,DSN=&prefix.SAMPLE(ACCDATA)
```

Submit the JCL to run the ACCDATA job; once it completes, you can edit the SYSIN DD statement:

```
//SYSIN DD DISP=SHR,DSN=&prefix.SAMPLE(ACCRUN)
```

and run the ACCRUN job to make use of the Sample tables.

Note: Some customization of this job will be required.

Configuration for FastExporting (optional)

For optimal reads of large tables, SAS/ACCESS can perform FastExporting. To perform FastExporting, the Teradata FastExport Utility must be present on the system where you install SAS. The FastExport Utility is not required; SAS/ACCESS reads large tables quite efficiently without it. For further information, see the DBSLICEPARM option in your SAS/ACCESS Interface to Teradata documentation. Contact Teradata if you want to obtain the Teradata FastExport Utility.

Configuration for MultiLoading (optional)

SAS/ACCESS can interface with MultiLoad for loading large volumes of data. To perform MultiLoading, the Teradata MultiLoad Utility must be present on the system where you install SAS. The MultiLoad Utility is not required; SAS/ACCESS provides other options for loading tables. For further information, see the MULTISTMT option in your SAS/ACCESS Interface to Teradata documentation. Contact Teradata if you want to obtain the Teradata MultiLoad Utility.

Appendix J — Post-Installation Setup for SAS/ASSIST Software

This appendix describes how to add an optional master profile to SAS/ASSIST software. You can use a master profile to override the default settings as sent by SAS. This allows you to provide a customized setup for SAS/ASSIST software. With the master profile you can control the profile options of all SAS/ASSIST users from one central place. For more information on the profile options, refer to *SAS/ASSIST Software: Changes and Enhancements*.

Adding a Master Profile

Complete the following steps to add a master profile to SAS/ASSIST software:

1. Specify the location of the master profile by creating a new SAS library that all users of SAS/ASSIST Software will have read-access to. The following definitions are needed:

```
Data Set Name prefix.ASSIST.MASTER
Space units      CYLINDER
Primary quantity 1
Secondary quantity 1
Directory blocks 0
Record format FS
Record length 6144
Block size       6144
```

Example: prefix.ASSIST.MASTER

All users with write-access to this library will automatically also have write-access to the master profile in SAS/ASSIST software. Select a name that conforms to the naming conventions at your installation. The name of this new library must be stored in an entry in the SASHELP library. This requires that you have write access to the SASHELP library.

- ❑ Modify the CLIST provided with installation to change the disposition of the SASHELP library to OLD.
- ❑ Invoke SAS 9.2 Foundation. Only one user at a time will be able to run SAS using the modified CLIST.
- ❑ On line 1 of the Program Editor type the physical pathname to be used as the SAS data library to store the master profile. Execute the `Save` command to save the entry as shown in the following example:

```
SAVE SASHELP.QASSIST.PARMS.SOURCE
00001 prefix.ASSIST.MASTER
00002
```

Note: The entry must be saved with the name `SASHELP.QASSIST.PARMS.SOURCE`.

The location of the master profile is now known by SAS/ASSIST software.

- ❑ Exit SAS 9.2 Foundation.

- ❑ Free the SASHELP data set to release the exclusive lock:

```
FREE DS('&prefix.SASHELP')
```

- ❑ Modify the CLIST provided with installation to change the disposition of the SASHELP library back to SHR. Now other users are able to run SAS concurrently.

2. Create the master profile.

The first time SAS/ASSIST software is started a master profile is created if the SASHELP.QASSIST.PARMS.SOURCE contains the name of an existing library, and the person who starts SAS/ASSIST software has write-access to this library.

3. Customize the master profile by starting SAS/ASSIST and typing the command ASSIST on the SAS command line. Then select

```
Setup ... Profiles ... Master/group ...
```

If you have write-access to the SAS data library containing the master profile you can specify default values for your installation. These values will be used by new users as they start SAS/ASSIST software.

Note: *If you restrict values by typing R in Status, users will not be allowed to change the values you define.*

You can run SAS/ASSIST software in two different styles – workplace or block menu. The block menu can be new style or old style. You can control this using the profile options below.

Run workplace:

```
SAS/Assist style:           Workplace
```

```
Run new style with block menu:
```

```
SAS/Assist style:           Block Menu
Menu style:                  New
```

```
Run old style:
```

```
SAS/Assist style:           Block Menu
Menu Style:                  Old
```

4. Create Group profiles.

From the master profile it is possible to create group profiles to allow groups of users to have different setups. The master profile controls group profiles and user profiles when a user is not a member of any group. All users are indirectly controlled by the master profile when option values are set to a restricted (R) status.

From Setup ... Profile ... Master/Group ... select Tools ... Create Group Profile. To add users to a group profile, select Tools ... Update User Group. By default, the user ID is found in the macro variable &SYSJOBID. The value is set in the option Userid in the master profile (option type System Administration). Change the value if your site uses another variable to keep the user ID.

Installing Sample DB2 Tables and a Sample Query Manager

This section describes how to customize SAS/ASSIST software to run queries against DB2 sample data. It is recommended that you do this in order to get acquainted with the software before you start working with your own DB2 data. This example sets up a personal query manager. See Chapter 9, "Setting Up Query Managers," in *SAS/ASSIST Software: Changes and Enhancements* for more information.

Complete the following steps:

4. Create DB2 Sample Tables.

Query and Reporting contains its own sample DB2 tables. These tables are used in the documentation and in the SAS training courses. Complete the following steps to create DB2 sample tables:

- a. Invoke SAS and enter the following on the command line in the Program Editor:
COPY SASHELP.QASSIST.SAMPLTAB.SOURCE
- b. Specify the DB2 subsystem id, creator, and database name (lines 54-59):
See documentation in the program for further information.
- c. Execute this SAS program and the DB2 sample data will be created.

5. Create Query Manager Tables on the DB2 Sample Data

Complete the following steps to create query manager tables for installation verification purposes:

- a. In the Program Editor, use the `COPY` command to copy the sample source from `SASHELP.QASSIST.DB2METAU.SOURCE`. This program will generate a set of Query Manager tables, which contain information on DB2 tables available to specific creators.
- b. Specify the creators you want to use in line 21 as shown in the following example:

```
%let creators=( 'SASQR' , 'DSN9030' ,USER) ;
```

Note: In this example `SASQR` and `DSN9030` are two specific creators which contain sample data from SAS and IBM. Add `USER` if you want access to a DB2 table, which has a creator equal to your user ID.

- c. Specify the location of the Query Manager tables on line 25.

```
%let saslib=SASUSER; *<--User QM library;
```

- d. Execute this SAS program and the Query Manager tables will be created. The following SAS tables will be created:

SASUSER_DB2TAB	DB2 table information
SASUSER_DB2COLS	DB2 column information
SASUSER_DB2RELS	Relations for joining tables
SASUSER_DB2INDX	Defined indexes in DB2.

6. Create the Query Manager Definitions

The Query Manager contains information that indicates the database to be used, as well as the location of the SAS Query Manager tables created in the last step. You can also specify the name of the program to be used to recreate (refresh) the SAS Query Manager tables. See Chapter 9 in *SAS/ASSIST Software: Changes and Enhancements* for more information.

- a. Select the following from the SAS/ASSIST Primary menu to get to the Query Manager Administration window:

```
DATA MGMT...
QUERY...
Query and Reporting...
SETUP...
Administration
```

- b. Enter the following information to identify the location of the Query Manager tables as shown below:

- For Database, enter DB2
- For Program to Generate Query Manager tables, select No
- For Location of the Query Manager tables, enter SASUSER

Note: The remaining parameters are selected by default.

- c. Save the Query Manager definition.

Select **Save As** from the **File** menu.

- d. Enter name and description as shown in the following example (the Type and Catalog are already specified):

```
Type:          MANAGER
Name:          db2samp
Description:    Sample DB2 data
Catalog:       SASUSER.MANAGER
```

- e. Select OK to save it.

7. Using the Query Manager

- a. Select the following from the SAS/ASSIST Primary menu to get to the Query window:

```
DATA MGMT...
QUERY...
Query and Reporting...
QUERY...
```

The Query window might attempt to load a Query Manager other than the one you defined (in Step 3 above), or you might be notified that one or more Query Manager tables are not found. You will be asked if you want to generate them. Select **No**.

- b. To use the Query Manager that you defined (see Steps 2 and 3 above), select **Select Query Manager...** from the **Tools** menu.

A pop-up menu is displayed. Select the Query Manager you just defined DB2SAMP (in Step 3 above).

- c. Select the **select** button to get a list of DB2 data.

Appendix K — Installing the BMDP Interface

Introduction

The BMDP procedure (invoked in the SAS language as PROC BMDP) allows you to call BMDP modules. BMDP modules (such as BMDP1D) were part of the Biomedical Programs package available in the past from BMDP Statistical Software Inc. (later part of SPSS Inc.). The BMDP package is no longer available, but sites might still be using this software; therefore, the BMDP procedure continues to be supported within the SAS language. Please note that the BMDP package is *not* Y2K compliant, and precautions should be taken when running modules in this package. The BMDP procedure within the SAS language is Y2K compliant, however.

The BMDP procedure (PROC BMDP) converts an input SAS data set into a BMDP save file and loads user-provided BMDP control statements for processing the file. It then calls a BMDP program (for example, BMDP1D) to execute the control statements. PROC BMDP then prints the output of the BMDP program, interspersing by-line information if a BY statement is given.

The SAS BMDP interface is not necessary if your site does not have BMDP, or if you are only accessing BMDP data through the BMDP engine (via the LIBNAME statement or PROC CONVERT). You might want to survey your SAS users to determine if and how the BMDP interface will be used.

Installation of the BMDP Interface

Note: This task is optional.

If your users plan to run PROC BMDP, you need to customize the SASBMDP CLIST and cataloged procedure (PROC) and install them in the appropriate libraries for user access at your site.

STEP 1: Edit the BMDPEDTP member of the CNTL data set to specify the desired values for the BMDP parameters.

The BMDP parameters are defined as follows:

- **BMDPPNM=** specifies the name of the PROC to execute the SAS BMDP interface at your site.
- **BMDPCNM=** specifies the name of the CLIST to invoke the SAS BMDP interface at your site.
- **BMDPLOD=** specifies the name of the load library that contains the BMDP programs. This item is **required**.
- **BMDPNWS=** specifies the BMDP news file name. This is **optional**, and displays as part of the BMDP program's output. Use NULLFILE if you do not want to view the BMDP news file name in your log.
- **BMDPMAC=** specifies the name of the BMDP macro file. This is only used by the BMDP program and is optional. Use NULLFILE if it is not wanted.

STEP 2: Review and submit the BMDPPOST job in the CNTL data set.

Submit the BMDPPOST job to tailor the CLIST and PROC, as well as the BABMDP job.

STEP 3: Review the SASBMDP CLIST and PROC located in your CNTL data set.

Member BACLST02 contains the CLIST. BAPROC02 contains the PROC. The cataloged procedure and CLIST contain file allocations for three sets of files. These files are required for SAS 9.2 Foundation, PROC BMDP, and the BMDP program.

If you have customized your SAS CLIST and cataloged procedure as described in the section “Customizing your SAS CLIST and Cataloged Procedure” on page 21, you might want to apply the same customizations here as well.

The files required for PROC BMDP are identified by the DDnames FT05F001, FT06F001, and FT03F001, which are defined as follows:

- **FT05F001** identifies the file into which PROC BMDP writes the BMDP control statements.
- **FT06F001** identifies the file into which PROC BMDP directs the BMDP program output listing for post-processing.
- **FT03F001** identifies the file into which PROC BMDP writes the converted BMDP save file. While the user can have PROC BMDP write to any unit number, FT03F001 is the default. The user must supply the additional JCL if another DDname is used.

STEP 4: Review and then submit the BABMDP job in the CNTL data set that was tailored by the BMDPPOST job submitted in Step 2. This job copies the tailored SASBMDP CLIST and PROC to the libraries that you specified earlier in SASEDITP and executes the TESTBMDP validation program to verify the installation of the BMDP interface.

Appendix L — Post-Installation Setup for SAS/CONNECT Software

Note: *The post-installation setup for the SAS/CONNECT Spawner for z/OS is different than previous releases. Please read this entire section carefully.*

Note: *For further information on the communication part of implementing and using SAS/CONNECT software, refer to Communications Access Methods for SAS/CONNECT and SAS/SHARE Software and the SAS/CONNECT User's Guide which are both in the SAS OnlineDoc.*

Note: *To use SAS/CONNECT software, both the local and remote hosts must be running and licensed for SAS/CONNECT.*

Note: *Installation of the SAS 9.2 Foundation SVC is **absolutely required** if the SAS/CONNECT Spawner for z/OS or SAS/CONNECT to the same multi-process machine on z/OS is to be implemented. See "Installing the SAS 9.2 SVC Routine" on page 41 for details.*

Configuring SAS/CONNECT

This appendix shows you how to configure SAS/CONNECT. The following topics are covered:

1. communication access methods supported for SAS/CONNECT on z/OS
 - A. TCP (used with basic telnet session and with the z/OS SAS/CONNECT Spawner)
 - B. XMS (used in Configuration of SAS/CONNECT to same multi-processor machine)
2. storing and locating SAS/CONNECT Script files
3. connection types available with SAS/CONNECT on z/OS
 - A. SAS/CONNECT Basic Telnet session
 - B. SAS/CONNECT Spawner
 - C. SAS/CONNECT to same multi-processor machine

1. Communication Access Methods Supported for SAS/CONNECT on z/OS

Which access method to use depends on the type of connection being set up. The access methods supported for z/OS are TCP/IP and XMS. Refer to the appropriate sections for the access methods you will be using at your site, for requirement information. Refer to *Communications Access Methods for SAS/CONNECT 9.2 and SAS/SHARE 9.2* for additional details on the access methods supported by other systems.

A. System Configuration for the TCP Access Method

To use the TCP/IP access method with SAS/CONNECT software you must have installed an configured the IBM z/OS Communications Server package, or another vendor's software that is functionally compatible with the IBM product. For details on configuring TCP/IP for use with SAS/CONNECT Software, refer to "System Configuration for Using SAS with TCP/IP" on page 21.

B. System Configuration XMS (Cross Memory Services) Access Method

For details on configuring XMS refer to section "SAS/CONNECT to the Same Multi-Process Machine on z/OS" on page 115.

2. Storing and Locating SAS/CONNECT Script Files

Several sample script files are shipped with SAS/CONNECT software. SAS/CONNECT software uses these script files to establish a connection to a remote SAS session by physically logging the user on to the remote host. The install process places the script files into your `&prefix.CTMISC` data set. These script files must be customized to match the logon procedures that are specific to your site. Script files are used to establish a SAS/CONNECT basic Telnet session and can optionally be used to establish a SAS/CONNECT Spawner session.

To use the script file with the SAS/CONNECT product, a file reference of RLINK must be allocated to a customized script file. There are other methods of allocating the script file, for more details on this, refer to the *SAS/CONNECT User's Guide*.

The SAS option, `SASSCRIPT`, points to the location of the SAS/CONNECT script files. The `SASSCRIPT` option is only used by SAS/ASSIST and by user-written SCL applications.

The value of the `SASSCRIPT` option can be one or more concatenated data set names (including PDS files).

```
SASSCRIPT=( 'MVS:prefix.CTMISC' 'userid.CTMISC' )
```

The simplest way for you to give your users automatic access to the installed sample script files is to place the above option in your configuration file.

Note: The script file must be customized to match the logon process to your remote host. Connecting to the remote host via Telnet outside of SAS is recommended to see the necessary screens and messages that need to be handled by the script.

3. Types of connections available with SAS/CONNECT on z/OS

SAS/CONNECT Basic Telnet session

Note: The SAS/CONNECT basic Telnet session requires that you configure SAS for communication with TCP/IP. For this reason, please ensure that you have reviewed and completed the steps in the section "System Configuration for Using SAS with TCP/IP" on page 21.

Requirements:

- To use the basic telnet SAS/CONNECT signon, telnet must be enabled on both local and remote hosts (either in line mode or fullscreen/TN3270).
- A customized script needs to be set up on the local host. See the previous section "2. Storing and Locating SAS/CONNECT Script Files" for more information.
- A valid user ID for the remote host.

For details on using and implementing the basic telnet SAS/CONNECT feature, refer to the *SAS/CONNECT User's Guide*.

SAS/CONNECT Spawner for z/OS

*Note: The SAS/CONNECT Spawner for z/OS requires that you configure SAS for communication with TCP/IP. For this reason, please ensure that you have reviewed and completed the steps in the section “**System Configuration for Using SAS** with TCP/IP” on page 21.*

Benefits to Using the Spawner:

There are several benefits to using the spawner. Use of the spawner

- does not require TELNET.
- does not require being physically logged on to TSO. Since this is not a TSO logon session, the user can sign on several times concurrently.
- does not require using a signon script (it is optional).
- initiates SAS/CONNECT sessions on z/OS systems without requiring that username and password pairs be passed over the network in clear text mode.
- encrypts all data that flows from the local host to the spawner program during sign on by default (if the local SAS session is running Release 6.09E , Release 6.11 TS040, or a subsequent release,). The SAS/CONNECT Spawner program supports encrypted signons, by default, to a z/OS system with or without scripts.
- facilitates use of SAS/CONNECT through a firewall. The spawner supports socket inheritance thereby limiting the number of open ports.

SAS/CONNECT Spawner Security Configuration

Security requirements for the SAS 9.2 SAS/CONNECT Spawner running on z/OS differ from previous releases.

- The spawner no longer needs to run with a uid=0
- Load libraries do not need to be APF-authorized
- The user ID of the spawner process does not require READ access to the BPX.DAEMON RACF profile

The spawner runs as a daemon process, so correct daemon security needs to be implemented based on the security software running on your machine (RACF, ACF2, or TOP Secret). IBM defines two levels of security for daemon processes, traditional UNIX security and z/OS UNIX System Services security. For SAS/CONNECT Spawner 9.2, these daemon processes do not require a UID of '0' but should be loaded from a program-controlled library if the BPX.DAEMON is defined.

The spawner validates user IDs and passwords using the `__passwd` service. It then starts an address space for the SAS/CONNECT session, using the USS spawn service.

Security Requirements for Client/User

You must have an OMVS segment defined in the security software (either an individual segment or a site-default OMVS segment.)

Security Requirements for Spawner

- An OMVS segment.
- Read Access for the load library.
- Update access to the HFS logs defined in the started task.
- If the BPX.DAEMON is defined, then the `&prefix.LIBRARY` needs to be under program control.

The SAS/CONNECT Spawner uses the UNIX System Services (USS) `__passwd` service to validate the user ID and password of clients connecting to the spawner. If the client authenticates successfully, the spawner uses the USS `spawn` service to invoke a remote SAS session for the client.

What follows are examples of security definitions for RACF. If your site uses ACF2 and TOP Secret security products, please consult the associated security software documentation for configuring UNIX System Services daemon security.

RACF Security

The RACF Security Server defines the `BPX.DAEMON` profile of the RACF facility class to allow a daemon program to spawn processes with user IDs different from that of the daemon. This allows the site to run a daemon and associate it with a minimally-authorized user ID that does not have root user (`uid=0`) authority. To assign a user ID to the started task, either add the started task to the RACF Started Procedures Table `ICHRIN03`, or define a profile for the started task in the RACF started class.

If RACF program control is enabled on your system and if the `BPX.DAEMON` profile is active, RACF requires that all modules loaded into the spawner address space come from RACF program-controlled libraries. This applies to the SAS load library (`'prefix.LIBRARY'`). If this is not set up correctly, the spawner will receive a USS `JREnvDirty` reason code on the `__passwd` call. These messages will be issued to the system log.

```
ICH420I PROGRAM xxxxxxxx CAUSED THE ENVIRONMENT TO BECOME UNCONTROLLED .
BPXP014I ENVIRONMENT MUST BE CONTROLLED FOR DAEMON (BPX.DAEMON)
PROCESSING .
```

To diagnose a dirty address space, look at informational APAR II08176 and the IBM documentation it references.

To list the status of Program Control, issue the following command:

```
SETROPTS LIST
```

If the results of the above command list `WHEN(PROGRAM)` in the `ATTRIBUTES`, then RACF Program Control is already active.

Otherwise, to enable Program Control, use the following command:

```
SETROPTS WHEN(PROGRAM)
```

When Program Control is enabled, all modules loaded into the spawner's address space must come from a RACF Program-Controlled Library. The command to make the SAS load library program controlled is:

```
RDEFINE PROGRAM ** UACC(READ) ADDMEM +
('prefix.SAS.LOAD'//NOPADCHK)
```

Use `RALTER` command instead of `REDEFINE` if `*` or `**` already exists from a previous definition.

To refresh any changes, issue the following RACF command. It does not matter if the RACF databases are shared or not.

```
SETROPTS WHEN(PROGRAM) REFRESH
```

To enable the RACF BPX.DAEMON profile of the FACILITY class, issue the following commands:

```
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
SETROPTS CLASSACT(FACILITY)
SETROPTS RACLIST(FACILITY)
SETROPTS RACLIST(FACILITY) REFRESH
```

Note that the spawner user ID does not require READ access to the BPX.DAEMON profile. For more information, please refer to your *z/OS Security Server (RACF) Command Language Reference*.

Setting up the SAS/CONNECT Spawner Procedure

The SAS/CONNECT Spawner runs as a z/OS started task and uses z/OS UNIX System Services (USS) to start each user's SAS/CONNECT session. Each session runs in a BPXAS address space, executing the UNIX System Services `/bin/tso` command to run the SAS REXX startup command.

The spawner module is SASTCPD and is located in the SAS load module library. This library might be installed in LPA, LINKLIST, or allocated to the STEPLIB DD in the Spawner Started Task JCL. A sample spawner started task procedure can be found in `'&prefix.BAMISC(SPNCCNTL)'`.

The Spawner Started Task requires a parms file. A sample parms file can be found in `'&prefix.BAMISC(SPNC Parm)'`. Here is the syntax for the options you might specify in the parms file to configure the SAS/CONNECT Spawner program for z/OS:

```
<-HELP>
<-INHERITANCE | -NOINHERITANCE>
<-NETENCRALG algorithm>
<-NETENCRYPT>
<-NETENCRKEY n>
<-NOCLEARTEXT>
<-NOSCRIP T>
<-SASCMD command>
<-SERVICE service-name>
```

❑ **-HELP**

prints a list of valid options and terminates the spawner started task.

❑ **-INHERITANCE | -NOINHERITANCE**

Specifies whether the SAS session that is spawned inherits the socket that was created when the spawner accepted the initial connection from the local SAS session. This option is useful if your configuration involves a firewall and you want to minimize the number of ports that you define to the firewall for use by SAS/CONNECT. The default is `-INHERITANCE`.

If you start a spawner with the `-INHERITANCE` option, you then define the port that the spawner is listening on to the firewall and map it to the server machine's port. This will enable any number of SAS/CONNECT clients to connect through this single port and SIGNON to a remote host on the inside of the firewall. Each client just opens a unique socket on the defined port. This eliminates the need to

define an individual port for each client that might need to come in through the firewall. In this configuration you set your REMOTE= value to a two-level name where the first level is the name of the host running the firewall and the second level is the well-known service name of the port that you have enabled for connections.

❑ **-NETENCALG algorithm**

To specify more than one algorithm, simply repeat this option.

Set this option at the remote host and, optionally, at the local host to specify one or more encryption algorithms to use in a SAS/CONNECT session. However, the local host and the remote host must share an encryption algorithm in common. If you specify the option in the remote host session only, the local host attempts to select an algorithm that was specified at the remote host. If you also set the option at the local host and specify an algorithm that is not specified at the remote host, the local host's attempt to connect to that remote host fails.

The following are valid values for this option:

- SSL
- AES*
- RC2*
- RC4*
- DES*
- TripleDES*
- SASPROPRIETARY

* indicates the algorithm is available with SAS/SECURE

See the *SAS/CONNECT User's Guide* or the *SAS/SHARE User's Guide* for more information about the NETENCALG option.

❑ **-NETENCRYPT**

Set this option at the local host and/or the remote host. At the remote host, this option specifies that encryption is required for each connection from a local host SAS session. At the local host, this option specifies that the local host must connect only to a remote host that supports encryption.

The default for this option is that encryption is used if the NETENCALG option is set and if both the local host and the remote host are capable of encryption. If encryption algorithms were specified but either the local host or the remote host is incapable of encryption, then encryption will not be performed.

Encryption might not be supported at the local host or the remote host for the following reasons:

- You are running a release of SAS (prior to Version 7) that does not support encryption.
- Your site has not purchased a SAS/SECURE license for a specific platform.
- You specified encryption algorithms in the local host and the remote host SAS sessions that are incompatible.

See the *SAS/CONNECT User's Guide* or the *SAS/SHARE User's Guide* for more information about the -NETENCRYPT option.

❑ **-NETENCRKEY n**

Set this option in either the local host or the remote host SAS session. It specifies the key length to be used by the encryption algorithm.

Valid values for this option are as follows:

- 128 specifies 1024-bit RSA and 128-bit RC2 and RC4 key algorithms.
- 40 specifies 512-bit RSA and 40-bit RC2 and RC4 key algorithms. Note that specifying 40 is incompatible with DES and TRIPLEDES, and will not allow negotiation of these algorithms.
- 0 no value is set. This is the default.

If extra security is needed, set the -NETENCRKEY option to 128. If you prefer to save CPU cycles, then set the -NETENCRKEY option to 40.

By default, if you try to connect a host that is capable of only a 40-bit key length with a host that is capable of both a 40-bit and a 128-bit key length, then the connection is made using the lesser key length. If both hosts are capable of 128-bit key lengths, then a 128-bit key length is used.

See the *SAS/CONNECT User's Guide* or the *SAS/SHARE User's Guide* for more information about the -NETENCRKEY option.

❑ -NOCLEARTEXT

prevents a sign on from a local host that does not support username and password encryption. This option prevents local hosts in a SAS session that are running releases prior to 6.09E and 6.11 TS040 from signing on to the spawner program. The default is to accept both encrypted and clear-text userids and passwords. This allows local hosts in a SAS session that are running releases prior to 6.09E and 6.11 TS040 to sign on to the z/OS spawner program.

❑ -NOSCRIP

prevents signons from local hosts that use scripts, and allows signons only from local hosts that do not use scripts.

This option requires that the user set security in the local SAS session prior to sign on. For details about setting security (for example, by means of the USER= and PASSWORD= options in an appropriate statement), refer to "Setting Security for SAS/CONNECT and SAS/SHARE" in *Communications Access Methods for SAS/CONNECT 9.2 and SAS/SHARE 9.2*.

If you use the -NOSCRIP option, you must also use the -SASCMD option.

❑ -SASCMD command

specifies the UNIX System Services (USS) shell script that starts a SAS session when you sign on without a script. If the RLINK fileref is not defined in the local host SAS session, then the user is signing on without a script. In this case, the -SASCMD option must be specified.

The script interprets the command arguments and environment variables and builds a TSO command to invoke a SAS session. Please see "Defining the SAS Startup Shell Script" on page 112 for an example shell script.

❑ -SERVICE service-name or port number

specifies the name or TCP/IP port number of the service that the z/OS spawner program uses to listen for incoming requests. This value is identical to the service value in the REMOTE= option that the user specifies at the local host prior to sign on. Because there is no default, you must specify this value.

If the value of the service is given as a name, it must be defined identically in the SERVICES file on both the client and server. For details about the SERVICES files, refer to "Specifying the Remote Node Name"

in *Communications Access Methods for SAS/CONNECT 9.2 and SAS/SHARE 9.2*. If a port number is used, be sure that it does not conflict with any values assigned to other servers, and that it is in the range 1-65535.

Defining the SAS Startup Shell Script

Beginning with SAS 9, the SAS/CONNECT Spawner no longer executes a TSO CLIST directly as the –SASCMD, but instead a UNIX System Services (USS) shell script. The spawner invokes the USS shell script specified in either the SAS/Connect signon script or the –SASCMD spawner option. An example of a –SASCMD follows:

```
-sascmd "/usr/lpp/SAS/spawnsas.sh nosasuser -dmr -noterminal -nosyntaxcheck -comamid tcp"
```

In this example, the command assumes a shell script named `spawnsas.sh` is installed in `/usr/lpp/SAS`. The command specifies the `NOSASUSER` SAS REXX option, and four SAS options: `DMR`, `NOTERMINAL`, `NOSYNTAXCHECK`, and `COMAMID=TCP`.

The shell script will interpret the parameters passed in by the spawner and build a TSO command for invoking SAS. The following is an example shell script. A sample shell script can be found in `'&prefix.BAMISC(SPNCHEL)'`. The sample shell script can be copied to USS using the TSO `OCOPY` or `OPUT` commands. For example:

```
OPUT '&prefix.BAMISC(SPNCHEL)' '/usr/lpp/SAS/spawnsas.sh' text
```

In the example shell script below, the shell script parses a command and interprets environment variables to build a TSO command to start SAS. This command is executed using the USS `/bin/tso` command.

In this example, change the value of `&prefix` to the high-level qualifier of your REXX and `SASRXCFG` libraries that contains the TSO command to start SAS, and `&enc` to the encoding id for the SAS installation.

```
#!/bin/sh
#
# Enable JWT idle timeout
#
export TMOUT=1440
#
# Initialize SAS startup command...
#
cmd="/bin/tso -t %SASTREXX \
-sasrxsysconfig '&prefix.SASRXCFG(REXX&enc)' \
$@"
```

```

#
# Construct REXX parameters from environment variables
# for sessions spawned by the SAS/Connect spawner
#
if [ -n "$INHERIT" ] ; then
    cmd="$cmd -inherit $INHERIT"
fi
if [ -n "$NETENCALG" ] ; then
    cmd="$cmd -netencalg $NETENCALG"
fi
if [ -n "$SASDAEMONPORT" ] ; then
    cmd="$cmd -sasdaemonport $SASDAEMONPORT"
fi
if [ -n "$SASCLIENTPORT" ] ; then
    cmd="$cmd -sasclientport $SASCLIENTPORT"
fi
#
# Set additional environment variables...
# SYSPROC specifies the data set containing the SAS REXX
#
export SYSPROC=&prefix.SASRX
export STEPLIB=
export TSOOUT=
#
# Start SAS
#
exec $cmd

```

Additional TKMVSENV Configuration Note

To ensure responsiveness of SAS spawners and servers to various conditions outside of normal request processing, when idle they periodically awaken to check for requests. The intervals in seconds for these checks are governed by two variables. Generally the default settings should be acceptable. However, if you wish to configure the intervals, set them in the TKMVSENV file. A value of zero for each of these variables means remain idle and only awaken for request processing. Set these variables in the TKMVSENV file as follows:

```

set CONNECTWDWAIT=0
set TCP_POLL_INTERVAL=0

```

See the section "TK Options" in the "Architectural Changes Overview" for more information regarding the specification of the TKMVSENV data set. These variables can be set in both the spawner and in the SAS server sessions it spawns.

Starting/Stopping the SAS/CONNECT Spawner for z/OS Program

To start the spawner, enter the following operator console command:

```
START SPAWNER
```

To stop the spawner, enter the following operator console command:

```
STOP SPAWNER
```

Examples of Starting and Connecting to the z/OS Spawner Program

The following examples illustrate how to start the spawner program and how to connect to it.

Example 1

The following z/OS command starts the spawner program at the remote z/OS host with the service-name spawner and disallows clear-text sign ons from local hosts that use a script.

```
START SPAWNER
```

It uses the following PARMs file:

```
-service spawner  
-nocleartext
```

At a local host, the following statements specify a script file named `tcpmvs.scr` that makes a connection to the spawner program on the system `RMTHOST`, which is listening on the port that is named `spawner`. The value for `REMOTE=` is the host name of the z/OS node, or it can be a macro variable that contains that host name, where the spawner program is running.

```
options comamid=tcp;  
filename rlink '!sasroot\connect\saslink\tcpmvs.scr';  
signon rmthost.spawner;
```

Example 2

In the following examples, the command specified in the `-sascmd` option is a USS (Unix System Services) shell script that invokes a SAS session.. The following z/OS command starts the spawner program at the remote z/OS host.

```
START SPAWNER
```

It uses the following PARMs file:

```
-service spawner
-inheritance
-noscript
-netencralg rc2
-netencralg des
-sascmd "/usr/lpp/SAS/spawnsas.sh nosasuser -dmr -noterminal -nosyntaxcheck
-comamid tcp"
```

At a local host, the TCP/IP access method is used to connect to the remote host named RMTHOST. This must be either the host name of the z/OS node, or a macro variable that contains that host name, where the spawner program is running. The `USER=` option in the `SIGNON` statement prompts the user for a user ID and password when connecting to RMTHOST, on which the z/OS spawner program named spawner runs.

```
options comamid=tcp;
signon rmthost.spawner user=_prompt_ ;
```

SAS/CONNECT to the Same Multi-Process Machine on z/OS

This signon type was formerly documented as “MP CONNECT.” “Multi-Process (MP) CONNECT” exploits a local host’s multi-processor capability, as well as multi-processors across a network, by allowing parallel processing of self-contained tasks and the coordination of all the results into the original SAS session. SAS/CONNECT accomplishes multi-processing by means of asynchronous rsubmits.

When MP CONNECT processing needs to be done on the same host as the local host, a SASCMD signon might be used to initiate one or more “remote” SAS sessions. For z/OS hosts, the SASCMD signon uses the cross-memory services (XMS) access method. The `SASCMD=` option is used for specifying options and values that are passed to a dynamically created DMR session. For details on usage of the MP CONNECT feature, refer to the “SAS/CONNECT User’s Guide.”

System Configuration for SAS/CONNECT to Same Multi-Processor Machine

To implement the XMS access method, which is used by this type of signon, complete the following steps:

STEP 1: Install the SASVXMS Load Module.

This procedure is described on page 151 of the appendix “Appendix Q — Implementing SAS/SHARE Software.” The SAS/CONNECT to same multi-processor machine and SAS/SHARE both require the installation of the SASVXMS module. During the installation of SAS/SHARE, you might choose to rename the SASVXMS0 load module to SASVXMSn, where n is any alpha-numeric character. However, SAS/CONNECT to same multi-processor machine by default looks for the unsuffixed module name of SASVXMS. If you need SAS/SHARE and SAS/CONNECT to use the same suffixed SASVXMSn module, a zap will need to be applied for SAS/CONNECT. See usage note SN-004319.

STEP 2: Define an Anchor Point.

This procedure is described on page 152 of “Appendix Q – Implementing SAS/SHARE Software.”

STEP 3: Install the SAS 9.2 Foundation SVC Routine.

This procedure is described in “Installing the SAS 9.2 SVC Routine” on page 41.

Considerations for Using XMS with SAS/CONNECT to Same Multi-Processor Machine

System administrators should note that the remote (DMR) SAS sessions spawned by MP CONNECT will be connected to their local client z/OS session exclusively with the cross-memory access method.

Like XMS SHARE servers, these DMR server sessions will have their z/OS ASID marked “temporarily non-reusable” by the operating system when their address space terminates. The DMR address space will be terminated in response to the signoff command. The ASID will remain non-reusable until the client address space is terminated -- when the client TSO session logs off, started task ends or batch initiator is drained. Customers should therefore follow IBM recommendations by not spawning DMR sessions from long running address spaces.

In addition, customers should guard against spawning excessive numbers of DMR sessions from a single client session. A complete discussion of how z/OS manages cross memory ASIDs can be found in section 3.9 of *z/OS V1R9.0 MVS Extended Addressability Guide (SA22-7614-05)*. SAS XMS servers, both SHARE and DMR, create non-system LXs for purposes of that discussion.

Note also that the spawning of XMS DMR session utilizes z/OS UNIX System Services, including the BPX1FRK interface (UNIX fork support). If these services are not present in the host z/OS system and usable by the client session, the DMR server creation will fail.

Appendix M — Implementing SAS/GRAPH Software

Understanding the Organization of this Appendix

This appendix is divided into five parts. Each part describes a set of the post-installation tasks that might be necessary to use SAS/GRAPH software at your site. The list is an overview of each part. Use this overview to help locate the information you require.

- Part 1, Accessing the SAS/GRAPH Maps Data Sets
describes how to allocate the MAPS library in your installed CLIST, REXX exec or cataloged procedure. You must perform this task if users at your site need to access maps.
- Part 2, Customizing Devices
provides the information necessary to customize device drivers for your graphic devices.
- Part 3, Setting up and Modifying Device Catalogs
describes how to create or modify device catalog entries in order to customize device driver output to the needs of your site.
- Part 4, Device Help Screens
describes how to use SAS/GRAPH device drivers and to set up system parameters that are required to use certain drivers.
- Part 5, JAVAIMG – Server-side Java graphs
describes how to generate graphs rendered with SAS/GRAPH Java applets on MVS.

Part 1 — Accessing the SAS/GRAPH Maps Data Sets

All installed SAS/GRAPH maps data sets are merged into the common MAPS library by the installation jobs. To enable your users to access the maps data sets when operating under TSO, add the following option specification to your SAS config file (TSOW0):

```
MAPS= '&prefix.MAPS'
```

To access the maps data sets when running under batch, modify your installed SAS cataloged procedure to include the following DD statement for the MAPS library.

```
//MAPS DD DSN=&prefix.MAPS,DISP=SHR
```

Part 2 — Customizing Devices

Setting up a SAS/GRAPH Translate Table

A translate table is only needed if you are using ASCII terminals or attached plotters interactively on ASCII lines. You do not have to worry about translate tables if all of your asynchronous devices are used with 3270-type or 3287-type protocol converters.

If you encounter problems when using SAS/GRAPH software with ASCII terminals or attached plotters interactively on ASCII lines, contact SAS Technical Support.

Using SAS/GRAPH Software with ASCII Terminals and ASCII Terminal Emulators

If you encounter problems when using SAS/GRAPH software with ASCII terminals and PCs running ASCII terminal emulation software, contact SAS Technical Support.

Using SAS/GRAPH Software with ASCII Printers

When using SAS/GRAPH software with ASCII printers, the typical GOPTIONS statement that you need to specify in your SAS/GRAPH program is as follows:

GOPTIONS	DEVICE	= driver-name
	GPROTOCOL	= protocol-module
	GSFNAME	= GSASFILE
	GSFMODE	= REPLACE
	GSFLEN	= 80;

These GOPTIONS tell the device driver to direct the graphics output to the fileref (or DDname) of GSASFILE. To use a FILENAME statement to assign the fileref of GSASFILE to a permanent data set, specify the following:

```
FILENAME GSASFILE 'your-host-file' LRECL=132 RECFM=VB;
```

To use a FILENAME statement to assign the fileref of GSASFILE to a SYSOUT class, specify the following:

```
FILENAME GSASFILE SYSOUT=sysout-class DEST=printer-dest;
```

The following are valid values for the 'protocol-module' value shown above:

SASGPASC	formats the graphics data stream as straight ASCII.
SASGPSTD	formats the graphics data stream as straight EBCDIC.
SASGPAGL	use with an AGILE protocol converter.
SASGPISI	use with an ISI 87 converter (as well as some AGILE converters).
SASGPLCL	Use with converters from Andrew Corporation (as well as older converters from KMW and Local Data).
SASGPAXI	Use with converters from AXIS Corporation.
SASGPVAT	Use with converters from Avatar.
SASGPIDX	Use with converters from IDEX Corporation.
SASGPNET	Use with converters from NetCommander, I-Data, and several others.
SASGPDCA	Use with IRMAprint, IRMAprint2, TEK 4512, and QMS AFPlink converters.
SASGPCAB	Use with converters from Cablenet.
SASGPCHK	Use with a Cherokee protocol converter.
SASGPIOC	Use with converters from I/O Corporation.

For more information about using SAS/GRAPH software with ASCII printers, consult the on-line help for SAS/GRAPH device drivers. If you encounter problems using SAS/GRAPH software with ASCII printers, contact SAS Technical Support.

Installing the Linkable Driver

The Linkable device driver is a special SAS/GRAPH device driver that makes calls to vendor-supplied "CalComp compatible" plotting routines. Before you can use the Linkable driver, parts of it must be compiled and link-edited with your plotting routines.

Devices that typically require the use of the Linkable driver include Xerox 9700, 9790, and 8700 printers. If your site does not have a device that uses vendor-supplied subroutines, you do not have to worry about building a Linkable driver. If you do have devices that use vendor-supplied subroutines, contact SAS Technical Support for more information.

Using SAS/GRAPH Software with IBM 3270-Type Terminals and 3270 Emulators

If you are using a display device that supports mainframe graphics and you are running SAS 9.2 Foundation in the windowing environment, output is automatically displayed on your screen when you run a SAS/GRAPH procedure and it is not necessary to specify a SAS/GRAPH device driver. Note that you must be running your SAS session interactively under TSO to display SAS/GRAPH output on your screen.

If you encounter problems displaying SAS/GRAPH output on the screen of your 3270 display device, the problem is usually that your 3270 display device is not properly configured to support mainframe graphics. To determine the graphics capability of your 3270 display device, invoke the windowing environment and issue the TERMSTAT command from any windowing environment command line. The TERMSTAT command writes device configuration information to the log window. In the SAS log, check that the following two lines appear under the DEVICE FEATURES section:

- Extended Data Stream
- Vector Graphics

If the two lines above do not appear under the DEVICE FEATURES section of the TERMSTAT information, check the following:

- If you have a real IBM graphics terminal, check to be sure that it supports mainframe graphics.
- If you have a PC running 3270 emulation software, make sure that your 3270 emulation software supports host graphics and also that your 3270 host session has been configured to support host graphics.
- Make sure that your 3270 display device is defined to VTAM as a device which supports extended data streams.
- Make sure that the control unit that your 3270 display device is attached to supports (or is configured to support) 3270 graphic data streams.

For more information, consult the on-line help for SAS/GRAPH device drivers. If you encounter problems when using SAS/GRAPH with 3270 display devices, contact SAS Technical Support.

Using SAS/GRAPH Software with IBM 3287, 3268, and 4224 Printers

SAS/GRAPH software supports IBM graphics printers using either native (non-GDDM) device drivers or GDDM device drivers. For detailed information about using SAS/GRAPH software with IBM printers, consult the on-line help for SAS/GRAPH device drivers. If you encounter problems using SAS/GRAPH software with IBM graphics printers, contact SAS Technical Support.

Using SAS/GRAPH Software with GDDM

SAS provides a set of drivers that interface with IBM's GDDM base product and can be used to direct output to any device supported by GDDM. The set of GDDM drivers comes standard with SAS/GRAPH software. You do not need to do anything extra to install the GDDM device drivers. Some of the GDDM device drivers are provided as an alternative to the SAS/GRAPH device drivers. For example, both the GDDMPCG and IBM3179 drivers produce graphics on an IBM 3179 Model G graphics terminal. The advantages of one over the other vary and depend on a particular site's requirements. On the other hand,

some graphics devices require the use of a GDDM driver. These devices include IEEE-attached plotters (IBM 7372, IBM 6180, etc.) and IBM 3800-type laser printers (IBM 3800, 3812, 3820, etc.).

To use any of the GDDM drivers, the GDDM base product (IBM Program Number 5748-XXH) must be installed on your system. The GDDM load library is commonly installed in a system link list library so that the SAS/GRAPH GDDM drivers can load the module `ADMASPT`. `ADMASPT` is GDDM's System Programmer Interface routine. If the GDDM load library is not installed in a system link list library, concatenate it to the `STEPLIB DD` statement (if batch) or use the `LOAD` parameter in the SAS CLIST (if running interactively under TSO).

For more information about using SAS/GRAPH software with GDDM, consult the on-line help for SAS/GRAPH device drivers. If you encounter problems using SAS/GRAPH software with GDDM, contact SAS Technical Support.

Part 3 — Setting Up and Modifying Device Catalogs

Note: The tasks described in this section are optional.

After installing SAS/GRAPH software, you might need to create or modify device catalog entries in order to customize device driver output to your site's needs. This section gives a brief explanation of device catalogs, and how to handle situations where catalog entries might need to be modified. For complete details on managing device catalogs, refer to "The GDEVICE Procedure," in the on-line help screens for SAS/Graph Software.

How Device Catalogs Are Used

In SAS/GRAPH 9.2 software, when you specify the name of a graphics device driver with the `DEVICE=` or `TARGET=` options, or when prompted, the name you specify corresponds to an entry in a device catalog. Device catalog entries contain default characteristics used by the driver. For example, the parameters can determine details such as graph size, picture orientation, default colors, and whether to send the graphics output directly to a device or store it in a file. A device catalog entry can also be set up to control the attributes of a graphics stream file or to execute the necessary host commands to send graphics output directly to the device. This feature enables you to develop applications that do not require the end user to specify special `GOPTIONS` or issue host commands to produce hardcopy output. You can change the characteristics used by a driver either by modifying its entry in the device catalog, or by specifying `GOPTIONS` that override settings in the catalog.

For example, if you specify `DEVICE=HP7550`, the SAS/GRAPH procedure attempts to find an entry named `HP7550` in available device catalogs. The parameters found in the entry (such as the default graph size, graph orientation, or where output is sent) are used in generating the graph. If you want to change the way the driver produces output, you can use the `GDEVICE` procedure to modify parameters in the catalog entry, or you can override them with a `GOPTIONS` statement. In general, if you want to change defaults for a single session or job, you can use a `GOPTIONS` statement. If you want to permanently change the default parameters used by a driver, you can create a new device entry or modify an existing one.

A device catalog supplied by SAS, `SASHELP.DEVICES`, is installed and made available as part of the overall SAS/GRAPH installation process. This catalog contains over 300 entries, covering each graphics device and model that SAS/GRAPH software supports. Individual users or groups can also create their own device catalogs. These are given the names `GDEVICE0.DEVICES`, `GDEVICE1.DEVICES` and so on, through `GDEVICE9.DEVICES`. When a driver name is specified in a SAS program, SAS 9.2 Foundation looks for the corresponding entry in `GDEVICE0.DEVICES`, `GDEVICE1.DEVICES`, etc. If the entry is not found in any of the user catalogs (or the catalogs do not exist), the supplied catalog, `SASHELP.DEVICES`, is searched.

How and When to Modify Catalog Entries

If you need to make changes to a device entry, how you do so depends on whether the changes affect an individual user or most or all users at a site. If the change affects only one user, that user should create his or her own device catalog (`GDEVICE0.DEVICES`), copy the device entry from `SASHELP.DEVICES`, and make the changes to the entry in `GDEVICE0.DEVICES`. Note that `GDEVICE0.DEVICES`, being a “personal” catalog, is usually a different catalog for each user. If the modification affects a large number of users, the SAS Installation Representative can make modifications to an entry in `SASHELP.DEVICES`.

The following guidelines should be used when creating or modifying device entries:

- Only the SAS Installation Representative should add or modify entries in `SASHELP.DEVICES`. End users should not have update access to `SASHELP.DEVICES`. If individual users need to make modifications, they should create their own `GDEVICE0.DEVICES` catalog.
- If a catalog entry needs to be modified, create a new entry (with a different name) and modify the new entry. By renaming modified entries, users are ensured that the original entries supplied in the `SASHELP.DEVICES` catalog use default settings.
- Any options specified in a `GOPTIONS` statement override equivalent parameters in device catalogs. If a parameter needs to be changed only for a single session, it is probably easier to use a `GOPTIONS` statement than to create a new device entry.

Examples

The following examples illustrate how to use device catalog parameters to create graphics stream files or to spool output directly to a hardcopy device. The examples first illustrate `GOPTIONS` and `FILENAME` or host statements that can be used to produce output, and then show how equivalent parameters can be specified in device entries, eliminating the need for the statements in the end users’ programs. For complete details on managing device catalogs, refer to Chapter 25, “The GDEVICE Procedure,” in the *SAS/GRAPH 9.2 Reference*.

Creating a Graphics Stream File

Suppose you want to use the HP7550 driver to create a graphics stream file of HPGL commands to be transferred to another computer or application. To create the file using `GOPTIONS` and `FILENAME` statements, you can use the following statements:

```
/* define fileref for graphics stream file */
  filename gsasfile 'my.gsf.file';
  /* specify device driver, fileref, mode, and record length */
  goptions dev=hp7550 gaccess=gsasfile gsfmode=replace gsflen=80;
```

You can achieve the same results by creating your own driver with the `GDEVICE` procedure and specifying host file options. The following display shows the Host File Options window for the modified device entry `MYHP7550`. You can enter these values using `GDEVICE` windows or with line-mode `GDEVICE` statements.

```
GDEVICE: Host File Options _____
Command === >

          Catalog:  GDEVICE0.DEVICES      Entry:  MYHP7550

Gaccess:
SASGASTD>my.gsf.file
Gsfname:  _____      Gsfmode:  REPLACE      GSflen:  80

Trantab:  _____      Devmap:  _____      Devtype:  DISK

Gprotocol:  _____

Host file options:
_____
_____

* Close file at end of driver or procedure termination
o Close file at end of each graph

_____ ZOOM  _R _____
```

Notice that the `DEVTYPE` field specifies `DISK`. This prevents the driver from sending output to the terminal. In addition, notice that the `GACCESS` field contains the complete name of the external file (without surrounding quotes). If the file does not already exist, you can have the driver allocate it by specifying the `DISP=` and `SPACE=` parameters in the `Host File Options` field.

Spooling Directly to a Graphics Device

Suppose you want to use the HPLJ300 driver and send the output directly to a Hewlett-Packard LaserJet printer attached to an AGILE 6287 protocol converter. Your site has system software such as VPS that enables you to define the printer as a JES destination with a `SYSOUT` class of `A`, and a `DEST` of `HPPRINT`.

The following `GOPTIONS` and `FILENAME` statements are used to send graphics output to the LaserJet printer.

```
/* define fileref and JES parameters for graphics stream file */
filename gsf file sysout=a dest=hpprint;

/* specify device driver, fileref for GSF, */
/* protocol converter, and record length */
goptions dev=hplj300 gaccess=gsf gprotocol=sasgpagl gsflen=128;
```

You can achieve the same results by creating your own driver with the `GDEVICE` procedure and specifying host file options. The following display shows the `Host File Options` window for the modified device entry `MYHP300`. You can enter these values using `GDEVICE` windows or with line-mode `GDEVICE` statements.

```

GDEVICE: Host File Options
Command===>

          Catalog:  GDEVICE0.DEVICES      Entry:  MYHP300

Gaccess:  _____

Gsfname:  _____      Gsfmode:  REPLACE      GSflen:      0
Trantab:  _____      Devmap:  _____      Devtype:  PRINTER

Gprotocol: SASGPAGL

Host file options:
SYSOUT=A DEST=HPPRINT
_____

* Close file at end of driver or procedure termination
o Close file at end of each graph

_____ ZOOM  - R_____

```

When the preceding parameters are specified, SAS/GRAPH software dynamically allocates a `SYSOUT` file with a `SYSOUT` class of `A` and a destination of `HPPRINT`, and directs the driver output to that file.

Note: Because a temporary spool file is dynamically allocated, it is not necessary to specify `GACCESS=GSASFILE`.

Part 4 — Device HELP Screens

The device HELP screens for SAS/GRAPH 9.2 contain information on setting up system parameters required to use certain drivers and how to use SAS/Graph device drivers. If you are using SAS 9.2 Foundation interactively on a full-screen terminal, you can also obtain details on using graphics devices by first making sure that you have enabled `PMENUS`, and then selecting Help from the pulldown menu. Then, select SAS System Help, then Help on SAS Software Products, and then SAS/GRAPH. On the “About SAS/GRAPH” screen, select Using SAS/GRAPH Software and then Using Graphics Devices.

Part 5 — JAVAIMG – Server-side Java Graphs

In SAS 9.2 Foundation, you can now generate graphs rendered with SAS/GRAPH Java applets on MVS. For more information, see the section on JAVAIMG in the *SAS/GRAPH Companion*.

The following sample SAS program generates an HTML file (`sampzos.html`) and JAVAIMG chart (`sampzos.png`) and writes them to HFS files. The HTML file with embedded image can be viewed with a Web browser.

Note: Before running this example, please see “Configuring SAS Software for Use with the Java Platform” on page 51.

```
ods listing close;
```

```
ods html path='/u/mvmdir/public_html'  
         gpath='/u/mvmdir/public_html' (url=none)  
         body='sampzos.html';
```

```
goptions device=javaimg;
```

```
title1 "JAVAIMG Example on z/OS";
```

```
proc gchart data=sashelp.class;  
  vbar age /  
  name='sampzos';  
run;
```

```
quit;
```

```
ods html close;
```

To view this sample HTML file, enter the following address in your Web browser (where *zoshost* is the address of the z/OS machine):

```
http://zoshost/~mvmdir/sampzos.html
```

Appendix N — Post-Installation Configuration for SAS/IntrNet Software

This chapter has information for your SAS/IntrNet installation. It will help you install, configure, and test your SAS/IntrNet components.

The procedures for installing SAS software using the SAS Deployment Wizard are described in other documentation and not available from this chapter. Furthermore, the installation of your Web server is your responsibility and not described in SAS documentation.

When the SAS/IntrNet software has been installed, configured and tested using the procedures described in this chapter, review the latest version of the SAS/IntrNet product documentation online at <http://support.sas.com/documentation/onlinedoc/IntrNet/index.html>. The “What's New” page at this Web site lists any recent changes to the product or documentation.

SAS/IntrNet software has the following prerequisite on z/OS:

- SAS/IntrNet requires that SAS be configured for communication with TCP/IP. Please ensure that the steps in the section “System Configuration for Using SAS with TCP/IP” on page 21 have been reviewed and completed.

Overview

All SAS/IntrNet installations are made up of two components:

1. The SAS/IntrNet server (also referred to as the Application Server). This is where SAS Foundation is installed.
2. CGI Tools (also referred to as the Broker). This is where the `broker.cfg` file and its supporting files are installed.

When you install SAS/IntrNet, choose between two installation configurations:

Type A - The SAS/IntrNet server and CGI Tools components are both installed on the same system machine. The Web server **must** be installed before starting the SAS installation.

Type B -The SAS/IntrNet server component is installed on one system machine and the CGI Tools component is installed on a different system machine. The Web server **must** be installed on the CGI Tools system prior to installing CGI Tools.

Type A and Type B require different installation steps:

Type A Installation Steps	Type B Installation Steps
Confirm that the Web server software (IIS, Apache etc.) is on the same server as your SAS/IntrNet system.	Confirm that the Web server software (IIS, Apache, etc.) is on the machine that you will install the CGI Tools to.
Install your SAS products including “CGI Tools for the Web Server” and “SAS/GRAPH Java Applets.”	On your application server system, start your SAS installation. Uncheck “CGI Tools for the Web Server” in the “Select Products to Install” menu.

Type A Installation Steps	Type B Installation Steps
	<p>On your Web server system machine, start your SAS installation. Uncheck all products except “CGI Tools for the Web Server” and “SAS/GRAPH Java Applets” in the “Select Products to Install” menu.</p> <p>You can optionally check the IntrNet Monitor or Connect Drivers</p>
Test the Broker	Test the Broker
Configure a Socket Service	Configure a Socket Service
Start the Socket Service	Start the Socket Service
Test the Socket Service	Test the Socket Service

The steps are described more thoroughly in the sections that follow.

Installing and Configuring SAS/IntrNet Software

Install Your Web Server Software

Refer to your Web server’s documentation for its installation procedures.

Install Your SAS Software

Refer to your *QuickStart Guide* for a description of how to start your SAS software installation.

If you are performing a Type A installation (as described in the "Overview" above), confirm that your Web server software is installed before starting your SAS software installation.

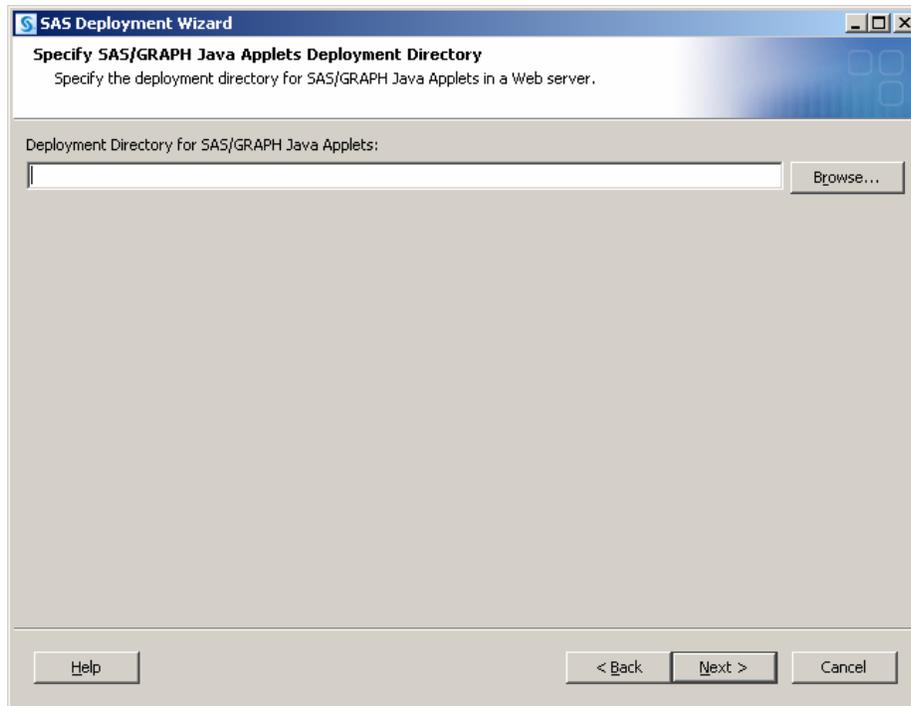
If you are performing a Type B installation, do **both** of the following:

- Install the SAS software on the SAS System side.
- Start the SAS software install on the Web server and check “CGI Tools for the Web Server” and “SAS/GRAPH Java Applets” in the **Select Products to Install** dialog. SAS/IntrNet Monitor and SAS/CONNECT Driver for Java are optional selections. Uncheck everything else.

CGI Tools Installation Dialogs on PC and UNIX

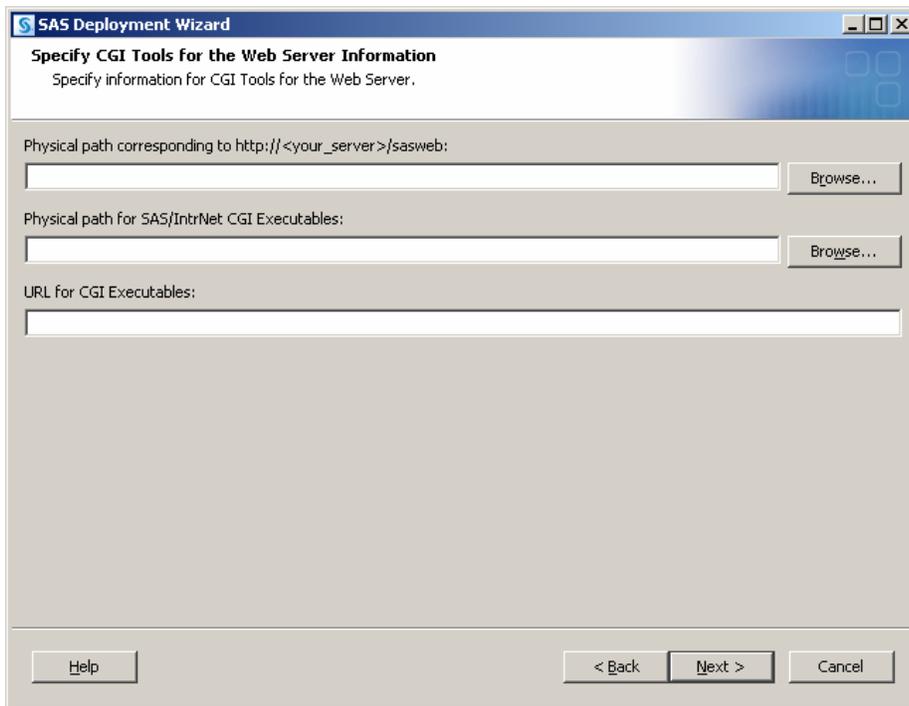
The following screens appear for CGI Tools for the Web Server and SAS/GRAPH Java Applets for all installations. Click **Help** on any dialog for information about the fields.

Customary entries are documented following each screen shown below. Customize the entries according to your environment.

**Deployment Directory for SAS/GRAPH Java Applets:**

This is part of the SAS/GRAPH Java applets install. The applets can be installed in any valid folder location.

This location will be available to the CGI Tools installation, which will copy the graph applets to the /sasweb/graph directory on the Web server.



The following are examples of common entries for popular Web servers. Customize your entries according to your own Web server environment. These fields will tell SAS where your Web server software is located.

Physical path corresponding to `http://<your_server>/sasweb`:

- IIS: `C:\Inetpub\wwwroot\sasweb`
- Apache (Windows): `C:\program files\Apache Software Foundation\Apache2.2\htdocs\sasweb`
- Apache (UNIX): `/usr/local/apache2/htdocs/sasweb`

Physical path for SAS/IntrNet CGI Executables:

- IIS: `C:\Inetpub\scripts`
- Apache (Windows): `C:\program files\Apache Software Foundation\Apache2.2\cgi-bin`
- Apache (UNIX): `/usr/local/apache2/cgi-bin`

URL for CGI Executables:

- IIS: `http://<web_servername>/scripts`
Example: `http://abcserver.comp.com/scripts`
- Apache (Windows): `http:// <web_servername>/cgi-bin`
Example: `http://abcserver.comp.com/cgi-bin`
- Apache (UNIX): `http://<web_servername>/cgi-bin`
Example: `http://abcserver.comp.com/cgi-bin`

SAS Deployment Wizard

Specify CGI Tools for the Web Server Additional Settings
Specify additional settings for CGI Tools for the Web Server.

Name of the Service Administrator:

Email Address of the Service Administrator:

DNS Name or IP Address of Application Server Host:

TCP Port Number for Application Server:

Buttons: Help, < Back, Next >, Cancel

Note that your entries for this dialog are added to the `broker.cfg` file. The `broker.cfg` file is a text file that can be edited after the installation is complete.

Name of the Service Administrator:

(optional) Enter the name of the administrator (for example, John Doe).

Email Address of the Service Administrator:

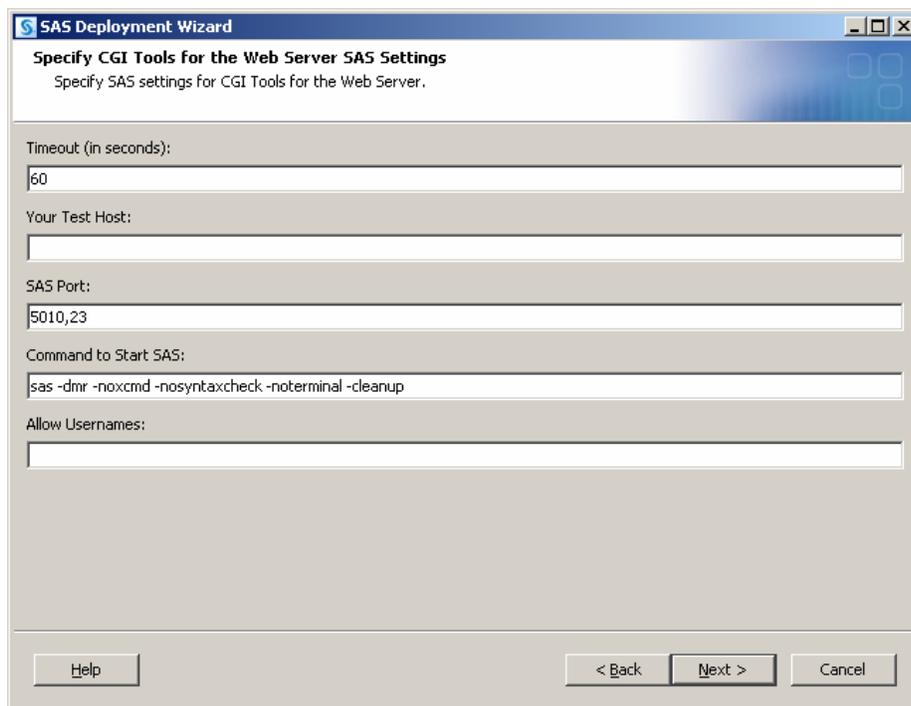
(optional) Enter the e-mail address of the administrator (for example, NetAdmin@comp.com).

DNS Name or IP Address of Application Server Host:

Enter the DNS name or IP address of the application server host where SAS Foundation is located.

TCP Port Number for Application Server:

The customary default port number is *5001*, but you can use any valid available port on your system between 256 – 65535.



Click **Next** at this dialog unless you are using JConnect. If you are using JConnect, enter values that are valid for your site.

CGI Tools Installation on z/OS

The CGI Tools are installed and configured as described in the following steps.

The USSUNTAR tool must be run before you can continue. This will copy CGI Tools to the hierarchical file system. Please refer to the `readme.txt` located at the following location where SASHOME is the directory into which USSUNTAR installed SAS:

```
SASHOME/CGIToolsForTheWebServer/9.2/websrv
```

The `readme.txt` file includes instructions on how to deploy CGI Tools to a z/OS Web server.

After following the instructions in the `readme.txt` file, copy the Load Manager executable from

```
/SASHOME/CGIToolsForTheWebServer/9.2/IntrNet9/tools/
```

to

```
/<Web-server-root>/sasweb/IntrNet9/tools/
```

Installing CGI Tools and SAS Foundation on Machines with Different Operating Systems

Your SAS Foundation system's operating system might be different than your CGI Tools system's operating system. For example, your SAS Foundation might be installed on a z/OS system and your CGI Tools might be installed on a UNIX system. The CGI Tools installation from the SAS Deployment Wizard will detect the destination operating system and install the appropriate operating system-specific software.

There are two methods to make the SAS Software Depot available to the installer on the destination CGI Tools system. The method you choose is dependent on the facilities available at your site. To access a SAS Software Depot on the destination CGI Tools system, do one of the following:

1. Launch the set-up from a SAS Software Depot residing on a remote system. You might need to use a cross-platform file access method, such as NFS or SAMBA, to connect the two systems.
2. Create media from an existing depot using the SAS Deployment Wizard and use that media on the host machine. This process is described more thoroughly in the *SAS Deployment Wizard Users Guide*, available from Install Center
(<http://support.sas.com/documentation/installcenter/92/documents/index.html>).

Note: SAS/IntrNet operation requires TCP/IP connectivity between the SAS Foundation system and the CGI Tools system regardless of which operating systems these components are installed on.

Test the Web Server

To determine if the Web server is running, launch the Web server's browser and enter `http://localhost`. This will return a Web page if the Web server is running.

If you do not receive a Web page, you must debug or reinstall your Web server before continuing.

Enabling the SAS/GRAPH Java Applets Documentation

If you want to enable online help for the SAS/GRAPH Java Applets, you must modify your Web server configuration. Add the following directives to your `httpd.conf` file:

```
AddEncoding    .ascii          8bit          1.0    # stored in ASCII
Pass    /sasweb/graph/doc/**/*.htm <install-path>/sasweb/graph/doc/**/*.htm.ascii
```

Test the Application Broker

To verify that CGI Tools was installed correctly and can access the `broker.cfg` file, point your Web browser to the following URL:

Windows:

IIS: `http://<web_servername>/scripts/broker.exe`

Apache: `http://<web_servername>/cgi-bin/broker.exe`

Other hosts:

`http://your_webserver/cgi-bin/broker`

Replace `your_webserver` with the name of the Web server. The URL path might also need to be changed if you installed CGI Tools to a different directory. You should see a Web page similar to the following:

SAS/IntrNet Application Dispatcher

Application Broker Version 9.2 (Build 1494)

Application Dispatcher Administration

SAS/IntrNet Samples

SAS/IntrNet Documentation - requires Internet access

If you do not receive this page, you must debug your Web server installation before continuing. Verify that your Web server is enabled for CGI execution in the directory where you installed the Application Broker (`broker.exe` and `broker.cfg` files). This directory was determined by what was entered for **Physical path for SAS/IntrNet CGI Executables** during the CGI Tools installation.

Configure a Socket Service

A **default** Application Dispatcher service must be defined to run the sample programs included with SAS/IntrNet. An Application Dispatcher service is a collection of one or more server tasks that will execute SAS/IntrNet programs. The default service described below is a simple server service with one server task that may be used to execute samples and is a convenient way to begin developing your own SAS/IntrNet applications. It is not required, but it is recommended for most installations.

A TCP/IP port (number) or service definition (name) must be reserved for your default service before you create the service. Note that TCP/IP ports or service definitions are not the same as Application Dispatcher service definitions. A single Application Dispatcher service may use one, many, or no TCP/IP ports or service definitions. For the default service definition you will need just one TCP/IP port. Consult your system administrator or check your TCP/IP services definition file (ETC.SERVICES and also /etc/services) to find an available port.

Services are configured by a batch job found the &prefix.CNTL(INETCFG) data set that was created during the first step of your SAS installation. To create a default service:

1. Edit the parameter file, member INETEDTP in the &prefix.CNTL data set, as described in the following steps. The INETEDTP member contains the parameters necessary for creating a service. Editing instructions are provided in the comments in INETEDTP, and the default values should be changed to the values required for the service being created.
2. Specify the name of the Application Dispatcher service that you are creating. Locate the line containing ISVC= and verify the service name is DEFAULT.
3. Specify the default service is a socket service. Uncomment the %SOCKETTYP line containing ISVCTYP=. Make sure the %POOLTYP line is commented out by placing an asterisk (*) in the first column.
4. Specify the TCP/IP port number or service name for the server. Change the value 5001 to the correct port number or service name for your server.
5. Review and update any other remaining parameters if necessary.
 - ADMINPW=YOURPASSWORD
Uncomment and supply password to enable administrator password for this service. Do not leave a blank between ADMINPW= and the supplied password. An administrator password is not required and can be added later if you chose to skip this step.
 - I\$SAMPLE=&prefix.SAMPLE
Location of SAS/IntrNet sample program PDS.
 - I\$SAMPSIO=&prefix.SAMPSIO
Location of SAS/IntrNet sample SAS Library.
 - INETENTRY=SAS
Set to installation default entry.

- ❑ INETWORK=500,200
Primary and secondary allocations for the PROC WORK Data set.
6. Save and close INETEDTP.
 7. Edit the INETCFG job to verify the job header information and the name of the service you are defining. The service name in the JCL should match the value you supplied for ISVC in INETEDTP. If you make changes, be sure to save them. Do NOT change SASEDITP to INETEDTP. This filename refers to your original SAS installation parameters file.
 8. Submit the INETCFG job for processing. The INETCFG job will submit another job (INETCFGGA). Verify that both jobs complete with a return code of 0. If the jobs complete successfully, the data sets and members necessary for running your service.

If the INETCFG job fails, examine the messages and SYSPRINT output for error messages. If the following message is received:

```
ERROR: THIS REPLACEMENT CAUSES RESULT TO EXCEED OUTPUT LRECL
```

You might have supplied a pathname in one of the INETEDTP parameters that is too long. Try shortening this pathname and rerun INETCFG.

Note: *Before running INETCFG again, any data sets created by the previous failure of INETCFG must be deleted. To find these data sets, look at the &prefix determined by the original SAS installation.*

For example, if SAS was installed with the &prefix name SYS.SAS and the failed INETCFG was trying to create the DEFAULT service, then delete all data sets beginning with the name &prefix SYS.SAS.WEB.DEFAULT before running INETCFG again.

9. The configuration utility creates a server root in a partitioned data set (PDS) named &prefix.WEB.DEFAULT, where &prefix is the data set &prefix that was supplied during the SAS installation. The PDS contains any JCL procedures and server startup code required for starting the service.

Verify that the following members have been created:

- ❑ APSTRn
contains the JCL necessary to run the corresponding @APSTXn member as a started task. This member should be moved to a started task library and enable it as started tasks.
- ❑ @APSTXn
contains the SAS code that invokes the server. The JCL calls this file in the corresponding APSTRn member. These SAS programs must remain in the PDS where they were created.

In addition to the server root PDS, the configuration utility creates an empty PDS named &prefix.WEB.DEFAULT.TDIR. The default service will use a member of this PDS named TBLIB1 as its scratch SAS data library.

10. The permissions for the data sets created above must be modified so the server can write to them as necessary. To modify the permissions, create a special security profile that applies to all the data sets in this service (&prefix.WEB.DEFAULT.*). The security system profile should also grant write access to the user ID of the Application Server.

11. If you installed CGI Tools for the Web Server on your z/OS system, complete this step.

The Application Broker must know about this service so that you can access it. Edit the Application Broker configuration file (named broker.cfg) on your Web server and customize the default service definition block for your installation. The service definition block for a socket service might look like this:

```
# This service contains one server (port 5001) on yourserv.yyy.com.
SocketService default "Reuse existing session"
    ServiceAdmin "Your Name"
    ServiceAdminMail "yourname@yyy.com"
    Server yourserv.yyy.com
    Port 5001
    FullDuplex    True
```

Starting the Socket Service

As stated above, the APSTRtn file for the default Application Dispatcher service should be moved from the file &prefix.WEB.DEFAULT to a started task library and enabled as a started task. To start the default service, issue a START command from the system console.

After initialization, the Application Server will pause. This indicates the server has begun waiting for Application Dispatcher requests from the Application Broker, and everything is functioning properly.

Testing the Socket Service

1. To make sure that the service was installed and started correctly, point your Web browser to this URL:

Windows:

IIS: http://<web_servername>/scripts/broker.exe

Apache: http://<web_servername>/cgi-bin/broker.exe

Other hosts:

http://your_webserver/cgi-bin/broker

Replace `your_webserver` with the name of the Web server. The URL path might also need to be changed if you installed CGI Tools to a different directory. You should see a Web page similar to the following:

SAS/IntrNet Application Dispatcher

Application Broker Version 9.2 (Build 1494)

Application Dispatcher Administration
SAS/IntrNet Samples
SAS/IntrNet Documentation - requires Internet access

2. Click on the **Application Dispatcher Administration** link to see if the Application Broker can read the `broker.cfg` file. The Application Dispatcher Services Web page should open.
3. Verify connectivity between the Application Server and the Web server. Click on the **Application Dispatcher Administration** link and then click on the **ping** link under **SocketService default** heading. If the ping is successful, you should see:

Ping. The Application Server <hostname>:<port_number> is functioning properly.

4. To complete installation testing, type this URL in your browser address line:

Windows:

IIS:

```
http://your_webserver/scripts/broker.exe?_service=default&_program=sample.webhello.sas
```

Apache:

```
http://your_webserver/cgi-bin/broker.exe?_service=default&_program=sample.webhello.sas
```

Other hosts:

```
http://your_webserver/cgi-bin/broker?_service=default&_program=sample.webhello.sas
```

You should see the string "Hello World!" in large bold type in your browser. If you do not, add the debug option to create a log:

Windows:

IIS:

```
http://your_webserver/scripts/broker.exe?_service=default&_program=sample.webhello.sas&_debug=131
```

Apache:

```
http://your_webserver/cgi-bin/broker.exe?_service=default&_program=sample.webhello.sas&_debug=131
```

Other hosts:

```
http://yourWebserver/cgi-bin/broker?_service=default&_program=sample.webhello.sas&_debug=131
```

Save the log screen on the browser for SAS Technical Support.

Configure Additional Services

This chapter only describes how to configure a simple default Application Dispatcher service. There are many reasons you may want to configure additional services, including segregating applications by security or performance requirements and implementing more scalable servers. See the “Using Services” section of the SAS/IntrNet Application Dispatcher documentation at

<http://support.sas.com/documentation/onlinedoc/intrnet/index.html> for information on configuring additional services, using the Load Manager, and adding pool services.

Appendix O — Post-Installation Setup for SAS/SECURE Software

SAS/SECURE software includes client components that you can use to create non-SAS 9.2 Foundation client applications which communicate with a SAS server in a secure environment. To use encryption between a non-SAS 9.2 Foundation client and a SAS Server with SAS/SECURE software licensed, you must install the SAS/SECURE client components on the client machine.

SAS/SECURE Client for Windows

The `secwin.exe` executable installs the files necessary for the IOM Bridge for COM to use the CryptoAPI algorithms. It also contains a TAR and ZIP file that is used to develop Java clients that utilize the encryption support.

SAS/SECURE Client for Java

The SAS/SECURE client for Java provides encryption support for Java applications. You can incorporate this support into applications that are written using the following components:

- SAS/SHARE driver for JDBC
- SAS/CONNECT driver for Java
- IOM Bridge for Java

Client Components

The SAS/SECURE client components are available on the SAS/SECURE CD included with your SAS Software distribution.

Appendix P — Implementing SAS/SESSION Software

Introduction

SAS/SESSION software enables terminal users connected to the Customer Information Control System (CICS) to communicate with SAS 9.2 Foundation in a z/OS environment. In reality, the user communicates with SAS 9.2 Foundation running in an APPC/MVS initiator. SAS 9.2 Foundation uses VTAM as the communication access method. Figure 1 illustrates the relationship among the various components.

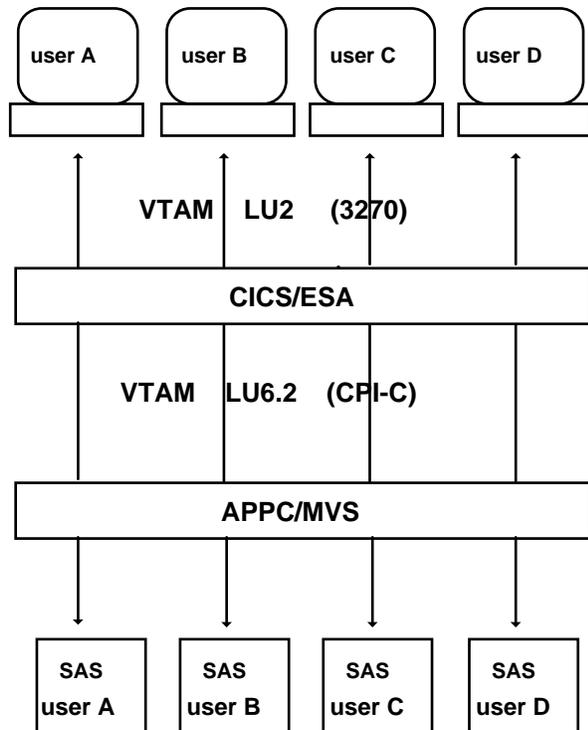


Figure 1

Installing the SAS/SESSION interface to SAS 9.2 Foundation consists of the following:

- defining the interface to VTAM
- defining the interface to APPC/MVS
- defining the interface to CICS.

These topics are covered in the next three sections. The discussions assume that Base SAS software, CICS, and APPC/MVS have already been installed.

Program names and argument values shown throughout this document serve as examples only. You can modify them to fit your naming conventions.

For more information on defining the interface, consult the following manuals:

- ❑ CICS Intercommunication Guide
- ❑ CICS Resource Definition Guide
- ❑ MVS Planning: APPC/MVS Management

Defining SAS/SESSION to the VTAM System

To define SAS/SESSION to VTAM requires two steps:

- ❑ Define the two VTAM applications needed by the interface
- ❑ Define an LU Type 6.2 entry in the VTAM logon mode table.

Define the VTAM Applications

Two VTAM applications need to be defined (or modified):

- ❑ SASSESS, to access SAS 9.2 Foundation through APPC/MVS
- ❑ MVSCICS, the CICS system application.

Note: The application names SASSESS and MVSCICS are examples for the purpose of discussion only. Contact your systems programmer to identify the correct names for your installation.

Use the VTAM APPL macro to define the applications. The VTAM application definition table contains an APPL macro expansion for each application to be used in a VTAM environment. You will need to add (or change) some parameters for the SASSESS and the MVSCICS applications:

```
SASSESS    APPL    APPC=YES , SRBEXIT=YES , SECACPT=ALREADYV , VERIFY=NONE ,    *
              DMINWNL=0 , DMINWNR=10 , DSESLIM=10

MVSCICS    APPL    AUTH=( ACQ ) , EAS=10 , APPC=NO , PARSESS=YES ,    *
              ACBNAME=MVSCICS
```

Examples of these APPL definitions are in member SESSAPPL of the &prefix.SEMISC data set.

Define the VTAM Logon Mode

The VTAM logon mode table contains various protocol definitions for use by applications within the VTAM system. SAS/SESSION uses an Advanced Program to Program Communication (APPC) logmode entry. If this type of entry already exists, the interface can use it.

Otherwise, use the following MODEENT macro. You can include the MODEENT macro in the existing VTAM logon mode table.

```
SASCLU62 MODEENT LOGMODE=SASCLU62,*
              TYPE=X'00' ,    *
              FMPROF=X'13' ,    *
              TSPROF=X'07' ,    *
              PRIPROT=X'B0' ,    *
              SECPROT=X'B0' ,    *
              COMPROT=X'50B1' ,    *
              PSERVIC=X'0602000000000000000002C00'
```

A copy of this mode table entry is in the &prefix.SEMISC data set, member SESSMODE.

Note: The mode name must match the value specified for the Modename parameter in the CICS SESSION resource. See "Defining SAS/SESSION to CICS" on page 143.

Defining SAS/SESSION to APPC/MVS

To define SAS/SESSION to the APPC/MVS system, you need to modify the SYS1.PARMLIB members for APPC/MVS initialization. These members are APPCPMxx and ASCHPMxx, where the xx is the two-character suffix of the specific members used by your system. APPCPMxx defines the logical unit that corresponds to the VTAM application defined for APPC/MVS (SASSESS). The following example is in &prefix.SEMISC, member SESSAPPM:

```
LUADD
    ACBNAME ( SASSESS )
TPDATA ( SYS1 . APPCTP )
```

Member ASCHPMxx defines a class of initiators for executing SAS 9.2 Foundation. The definition specifies the number of instances that are available, as in this example from &prefix.SEMISC, member SESSASPM:

```
CLASSADD CLASSNAME(SASSESS) MIN(1) MAX(10) RESPGOAL(1)
```

To invoke SAS in an initiator, add the following transaction program profile definition to SYS1.APPCTP. The TPNAME parameter must correspond to the TPNAME specified in the CICS PARTNER resource. See "Defining SAS/SESSION to CICS" on page 143. The CLASS parameter value (SASSESS) corresponds to that for the CLASSADD statement in the ASCHPMxx parmlib member.

```
TPADD
TPNAME ( SAS_SESSION )
TPSCHED_DELIMITER ( DLM1 )
CLASS ( SASSESS )
KEEP_MESSAGE_LOG ( ERROR )
JCL_DELIMITER ( DLM2 )
//SASSESS JOB ( ) , ' SAS / SESSION ( TM ) '
//PROCLIB JCLLIB ORDER = ( SYS2 . PROCLIB ) LIB CONTAINING SAS PROC
//CRTESUSR EXEC PGM = IEFBR14
//SASUSER DD DISP = ( MOD , CATLG ) , DSN = &SYSUID . . SASSESS . SASUSER ,
// UNIT = DISK , SPACE = ( CYL , ( 2 , 1 ) )
//SASAUTOS DD DISP = ( MOD , CATLG ) , DSN = &SYSUID . . SASSESS . SASAUTOS ,
// UNIT = DISK , SPACE = ( CYL , ( 2 , 1 ) )
//CONFIG DD DISP = ( MOD , CATLG ) , DSN = &SYSUID . . SASSESS . CONFIG ,
// UNIT = DISK , SPACE = ( TRK , ( 0 , 1 ) ) ,
// DCB = ( DSORG = PS , RECFM = FB , LRECL = 80 , BLKSIZE = 3120 )
//SASEXEC DD DISP = ( MOD , CATLG ) , DSN = &SYSUID . . SASSESS . SASEXEC ,
// UNIT = DISK , SPACE = ( TRK , ( 0 , 1 ) ) ,
// DCB = ( DSORG = PS , RECFM = FB , LRECL = 80 , BLKSIZE = 3120 )
//SAS EXEC SAS , OPTIONS = ' SESSION ' ,
```

```
// SASAUTO=&SYSUID..SASSESS.SASAUTOS,
// CONFIG=&SYSUID..SASSESS.CONFIG
//SASUSER DD DISP=OLD,DSN=&SYSUID..SASSESS.SASUSER
//SASEXEC DD DISP=SHR,DSN=&SYSUID..SASSESS.SASEXEC
DLM2
DLM1
```

Note: Setting `KEEP_MESSAGE_LOG(ERROR)` generates a message data set when a non-zero return code is returned to CICS. Refer to the appropriate IBM documentation for more information on the `MESSAGE_DATA_SET` and the `KEEP_MESSAGE_LOG` parameters.

An example job to update your TP profile data set with this profile is in the `&prefix.SEMISC` data set, member `SESSPROF`.

Security Considerations

Since SAS 9.2 Foundation executes in an APPC/MVS address space rather than under CICS, it acquires its own security environment. APPC/MVS defines this environment when CICS requests allocation of the transaction program. To create the environment, APPC/MVS uses the user ID that CICS passes. By default, CICS passes no user ID, so in this case, all input data sets used by APPC/MVS must have a universal access of read, and all output data sets must have a universal access of write.

In order for CICS to pass a user ID to APPC/MVS, you must establish a conversation security level of *already verified*. You can specify this using the `SECACPT=ALREADYV` parameter of the VTAM APPL definition for the APPC/MVS logical unit (`SASSESS`). Alternatively, if you define RACF session security between the CICS and APPC/MVS logical units, you can specify `CONVSEC(ALREADYV)` on the RACF APPCLU definition for the APPC/MVS LU.

If the security level is already verified and users do not sign on to CICS using the CESN transaction, CICS passes its default user ID on TP allocate requests. In this case, SAS 9.2 Foundation running under APPC/MVS has the same security as the CICS system and can access the same data sets.

Defining the security level to *already verified* and having users sign on to CICS allows users to customize their SAS environment. In order for users to use the CESN signon transaction, CICS external security must be active. The example TP profile defines four user-specific data sets: `SASUSER`, `SASAUTOS`, `CONFIG`, and `SASEXEC`. If you choose to retain these user-specific data sets in the TP profile, note that the `&SYSUID` system variable identifies them. This variable resolves to the user ID that CICS passes when issuing a TP allocation request. Therefore, if users do not sign on, or several users sign on with the same id, the potential exists for multiple users to attempt to access the same data sets for update. This could result in lockouts of users or in corrupted data.

In order to allow some users to access SAS 9.2 Foundation without first signing on to CICS (for example, those who do not desire any special customization), you can define a second TP profile with a key that specifies the CICS default user ID. This profile would not specify any output data sets with the `&SYSUID` system variable. For example, if the default user for your CICS system is `CICS1`, specify the following key on the `TPADD`:

```
TPADD
TPNAME(SAS_SESSION)
USERID(CICS1)
...
```

Note that to support user ID qualified TP profiles, the LU definition in your `APPCPMxx` member of `'SYS1.PARMLIB'` must specify a `TPLEVEL` of `USER`:

```
LUADD
ACBNAME ( SASSESS )
TPDATA ( SYS1 . APPCTP )
TPLEVEL ( USER )
```

Defining SAS/SESSION to CICS

To enable communication with APPC/MVS, be sure `ISC=YES` is specified in the system initialization parameters. To define the CICS resources required for SAS/SESSION, use the CEDA transaction of the Resource Definition Online (RDO) facility of CICS. For details on any of the parameters used, refer to the appropriate IBM documentation.

All of the resources for SAS/SESSION are contained in a single GROUP in the CICS System Definition (CSD) file. You can choose any name that is acceptable for groups (for example, `SASSESS`). The following are basic components of the `SASSESS` group.

Use the `DEFINE` function of the `CEDA` transaction for these definitions.

□ CONNECTION

defines the actual VTAM connection (SASC) between CICS and the APPC/MVS System. Note that the value of the Netname parameter (`SASSESS`) matches the `ACBNAME` for the `SASSESS` VTAM APPL definition.

Connection parameters required are as follows:

Connection	- SASC
Group	- SASSESS
Netname	- SASSESS
Accessmethod	- VTAM
Protocol	- APPC
Singlesess	- No
Datastream	- User
Recordformat	- U
Autoconnect	- All
Inservice	- Yes
Attachsec	- Local

□ **SESSION**

defines the session (`SASSESS`) on which the conversations will take place between CICS and SAS 9.2 Foundation. Note that the value of the `Connection` parameter (`SASC`) matches the name of the `Connection` in the preceding list. The `SASC` connection supports multiple sessions. Session parameters required are as follows:

<code>Session</code>	- <code>SASSESS</code>
<code>Group</code>	- <code>SASSESS</code>
<code>Connection</code>	- <code>SASC</code>
<code>Modename</code>	- <code>SASCLU62</code>
<code>Protocol</code>	- <code>APPC</code>
<code>Maximum</code>	- <code>00010,00010</code>
<code>Receivecount</code>	- <code>No</code>
<code>Sendcount</code>	- <code>No</code>
<code>Sendsize</code>	- <code>3840</code>
<code>Receivesize</code>	- <code>3840</code>
<code>Autoconnect</code>	- <code>All</code>
<code>Buildchain</code>	- <code>Yes</code>
<code>Discreq</code>	- <code>Yes</code>

The `Modename` `SASCLU62` refers to the VTAM logon mode table entry name for `APPC` (`LUTYPE6.2`). You can specify an existing entry in the VTAM logon mode table here. See "Define the VTAM Logon Mode" on page 140 for more information.

□ **PROGRAM**

defines the transaction program delivered with SAS/SESSION to CICS. The library that the program resides on must be concatenated with the CICS Relocatable Program Library (RPL), or the load member must be copied into the existing RPL. The required parameters for the `SASSESS` program are as follows:

<code>Program</code>	- <code>SASSESS</code>
<code>Group</code>	- <code>SASSESS</code>
<code>Language</code>	- <code>ASSEMBLER</code>
<code>Reload</code>	- <code>No</code>
<code>Resident</code>	- <code>No</code>
<code>Status</code>	- <code>Enabled</code>
<code>DataLocation</code>	- <code>Any</code>

□ TRANSACTION

defines the transaction (*SASC*), which invokes the program *SASSESS*, as indicated by the parameters. Note that the transaction name (*SASC*) matches the value of the Transaction parameter under the *SESSION* component. Transaction parameters required are as follows:

Transaction	-	<i>SASC</i>
Group	-	<i>SASSESS</i>
Program	-	<i>SASSESS</i>
Profile	-	<i>SASSESS</i>
Status	-	Enabled
TaskDataLoc	-	Any

□ PROFILE

defines the *SASSESS* profile. This profile makes the *SASC* transaction use the terminal's alternate display size (as SAS 9.2 Foundation does) in all communications with the terminal. The profile also defines the modename used for APPC communication with SAS 9.2 Foundation. Profile parameters required are as follows:

Profile	-	<i>SASSESS</i>
Group	-	<i>SASSESS</i>
Scrnsize	-	<i>ALTERNATE</i>
Modename	-	<i>SASCLU62</i>

If the user already has a profile defined that meets these requirements, that profile name can be used in the transaction component instead of *SASSESS*.

□ PARTNER

defines the *SASCSESS* partner. This partner defines the network LU name and the APPC/MVS transaction program name used to communicate with SAS 9.2 Foundation. It also specifies a profile that defines the modename for APPC communication. Partner parameters required are as follows:

Partner	-	<i>SASCSESS</i>
Group	-	<i>SASSESS</i>
Netname	-	<i>SASSESS</i>
Profile	-	<i>SASSESS</i>
Tpname	-	<i>SAS_SESSION</i>

The partner name must be the concatenation of the transaction name (*SASC*), and the suffix (*SESS*). This allows the installation to define different APPC/MVS transaction program profiles for different SAS 9.2 Foundation configurations.

Activating the Interface

SAS/SESSION on APPC/MVS

To activate SAS/SESSION on APPC/MVS, complete the following steps:

1. Start APPC/MVS and its transaction scheduler under the control of the Master Scheduler, as shown in the following:

```
START APPC , APPC=xx , SUB=MSTR
```

```
START ASCH , ASCH=xx , SUB=MSTR
```

where `xx` is the suffix of your `APPCPMxx` and `ASCHPMxx` members in `'SYS1.PARMLIB'`.

If APPC/MVS and its transaction scheduler are already started, activate your members using the `SET` command, as shown in the following:

```
SET APPC=xx
```

```
SET ASCH=xx
```

where `xx` is the suffix of your `APPCPMxx` and `ASCHPMxx` members in `'SYS1.PARMLIB'`.

2. Verify that the SASSESS LU is active. You can display its status by issuing the following command:

```
DISPLAY APPC , LU , ALL , LLUN=SASSESS
```

After activating the CICS interface, the LU display should show the following:

```
PARTNERS=00001
```

3. After the CICS interface is active and users begin using SAS/SESSION, periodically display the status of the transaction programs. You can display its status by issuing the following command:

```
DISPLAY APPC , TP , ALL , LLUN=SASSESS
```

For more information on managing APPC/MVS resources, refer to the appropriate IBM documentation.

SAS/SESSION on CICS

To activate SAS/SESSION on CICS, complete the following steps:

1. Install the group `SASSESS` by issuing the `CEDA` command:

```
CEDA INSTALL GROUP(SASSESS)
```

A message on the CEDA output display should say `Install Successful`. If it does not, check the parameters you supplied to determine the problem and reissue the command.

2. Press PF15 to terminate CEDA processing.

Note: If the auto install list includes the group `SASSESS`, you do not need to issue the `CEDA` command.

3. Issue the following command to verify the connection to APPC/MVS:

```
CEMT I CONN
```

The status of the SASC connection appears on the screen. After the group `SASSESS` is installed, the status shown should be `Ins Acq` (in service and acquired).

Once the connection has been acquired, the sessions associated with that connection are allocated automatically.

4. Press PF15 to terminate CEMT processing.
5. Clear the display.

Executing SAS 9.2 Foundation

To execute SAS 9.2 Foundation, first sign on to CICS to enter your user ID and password if required by your SAS administrator:

CESN

Then, use the SASC transaction to execute SAS 9.2 Foundation:

SASC <options>

where *options* are any valid SAS system options.

Note: *SASC refers to the CICS transaction name as defined in the Transaction component. Please see "Defining SAS/SESSION to CICS" on page 143 for more information.*

The session proceeds as if you had entered the SAS command from a TSO terminal. Refer to the *SAS 9.2 Companion for z/OS* for more information.

At the end of the session, the following message indicates that all resources associated with this transaction have been released in the CICS region:

SAS/SESSION complete, return code is 0

Refer to *z/OS Application Development: Writing Transaction Programs for APPC/MVS* for information on other return codes.

Appendix Q — Implementing SAS/SHARE Software

Note: For further information on implementing and using SAS/SHARE software, please refer to Communications Access Methods for SAS/CONNECT and SAS/SHARE Software and the SAS/SHARE User's Guide which are both in the SAS OnlineDoc.

Special Files for Use with SAS/SHARE Software

Customizing the Started Task JCL Procedure for a Server

Note: This task is only required if you want clients to be able to use the SHRMACS macros.

STEP 1: Edit the `SHREDITP` member of the `CNTL` data set and specify the following parameter values.

- SASSNM=** Supply your started task procedure name. This JCL procedure name and member name will be created in the procedure library, `&prefix.PROCLIB`
- SASSTP=** Supply your stop task procedure name. This JCL procedure name and member name will be created in the procedure library, `&prefix.PROCLIB`
- SERVERID=** Supply your default server ID
- PROC-DSN=** Supply your `PROCLIB`

STEP 2: Edit and submit the SHRPOST job.

This job modifies the SAS/SHARE started task JCL procedure, member `SHRPROC01` in the `CNTL` data set, with the `SHREDITP` parameter values from Step 1 above (`SASSNM=` and `SASSTP=`) and copies the procedure to the library specified with `PROC-DSN`.

STEP 3: Notify the server administrator that this file has been provided.

STEP 4: Please refer to the *SAS/Share User's Guide*, Appendix 2, "Creating the SAS/Share Server Environment," and Appendix 3 "Tuning Tips for Applications That Use SAS/Share Software."

Configuration File for a Server

Member `SRVCNFG` of the `CNTL` data set is provided as a default configuration file for a server's SAS execution. This member contains recommended SAS system option settings and is included in the `CONFIG` concatenation in the started task JCL procedure customized according to the procedure described in the previous section.

Customizing the SAS/SHARE Autocall Macros

Note: This task is required.

The installed `SASSAML` data set is an `APPLSYS` macro library used by the SAS/SHARE autocall macros. This library contains the required members `DEFAULTS` and `SERVERID`. These members contain instructions with examples in comment headers for adding entries to the tables used by the macros. To use this library, you must specify its name in the `SHRMACS` autocall macro.

STEP 1: Edit the SHRMACS autocall macro (required).

Member `SHRMACS` in the `AUTOLIB` data set must contain the correct data set reference for the `APPLSYS` macro library. Edit this member to change the name `SAS.SASSAML` to the correct installed `&prefix.SASSAML` data set name for your installation.

STEP 2: Notify the server administrator that this file has been provided.

Selecting Communications Access Methods to Use

Note: This task is required.

STEP 1: Determine the access method to use.

Communication between a SAS/SHARE server and user is handled by a part of SAS called a communications access method. There are two communications access methods available for use with this release of SAS/SHARE software under z/OS:

- cross-memory services
- TCP/IP

To use the cross-memory services access method, a server and user must be running on the same z/OS system.

You can choose to use one access method exclusively, or you can choose one as your primary access method and others as secondary access methods. If you choose to define primary and secondary access methods, SAS/SHARE software will attempt to establish a user-to-server connection using the primary access method first. If that attempt fails, SAS/SHARE software will then attempt to establish the connection using each of the secondary access methods in turn.

You should choose the access methods you will use based on your site's requirements and restrictions. The cross-memory access method is the default and is faster than the other access methods. However, the cross-memory access method can only be used for within-system communication and requires installation of a module in an authorized link list library and definition of an inactive z/OS subsystem.

Note: SAS/SHARE 9.2 is not compatible with the V8 cross-memory module (normally `SASVXMS`). The V9 module is backward compatible with V8 Share so it can service both releases.

STEP 2: Set SAS system options to specify selected access methods.

SAS 9.2 Foundation option `COMAMID=` specifies which access method SAS/SHARE software should use as the primary or only access method. SAS 9.2 Foundation option `COMAUX1=` specifies the secondary access method. These options are specified, typically in a SAS 9.2 Foundation configuration file, by the SAS/SHARE Software Consultant.

The following table shows the value of these options for each access method:

Access Method	COMAMID=/COMAUX1= Value
cross-memory services	XMS
TCP/IP	TCP

For a server, these two options have a related meaning; each access method specified by these options will be initialized when the server is started, making the server accessible to users via any of those access methods.

For example, for a server that is to be accessible only to users who use the cross-memory services access method, specify

```
COMAMID=XMS
COMAUX1=
```

For a server that is to be accessible to users who use either the cross-memory services access method or the TCP access method, specify

```
COMAMID=XMS
COMAUX1=TCP
```

Or

```
COMAMID=TCP
COMAUX1=XMS
```

For a user session, the access method specified by the `COMAMID=` option is the first one used to attempt to connect to a server.

Note that it is not necessary to specify `COMAUX1=` if you do not want to specify a secondary access method.

To cause a user session to try the cross-memory services and TCP/IP access methods, in that order, specify

```
COMAMID=XMS
COMAUX1=TCP
```

System Configuration for the Cross-Memory Access Method

Installing the SASVXMS Load Module

Note: This task is required.

To use the cross-memory access method for communication between a SAS/SHARE server and user, you must copy the module `SASVXMS0` from the SAS load library data set into an authorized library. You must then rename this module `SASVXMS` (removing the 0). It is very important that you perform these two tasks in that order.

When SAS/SHARE software loads the module `SASVXMS`, it must find that module to be marked authorized, re-entrant, and reusable, and to have been loaded from an authorized library.

The version of SASVXMS which was distributed with release 6 of SAS/SHARE software can be used ONLY with version 6. If you still have version 6 of SAS/SHARE software installed, be sure to follow the special instructions in Step 2.

STEP 1: Copy `SASVXMS0` into an authorized link list library.

Copy the module `SASVXMS0` into any authorized library. In a production environment, SAS recommends you copy the `SASVXMS0` module into an authorized link list library. Alternatively, you can install this module into the link pack area. You can use any standard utility program to copy the module `SASVXMS0`

from your `&prefix.LIBRARY` data set to your authorized library.



Note: A userabend 984 will occur if the SASVXMS module is not installed in an authorized library or the library is in a STEPLIB concatenation where one of the libraries is not authorized.

STEP 2: Rename SASVXMS0.

After copying SASVXMS0 into the appropriate library, you must rename it. You can use any standard utility to rename the module.

If you do **not** have a version 6 of SAS/SHARE software installed, rename SASVXMS0 to SASVXMS. Specify SAS 9.2 Foundation option COMAMID=XMS as described earlier.

If you have version 6 of SAS/SHARE software installed, rename SASVXMS0 to SASVXMSn, where n is version of SAS 9.2 Foundation. Specify SAS 9.2 Foundation option COMAMID=XMSn. For example, for SAS/SHARE 9.2, rename SASVXMS0 to SASVXMS9 and specify COMAMID=XMS9.



Note: The XMS access method does not support communication between a Version 6 SAS session and a SAS/SHARE 9.2 server, nor does it support communication between a SAS 9.2 Foundation session and a Version 6 SAS/SHARE server.

If you want to run SAS/SHARE 9.2 and Version 6 SAS/SHARE software concurrently, you **MUST** rename the Release 9.2 copy of SASVXMS0 and set the COMAMID= option appropriately. Failure to do so will generate errors, the most common being ERROR: XMS Communication Failure. Unable to locate system XVT Anchor.

Defining an Anchor Point

Note: This task is required.

To use the default cross-memory access method for communication between a SAS/SHARE server and user, you must define an anchor point. The anchor point is a place in common memory that can be located by servers and users and used to store and retrieve cross-memory communication information.



Note: If you have defined an anchor point for a previous release of SAS/SHARE software, it is not necessary to repeat this step now.

STEP 1: Define an inactive z/OS subsystem.

The anchor point is specified by defining an inactive z/OS subsystem. Defining an inactive subsystem causes z/OS to create a subsystem communications vector table (SSCVT) at IPL time. The SSCVT chain is in common memory and easily accessible to the cross-memory access method routines. The SSCTSUSE field of the SSCVT is available to these routines and is used as the anchor point for their control blocks.

You should note that, although you define a subsystem to z/OS, it will never be considered active and will provide no system services because the SSCTSSVT field of the SSCVT will never be non-zero.

You can define the inactive subsystem by adding an entry to any of the following:

- the IEFJSSNT member of SYS1.LINKLIB
- an IEFSSNxx member of SYS1.PARMLIB.

Consult z/OS system initialization and tuning documentation for the details of each alternative.

Regardless of which method you choose, you must include the subsystem name and you must **not** specify an initialization routine name. Use the name `SAS0` unless it conflicts with standards or conventions at your site.

STEP 2: Set SAS 9.2 Foundation option `SUBSYSID=` to specify the inactive subsystem you defined.

The name you specify for the inactive subsystem defined as the anchor point for the cross-memory access method must also be specified as the value of SAS 9.2 Foundation option `SUBSYSID=`. This option is specified, typically in a SAS 9.2 Foundation configuration file, by the SAS/SHARE Software Consultant. This option is described in *Communications Access Methods for SAS/CONNECT 9.2 and SAS/SHARE 9.2*.

System Configuration for TCP/IP

Note: *The TCP communications access method within SAS/SHARE requires that you configure SAS 9.2 Foundation for communication with TCP/IP. For this reason, please ensure that you have reviewed and completed the steps in the section “System Configuration for Using SAS with TCP/IP” on page 21.*

Specify SAS 9.2 Foundation Option `TCPSEC=_SECURE_` for the Server Execution

Note: *This task is required.*

Specify the SAS 9.2 option `TCPSEC=_SECURE_` in `&prefix.CNTL(SRVCNFG)`, described in “Configuration File for a Server” on page 149. This will cause the TCP/IP access method to require users to supply a valid user ID and password for the z/OS system where the server is running in order to connect to the server.

Testing the SAS/SHARE Server Startup

After bringing up your SAS/SHARE server, from a client SAS session execute the following code:

```
PROC OPERATE SERVER=server-ID SAPW=oapw;
    DISPLAY LIBRARY _ALL_ ;
    DISPLAY USER _ALL_ ;
    DISPLAY SERVER server-ID ;
RUN;
```

Note: *The server-ID must be identical to the service name that is specified in the SERVICES file. For details, see “The Services File” on page 35 .*

Note: *oapw is the operator password provided at invocation of the SAS/SHARE server. If OAPW= was not provided at invocation of the SAS/SHARE server, then remove the “SAPW=” option from the example.*

Client-Side Components

SAS/SHARE software includes client components that are used outside of your SAS installation. SAS/SHARE client components are available on the **SAS Client-Side Components CD** included with your SAS Software distribution. Please refer to these CDs for more information.

These components are described below:

SAS/SHARE Data Provider

The SAS/SHARE data provider enables you to access, update, and manipulate SAS data using OLE DB- and ADO-compliant applications on Windows platforms.

SAS ODBC Driver

The SAS ODBC driver enables you to access, update, and manipulate SAS data from ODBC-compliant applications on Windows platforms.

SAS/SHARE Driver for JDBC

The SAS/SHARE driver for JDBC enables you to write applets, applications, and servlets that access and update SAS data. The Java Tools package that includes the SAS/SHARE driver for JDBC also includes the SAS/CONNECT driver for Java. If you are writing Java programs using these interfaces, you might also want to use the tunnel feature. This optional feature can be used with the Java applets you write to solve some common configuration problems.

SAS/SHARE SQL Library for C

The SAS SQL Library for C provides an application programming interface (API) that enables your applications to send SQL queries and statements through a SAS/SHARE server to data on remote hosts.

Special Consideration for the SECPROFILE System Option

There are two installation requirements that must be met before the SECPROFILE system option can be used with the TCP access method:

1. The SAS 9.2 Foundation SVC routine must be installed, and must be at Level 8 or greater. See "Installing the SAS 9.2 Foundation SVC Routine" on page 41 for details on the SAS SVC.

Note: *The SVC routine shipped with V9 SAS is Level 8. The SVC routine shipped with V8 SAS was Level 7.*

2. The RACF security administrator must activate the PTKTDATA class, and define at least one PTKTDATA profile for use by SAS/SHARE. If the client and server are on different systems, these steps must be done on both systems, and the profile definitions must be identical on both systems.

Note: Provided that the Level 8 SVC Routine is installed (on both client and server systems, if they are different), SAS 9.2 Foundation clients can use the SECPROFILE option to connect to SAS Version 6, 7, or 8 SAS/SHARE servers without a password. The only restriction is that, since the SECPROFILE option does not exist in SAS versions prior to SAS 9.2 Foundation, the PTKTDATA profile name can only be the RACF default name. For MVS batch jobs, this is typically the characters "MVS" prefixed to the SMF system identifier of the MVS system, although it can be changed by an ICHRIX01exit.

Appendix R — Implementing SAS/TOOLKIT Software

The SAS/TOOLKIT installation CNTL data set for each language contains sample JCL members that include in-stream cataloged procedures. To make it easier for your users to take advantage of the SAS/TOOLKIT product, you can customize these cataloged procedures for your site and install them in your standard cataloged procedure library.

The member APROC in the `&prefix.TLKT.ASM.CNTL` library contains the definitions of the cataloged procedures UWASM and UWLINKA. These are needed by SAS/TOOLKIT assembler users.

The member CPROC in the `&prefix.TLKT.C.CNTL` library contains the definitions of the cataloged procedures UWC, UWCLINK, and UWLINKC. These are needed by SAS/TOOLKIT SAS/C users.

In all cases, the procedures are defined as in-stream procedures. You must remove the PEND line as you add each member to the cataloged procedure library at your site. Also, follow the notes in the cataloged procedure text to see where you need to change data set names, entry points, etc. Any lowercase data set names should be changed to the correct data set names for your installation.

Note: *Some early installations of SAS 9.2 may see data set names beginning with `&prefix.TOOLKIT` (as opposed to `&prefix.TLKT`.) These `&prefix.TOOLKIT` names may be used as is and APROC and CPROC will use the longer names.*

Appendix S — Customizing SAS System Forms

As the SAS Consultant for your site, you have the ability to customize all SAS forms for your operating system. Associated with each form is a list of available printers. Review this information if users at your site require a site customized print form for use in windowing environments, SAS/FSP, SAS/AF, or SAS/ASSIST.

Note: You must have SAS/AF Software licensed to modify the site form.

Customizing the Printer Selection List

Note: This task is optional.

Whenever you create a SAS 9.2 Foundation form, a list of printers is displayed. You can modify this list to reflect only those printers available for your site. Information on changing the printer selection list has been included in a help file within the SASHELP library. To find out more about customizing this printer list, issue the following command from the windowing environment command line:

```
af c=sashelp.base.pdevice.cbt
```

This command displays a series of help screens that provide instructions for adding, deleting, and modifying entries in the PDEVICE Catalog.

Appendix T — Licensing the SAS 9.2 Foundation

In order to run each software product that you license from SAS, you must apply the SETINIT information, which contains current data about your licensing agreement with SAS. The SETINIT information is stored in the SASHELP library of the SAS system. SAS software will not run without up-to-date SETINIT information. You must update your SETINIT information whenever you renew your license, add new products, or change the hardware for the system.

Requests for changes to your license parameters are accepted by our Customer Service Department by phone or in writing on your company's official stationery. Such requests include changes to the expiration date for your SAS software as well as updates of the serial number or CPU model specification when you change your hardware.

Note: Only the authorized SAS Installation Representative should change the SETINIT information. Your site designated the SAS Installation Representative when you licensed the SAS system.

Processing Renewal of the SAS System

Each SAS software product you install contains a file with a list of SAS statements used to invoke the SETINIT procedure. The information supplied with the SAS installation data file (SID file) reflects your current license agreement with SAS, and contains code to update the SETINIT information for each licensed product.

Important: Update access to the SASHELP library is required to update your SETINIT information. Be sure to stop all SAS Servers and SAS Spawners before renewing your software.

This procedure involves using both a PC (running a terminal emulator which has access your mainframe system) and your mainframe system. Follow these steps to renew your SAS software license:

1. Copy the attached file (`sas92_XXXXXX.txt`) to the PC from which you will invoke your terminal emulator to connect to the mainframe.
2. Edit the renewal SID file (`sas92_XXXXXX.txt`) that you saved in Step 1. Copy the lines starting with `'PROC SETINIT RELEASE='9.2';'` through `'SAVE; RUN;'`.
3. Use the terminal emulator on your PC to log on to your mainframe system where SAS is running. Replace the content of the RENEWPRM member of the `&prefix.CNTL.RENEW` data set with the information you copied in Step 2.

Note: `&prefix` is the high-level qualifier for your SAS installation.

4. Submit the **RENEW** job from the `&prefix.CNTL.RENEW` data set.
5. Check the SAS LOG for the message that "Siteinfo data have been updated" to confirm the SETINIT procedure executed successfully.

Important: Restart all SAS Servers and SAS Spawners after renewing your software. In order to use any newly licensed products included in a new SID file and for new expiration dates to take effect, you must restart all SAS Servers and SAS Spawners after applying the new SETINIT.

SETINIT Troubleshooting

The following is a list of common error messages and solutions that may occur when attempting to update your SETINIT information. If you continue to receive errors after attempting troubleshooting, contact the Technical Support department at SAS.

❑ **ERROR:**

```
ERROR: INCORRECT INFORMATION WAS ENTERED FOR PROC SETINIT. ALL
INFORMATION MUST BE ENTERED EXACTLY AS IT APPEARS ON THE PROC SETINIT
DATA RECEIVED FROM SAS INSTITUTE.
```

Or

```
ERROR: INCORRECT INFORMATION WAS ENTERED FOR THE PASSWORD XXXXXXXX
```

SOLUTION:

The SETINIT information in the RENEWPRM member of the CNTLDSN must be entered *exactly* as it appears in the SID file which was attached to your Software Order E-mail. If any text of the SETINIT procedure is not the same, the above error occurs when you attempt to execute the RENEW job.

❑ **ERROR:**

```
THE SAS SYSTEM IS EXECUTING ON A PROCESSOR (CPU) WHOSE MODEL NAME,
MODEL NUMBER, AND SERIAL NUMBER ARE NOT INCLUDED IN THE SETINIT DATA
USED TO INITIALIZE THE SAS SYSTEM LIBRARY IN USE. THIS IS PERMITTED IF
THIS PROCESSOR IS A DESIGNATED BACKUP PROCESSOR FOR A LICENSED CPU.
FOR THIS SITE, THE SAS SYSTEM IS LICENSED FOR THE FOLLOWING CPU SERIAL
NUMBERS:
```

```
MODEL IBM xxxx-xxxx SERIAL NUMBER YYYYYY
```

SOLUTION:

When the SETINIT procedure is executed on a processor that is not included in the SETINIT information, the above error message is issued. Be sure that the SAS system is running on the processor indicated in the SETINIT information. If the model name, number, or serial number of your mainframe system is different than the one listed in the SETINIT, contact your SAS Customer Service Representative for an updated SETINIT.

❑ ERROR:

THE SITE VALIDATION DATA CANNOT BE UPDATED. THIS IS MOST LIKELY DUE TO THE FACT THAT THE SASHELP CATALOG IS NOT AVAILABLE IN WRITE MODE, AND/OR THAT THE SETINIT OPTION HAS NOT BEEN SPECIFIED WHEN USING THE SAS COMMAND.

SOLUTION:

The above error indicates that UPDATE access to the SASHELP library was denied. This is most likely due to not specifying the SETINIT option when using DISP=SHR, not having a DISP=OLD, or not having the appropriate access authority (UPDATE required) to the SASHELP library. Make sure the necessary changes were made to the RENEW job and resubmit the job.

For More Information

If you have questions about your SID file, please call our Customer Service Department at (919) 677-8003 between 9:00 a.m. and 8:00 p.m. Eastern Time, SAS business days. Please have your site number ready when you call.

If you encounter problems applying your SETINIT, please call our Technical Support Division at (919) 677-8008 between 9:00 a.m. and 8:00 p.m. Eastern Time, SAS business days. Ask the Technical Support receptionist for a z/OS consultant. Please have your site number ready when you call.

Requesting a Temporary License Extension

For emergency situations, you can download a temporary SID file that will extend the use of your licensed SAS software products for six days. Please visit the SAS Install Center, <http://support.sas.com/documentation/installcenter/index.html>. Under *SITE AND ACCOUNT DATA* on the right side of the page, select **Request a Temporary License Extension**. You may also select **Resend the SAS Installation Data**.

For security purposes, you will be required to enter a password, and the process for creating a password may also be started from the same Web page.

Appendix U — Logging Directly on to the SAS System

z/OS sites can choose to substitute SAS 9.2 Foundation for the standard TSO terminal monitor program. Sites can insulate users from the TSO environment by automatically invoking SAS 9.2 Foundation or a SAS application when users log on.

Because SAS 9.2 Foundation is running as its own terminal monitor program, TSO commands are not accessible to users.

This technique is intended for z/OS sites interested in restricting interactive user access to the TSO environment or shielding novice users from having to learn how to work in the mainframe environment. Sites that use this technique also save a little memory.

This appendix describes how to install and use the direct logon procedure, and provides an example. It also discusses the differences between logging onto SAS 9.2 Foundation using the windowing environment, using a windowing application, as well as the possibility of using the direct logon process with SAS/CONNECT software.

In most circumstances, only system administrators need to read this appendix. If you are not a system administrator and are interested in logging directly onto SAS 9.2 Foundation, see your SAS Installation Representative.

Installing the Direct Logon Procedure

When users log onto the system, a JCL stream called a logon procedure (logon proc) is automatically executed. Normally, the logon procedure activates the TSO terminal monitor program that sends the TSO READY prompt to the display when the logon process is complete. To make SAS 9.2 Foundation the logon environment, replace this procedure with a logon procedure that activates SAS 9.2 Foundation as the terminal monitor program.

To use this capability you must still start TSO. Any user logging onto SAS 9.2 Foundation must have a valid TSO user ID. However, invoking SAS 9.2 Foundation directly results in a reduction of approximately 50K in working set size for each user.

Complete the following steps to allow users to log directly onto SAS 9.2 Foundation:

- Create a logon procedure that is used by all users directly logging on to SAS 9.2 Foundation.
- Install the logon procedure into your site's logon procedure library.
- Specify the SAS logon procedure as each user's logon procedure.

The logon procedure that you create is similar to the standard SAS cataloged procedure. This procedure is in the PROCxx member of your site's control installation data set (CNTLDSN).

Example Logon Procedure

Example JCL for a logon procedure can be found in `&prefix.PROCLIB`.

After making any site-specific modifications, copy the PROC into a valid site logon procedure library. To enable users to access the SAS direct logon facility, modify their user IDs to use the modified logon procedure.

Using Direct Logon

You can use the direct logon technique to log onto the windowing environment of SAS 9.2 Foundation, or you can choose to log directly onto a windowing application. You can even use this technique in combination with SAS/CONNECT software to log directly onto SAS 9.2 Foundation on the mainframe from your workstation.

The purpose of combining the direct logon technique with SAS/CONNECT software is to restrict users that connect to the mainframe from having access to the TSO environment. For information on using SAS/CONNECT software, refer to *Communications Access Methods for SAS/CONNECT 9.2 and SAS/SHARE 9.2*.

Logging onto the SAS Display Manager System

Use the example logon previously described. If you need to allocate special files for each user, such as individual SASUSER files, you must create a separate logon procedure for each user because of a system restriction.

Unless you are using the Amdahl Logon Pre-prompt Exit Version 2.7.5, you do not have any control over supplying customized SAS system options, configuration files, or the dynamic allocation of a user's SASUSER data set to a single logon procedure.

Logging onto a Windowing Application

To log directly onto a windowing application, specify an autoexec file for the application. To do this, add a SASEXEC statement that supplies the data set name of the file containing the autoexec code to the example logon procedure. This SASEXEC statement has the following form:

```
//SASEXEC DD DISP=SHR,DSN=autoexec-file
```

For more information on modifying the way in which SAS 9.2 Foundation is invoked, see Chapter 1, "Initializing and Configuring SAS 9.2," in the *SAS 9.2 Companion for z/OS*.

Restrictions

Using SAS 9.2 Foundation as the logon environment implies certain restrictions. For example, because SAS 9.2 Foundation is the terminal monitor program, users cannot execute TSO commands or access TSO facilities such as ISPF from their SAS sessions. Nor can users issue the TSO or X command from their SAS sessions to gain access to the TSO environment.

However, the SAS windowing environment contains environment-dependent statements, windows, and a full-function editor that perform many of the same utilities available in ISPF. These services are available to users that log directly onto SAS 9.2 Foundation. Users can dynamically allocate any files they are authorized to access using LIBNAME and FILENAME statements. They can also use the INCLUDE command to include external files and members of partitioned data sets into SAS editor windows. For more information on these and other operating-system-dependent language features, see the *SAS 9.2 National Language Support (NLS) User's Guide*.

If the logon procedure you have provided includes a DDname assigned to the internal reader, users are able to submit batch jobs from within their SAS sessions.

Accounting Considerations

Substituting SAS 9.2 Foundation for the standard IBM terminal monitor program affects records produced by SMF and TSO/MON. SMF type 30, 34, and 35 records have the SAS entry name in the program name field rather than IKJEFT01 or ADFMDF03. Type 32 (TSO command) records are not produced.

If you are using LEGENT Corporation's TSO/MON product, TSO/MON system records contain complete resource usage, transaction, and response time information, but no command information. TSO/MON command detail records are not produced.



THE
POWER
TO KNOW.

support.sas.com

SAS is the world leader in providing software and services that enable customers to transform data from all areas of their business into intelligence. SAS solutions help organizations make better, more informed decisions and maximize customer, supplier, and organizational relationships. For more than 30 years, SAS has been giving customers around the world The Power to Know®. Visit us at **www.sas.com**.