Shared Virtual Array Administrator

Version 3.1

for OS/390

Installation, Customization, and Maintenance

Part Number: 311290809

We welcome your feedback. Please contact the Sun Learning Services Feedback System at:

SLSFS@Sun.com

or

Sun Learning Services
Sun Microsystems, Inc.
One StorageTek Drive
Louisville, CO 80028-3256
USA

Please include the publication name, part number, and edition number in your correspondence if they are available. This will expedite our response.

Contents

Figures

Tables

About This Book

This book describes how to install, customize, and maintain Version 3.1 of Shared Virtual Array Administrator (SVAA) on the OS/390, z/OS, or MVS operating system. SVAA is a software package that helps you configure, manage, and track performance for the Shared Virtual Array (SVA) subsystem.

Note: Although this book generally mentions only OS/390, the SVAA for OS/390 software runs identically on the **z/OS, OS/390, or MVS host operating system**.

Who Should Read This Book

This book is for the systems programmers responsible for initializing and customizing SVAA in an OS/390 operating system environment. It assumes that readers are familiar with the information contained in the *V2Xf Shared Virtual Array: Introduction*, *SVAA for OS/390 Configuration and Administration*, and *SVAA for OS/390 Reporting* manuals. It also assumes that readers have experience with the OS/390 operating system, System Modification Program Extended (SMP/E), Time Sharing Option/Extended (TSO), and the Interactive System Productivity Facility (ISPF).

Conventions Used in This Book

Use of "Subsystem"

The term “subsystem” is used in two ways: ***OS/390 subsystem*** refers to the SVAA software running under the master address space; ***SVA subsystem*** refers to the SVA hardware. In command descriptions, the variable ***ssname*** always represents the name of the installed SVAA subsystem, while ***subsys*** represents the name assigned to the SVA subsystem (the hardware).

Notation Used in Syntax Diagrams

Throughout this library, diagrams are used to illustrate the programming syntax. The following list tells you how to interpret the syntax diagrams:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

  The ►►── symbol indicates the beginning of a statement.

  The ──► symbol indicates that the statement syntax continues on the next line.

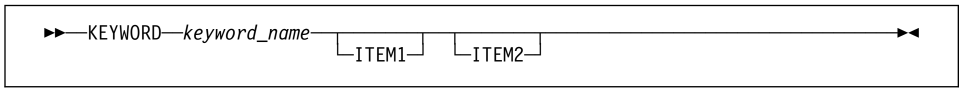
  The ►── symbol indicates that a statement is continued from the previous line.

  The ──►◄ symbol indicates the end of a statement.

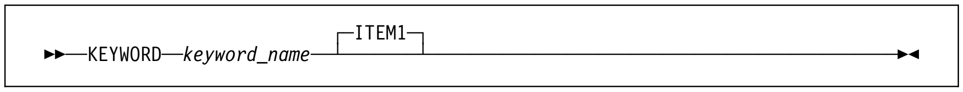
- Items shown on the main path of the statement are required.

```
►►──KEYWORD──keyword_name──────────────────────────────────────────►◄
```

- Items shown on branches below the main path are optional.

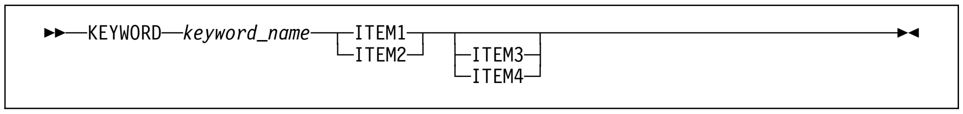


- Items shown on branches above the main path are default values.



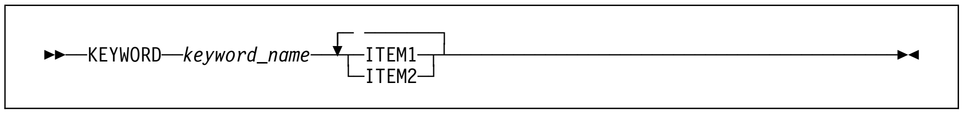
- Items appearing in a stack indicate that only one of the items can be specified. When one of the items in a stack appears on the main path, you must include one of the items.

  For example, in the following diagram, you must include either `ITEM1` or `ITEM2`. `ITEM3` and `ITEM4` both appear below the main path, so neither one is required.

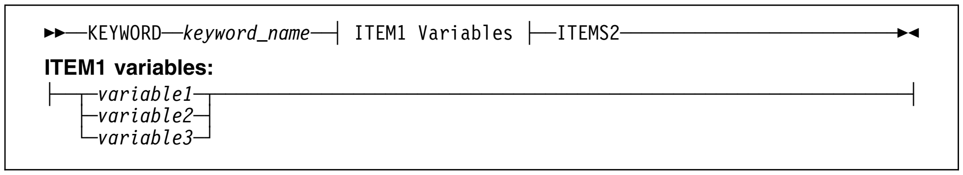


- A repeat arrow shown above an item or a stack of items indicates that you can specify the item multiple times or specify more than one of the items. A character (such as a comma or a blank) on the repeat arrow indicates that the items must be separated by that character.

  For example, in the following syntax diagram, you can specify both `ITEM1` and `ITEM2`, but you must use a blank to separate your choices in your programming syntax.



- In some cases, when an item has additional items associated with it, an additional syntax diagram is shown that represents the full syntax of that item. For example, in the following syntax diagram, additional information that can or must be specified for `ITEM1` appears in the “ITEM1 Variables” syntax diagram.



Shared Virtual Array Documentation

This section lists both software documentation and hardware documentation for the Shared Virtual Array products.

How to Obtain Software Documentation

All of the Shared Virtual Array software publications are available:

- On the “SVA Software Publications” CD-ROM (part number 3134524*nn*). To order a copy, contact StorageTek Publication Sales and Service at 800-436-5554 or send a fax to 303-661-7367.
- Online (for viewing and printing), at the StorageTek Customer Resource Center (CRC) web site at: **www.support.storagetek.com**
  Click on Software and go to the Shared Virtual Array Software list.

  Note: Access to the CRC site requires a password. To obtain a password, call StorageTek Customer Support at 800-678-4430.

SVA Administrator Library:

SVA Administrator for OS/390

- *Shared Virtual Array Administrator for OS/390 Configuration and Administration*
  3112905*nn*
- *Shared Virtual Array Administrator for OS/390 Installation, Customization, and Maintenance*
  3112908*nn*
- *Shared Virtual Array Administrator for OS/390 Reporting*
  3112906*nn*
- *Shared Virtual Array SnapShot for OS/390 Installation, Customization, and Maintenance*
  3112913*nn*

SVA Administrator for VM

- *Shared Virtual Array Administrator for VM Configuration and Administration*
  3134629*nn*
- *Shared Virtual Array Administrator for VM Installation, Customization, and Maintenance*
  3134631*nn*
- *Shared Virtual Array Administrator for VM Reporting*
  3134630*nn*

SVA Administrator for OS/390 and VM

- *Shared Virtual Array Administrator for OS/390 and VM Messages and Codes*
  3112907*nn*

For any StorageTek Software:

- *Requesting Help from Software Support*
  1121240*nn*

SVA Hardware Publications

Shared Virtual Array hardware publications are available:

- Online (for viewing and printing), at the StorageTek Customer Resource Center (CRC) web site at: **www.support.storagetek.com**
  Click on Disk Subsystems.

  Note: Access to the CRC site requires a password. To obtain a password, call StorageTek Customer Support at 800-678-4430.

V2Xf SVA Library:

- *V2Xf Shared Virtual Array*
  General Information
  MO9216*x*
- *V2Xf Shared Virtual Array*
  Introduction
  MO9217*x*
- *V2Xf Shared Virtual Array*
  Operation and Recovery
  MO9219*x*
- *V2Xf Shared Virtual Array*
  Planning
  MO9218*x*
- *V2Xf Shared Virtual Array*
  Reference
  MO9220*x*
- *V2Xf Shared Virtual Array*
  System Assurance
  MO9221*x*
- *V2Xf Shared Virtual Array*
  Peer-to-Peer Remote Copy Configuration Guide (PPRCfcn)
  MO9211*x*

V2X SVA Library:

- *V2X Shared Virtual Array*
  General Information
  MO9133*x*
- *V2X Shared Virtual Array*
  Introduction
  MO9135*x*
- *V2X Shared Virtual Array*
  Operation and Recovery
  MO9137*x*

- *V2X Shared Virtual Array Planning*
  MO9136*x*
- *V2X Shared Virtual Array Reference*
  MO9139*x*
- *V2X Shared Virtual Array System Assurance*
  MO9138*x*
- *V2X Shared Virtual Array System Assurance*
  MO9138*x*

V960 SVA Library:

- *V960 Shared Virtual Array General Information*
  MO5011*x*
- *V960 Shared Virtual Array Introduction*
  MO5006*x*
- *V960 Shared Virtual Array Operation and Recovery*
  MO5007*x*
- *V960 Shared Virtual Array Planning*
  MO5008*x*
- *V960 Shared Virtual Array Reference*
  MO5009*x*
- *V960 Shared Virtual Array System Assurance*
  MO5010*x*

Peer-to-Peer Remote Copy for V2X, V2X2, and V960:

- *Peer-to-Peer Remote Copy Configuration Guide*
  MP4007*x*

Trademarks

The following terms are trademarks or registered trademarks of Storage Technology Corporation:

- Iceberg
- Shared Virtual Array
- SnapVantage
- StorageTek
- SVA

The following terms are trademarks or registered trademarks of International Business Machines Corporation:

- DFSMS/MVS
- DFSMS/VM
- ESCON
- FICON
- IBM
- MVS
- OS/390
- RACF
- RMF
- VM/ESA
- z/OS
- z/VM

The following terms are trademarks or registered trademarks of Microsoft Corporation:

- DOS
- Excel
- Windows

The following terms are trademarks or registered trademarks of SAS Institute, Incorporated:

- SAS
- SAS/C
- SAS/GRAPH

The following terms are trademarks of Sun Microsystems, Inc.:

- Solaris
- Java

HP-UX is a trademark of Hewlett-Packard Company.

UNIX is a registered trademark of The Open Group.

Note: Other terms and product names used herein may be trademarks of their respective owners.

Product Support

StorageTek Customer Services provides 24-hour assistance for questions or problems related to StorageTek products. Calls from our customers receive immediate attention from trained diagnostic specialists. Call 800-678-4430.

Refer to the document *Requesting Help from Software Support* for detailed information about contacting StorageTek for technical support in your country or geographical location.

During problem resolution, Software Support may request that you provide specific diagnostic materials. Although printed data might be acceptable, data in machine-readable form is much preferred.

OS/390 Diagnostic Materials

Software Support may request one or more of the following kinds of diagnostic materials, depending on the nature of the problem:

- Details of circumstances
- OS/390 SYSLOG
- SYSMSG data set
- SYSPRINT data set
- SYS*x*DUMP and SYS1.DUMP*nn* data sets
- SYSHIST data set (LOG OFFLOAD)
- Database DUMPS/DEBUG
- EREP records (hardware and/or software)
- ISPF panel images
- ISPF panel names and SPFLOG
- External trace for SVAA via GTF
- CCW I/O traces
- SMF records
- Listings of SVAA files altered during installation, including the PROFSIBS and PROFSIBA macros
- Copies of logging files
- Output of SVAA started-task job
- Console dump with: SDATA=(ALLPSA,SQA,LSQA,RGN,LPA, TRT,CSA,SWA,SUMDUMP,ALLNUC,Q=YES,GRSQ)

Summary of Changes

Revision K, November 2006 EC 132753

This revision:

- Updates the SIBIOATX user exit in Chapter 3 to authorize PPRC subcommands.
- Updates table 3-2 in Chapter 3 to include facilities for PPRC subcommands.
- Updates table E-6 in Appendix E to include PPRC-related fields.

All significant changes are identified by a vertical bar in the left margin.

Tenth Edition (Rev J), March 2005 EC 132026

This edition:

- Updates the Software Prerequisites in Chapter 1 with new minimum Version and Release levels.
- Adds clarifications to the security resource requirements of Step 15 in Chapter 3.

All significant changes are identified by a vertical bar in the left margin.

Ninth Edition (Rev I), April 2004 EC 128971

This edition updates Steps 12 and 13 in Chapter 3 to remove the UID=0 restriction.

All significant changes are identified by a vertical bar in the left margin.

Eighth Edition (Rev H), November 2003 EC 128861

This edition:

- Adds a new appendix, Appendix F, which describes SVAA server cache. This cache is used by the SVAA Server as well as the point-in-time reports.
- Includes the term “Detached Operator Panel” or DOP wherever the Local Operator Panel or LOP is mentioned.
- Adds a “Migration Considerations” section to Chapter 1.

All significant changes are identified by a vertical bar in the left margin.

Seventh Edition (Rev G), February 2003 EC 128687

This edition:

- Updates Chapter 3 for the ability to snap offline volumes.
- Adds a new appendix, Appendix F, which describes SVAA server cache. This cache is used by the SVAA Server as well as the point-in-time reports.

All significant changes are identified by a vertical bar in the left margin.

Sixth Edition (Rev F), June 2002 EC 128518

This edition:

- Corrects the JCL in Step 2: Loading the Installation JCL. Emphasizes that the current installation JCL is on the SVAA Maintenance Tape—not the Product Tape.
- Updates Step 5h regarding the restrictions on the path name for the SVAA Server directories.

All significant changes are identified by a vertical bar in the left margin.

Fifth Edition (Rev E), January 2002 EC 123350

This edition:

- Updates the description and list of the Maintenance Tape contents on pages 2-2 and 6-1.
- Updates the Installation JCL shown on page 2-3.
- Updates the JCL shown in Steps 5a, 5b, and on page 6-2.
- Includes, for completeness, the VP parameter in the SIBSSIPL description (pages 3-5 and 3-8)—although it is ignored unless SnapShot is installed. Similarly, the VP option for the RIM parameter is added to the ALTER SSNAME description on pages 4-5 and 4-6.
- Adds a clarification regarding the SACLINK data set in Step 11d.
- Updates the contents of SIBSAMP listed in Appendix B.

Fourth Edition (Rev D), April 2001 EC 123253

This edition:

- In Tables 2-4 and 2-5 (pages 2-9 and 2-10), shows values for “Used 3390 Tracks” (instead of 3380).
- Corrects errors in the JCL shown in Steps 5e, 5j, and 5k.

Chapter 1. Overview

Installation Checklist

You may perform the tasks in the checklist below either before or after you install the SVA subsystem. If you choose to install SVAA before you install the SVA, you can use the Reporter functions of SVAA to collect statistics and generate reports for any non-SVA devices.

Use the checklist in Table 1-1 to ensure that you perform all the steps necessary to install and customize SVAA in the OS/390 environment.

Table 1-1 (Page 1 of 3). *Installation checklist*

Step	Description of action	Required or optional?	Page	√
1	Getting started	Required	1-8	
2	Load the installation JCL.	Required	2-3	
3	Configure the SMP/E database.	Required	2-5	
4	Allocate and define the SVAA data sets.	Required	2-8	
5	Install the SVAA functions.		2-13	
5a	Receive the SVAA functions.	Required	2-14	
5b	Receive the Maintenance PTFs and HOLDDATA.	Required	2-15	
5c	List the HOLDDATA for the SVAA SYSMODs.	Required	2-16	
5d	Apply the SVAA functions.	Required	2-17	
5e	Accept the SVAA functions.	Required	2-18	
5f	Apply the remaining Maintenance PTFs.	Required	2-19	
5g	Accept the remaining Maintenance PTFs.	Required	2-20	
5h	Create the SVAA Server directories.	Optional	2-21	
5i	Define the SVAA Server path names.	Optional	2-23	
5j	Apply the SVAA Server function.	Optional	2-24	
5k	Accept the SVAA Server function.	Optional	2-25	
6	Install the Multi-level Operating System Support.	Optional	2-26	

Table 1-1 (Page 2 of 3). *Installation checklist*

Step	Description of action	Required or optional?	Page	√
7	Install usermods.		2-29	
7a	Install the JES Offset Table support usermods.	Optional	2-29	
7b	Install the SVAA SIBCTUM usermod.	Optional but recommended	2-29	
8	Customize SYS1.PARMLIB for SVAA.		3-2	
8a	Customize the LNKLSTxx or PROGxx member.	Required	3-2	
8b	Customize the IEAAPFxx or PROGxx member.	Required	3-2	
8c	Customize the IECIOSxx member.	Required	3-3	
8d	Customize the IEFSSNxx member.	Required	3-4	
8e	Customize the SMFPRMxx member.	Required	3-8	
8f	Customize the MPFLSTxx member.	Required	3-10	
9	Define parmlib members for the SVAA started task.		3-11	
9a	Define the SIBSYS*xx* member.	Required	3-11	
9b	Define the SIBDSR*xx* member.	Required	3-13	
9c	Define the SIBSFC*xx* member.	Optional	3-14	
9d	Define the SIBSIM*xx* member.	Required	3-14	
9e	Define the SIBSRP*xx* member.	Required	3-15	
9f	Define the SIBVOP*xx* member.	Required	3-16	
10	Define the SIBSTK*xx* parmlib member.	Required	3-17	
11	Verify library requirements.		3-18	
11a	Verify SIBLINK library requirements.	Required	3-18	
11b	Verify SIBLOAD library requirements.	Required	3-18	
11c	Verify STKLOAD library requirements.	Required	3-19	
11d	Verify SACLINK library requirements.	Required	3-20	
12	Create the SVAA started-task JCL.	Required	3-21	
13	Create the SVAASRVR started-task JCL.	Optional	3-25	

Table 1-1 (Page 3 of 3). *Installation checklist*

Step	Description of action	Required or optional?	Page	√
14	Customize the TSO environment.		3-26	
14a	Allocate SVAA data sets for TSO.	Optional	3-26	
14b	Allocate SVAA data sets for ISPF	Optional	3-27	
14c	Set up to run SVAA under ISPF	Optional	3-33	
14d	Customize the TSO session.	Optional	3-34	
15	Define security resources to your security system.	Required	3-35	
16	Customize SVAA user exits.	Optional	3-36	
17	Customize the logging facility.	Optional	3-38	
18	Enable SYS1.PARMLIB changes,	Required	4-2	
19	Initialize SVAA with SIBMVSS.	Required	4-4	
20	Verify SVAA subsystem initialization.	Required	4-7	
21	Verify SVAA under TSO.	Optional	4-8	
22	Verify SVAA under ISPF.	Optional	4-9	
23	Verify SVAA under SIBBATCH.	Optional	4-10	
24	Verify SVAA using Shared Virtual Array Console	Optional	4-11	

Migration Considerations

If you use PPRC and you are upgrading an existing V2X or V2X2 subsystem to a V2Xf, you should be aware that Standard PPRC and PowerPPRC are not supported on the V2Xf. PPRCFCN replaces Standard PPRC and PowerPPRC. In addition, SCSI devices are no longer supported. Carefully review StorageTek's *V2Xf Shared Virtual Array Peer-to-Peer Remote Copy Configuration Guide* before beginning the hardware upgrade.

PowerPPRC Bridge Devices and SCSI Devices

Since PowerPPRC Bridge devices and SCSI devices are no longer supported, you must perform the following before first starting the hardware upgrade to a V2Xf:

1. Delete all non-Bridge PPRC pairs on the affected V2X or V2X2
2. Delete all Bridge PPRC pairs on the affected V2X or V2X2
3. Delete all PPRC paths on the affected V2X or V2X2

4. Delete all Bridge devices
5. Delete all SCSI devices

Note: Failure to delete all Bridge and SCSI devices before the hardware upgrade will result in the inability to IML the V2Xf microcode. A backout of the hardware upgrade, deletion of all Bridge and SCSI devices, and a repeat of the hardware upgrade is then required to recover from this condition.

System Adapter IDentifier (SAID) Values

The System Adapter IDentifier (SAID) values have changed for the V2Xf. These values are specified in the LINK parameter of the CESTPATH command. These changed values are documented in the *V2Xf Shared Virtual Array Peer-to-Peer Remote Copy Configuration Guide*

For More Information

For more information about PPRCFCN, refer to the section in this publication entitled *How to Obtain Software Documentation* or to the following publication located on the StorageTek Customer Resource Center's (CRC) web site at **www.support.storagetek.com**:

- *V2Xf Shared Virtual Array Peer-to-Peer Remote Copy Configuration Guide*

Software Prerequisites

The OS/390 version of SVAA requires **minimum** software releases as listed in the table below.

Table 1-2. *OS/390 System Software Prerequisites*

Software Description	Minimum Version/Release
Operating System -- with SVAA server	
z/OS (including UNIX System Services)	Version 1.4
Java (full version)	JRE 1.3
Operating System -- without SVAA server	
z/OS	Version 1.4
Job Entry Subsystem	
OS/390 JES2	Version 2.10
OS/390 JES3	Version 2.10
Dialog Support	
ISPF	Version 3.5
TSO/E	Release 2.0
Report Software	
SAS	Version 6.07
Installation Software	
Assembler	Assembler H
SMP/E	Release 8.1 or OS/390 SMP/E

Note: The SVAA Dynamic DDSR function no longer uses the IGGPRE00 hook when IBM APARs OW29642 and OW31787 are applied.

Special Considerations

This section discusses special considerations regarding the upgrade from IXFP, elimination of the JES Offset Tables, and the support of multiple levels of OS/390 operating systems.

Steps for Upgrading from IXFP

The following steps are required to upgrade from IXFP to SVAA:

- Install SVAA with SMP/E

  Two installation options are available:

Method 1	Install SVAA in its own newly allocated SMP/E CSI.
Method 2	Install SVAA in an existing SMP/E CSI but in new SMP/E target and distribution zones.

- Stop the current IXFP address space.
- Terminate the current IXFP subsystem using SIBMVSS. If you are using Dynamic DDSR, include the DX option of the ALTER SSNAME TERMinate command.
- Remove IXFP modules (SIBLINK, SIBLOAD, and STKLOAD) from the link list.
- Add the SVAA modules to the link list.
- Refresh the link list.
- Initialize the SVAA subsystem using SIBMVSS.
- Start the SVAA address space.

JES Offset Table Elimination

Applicability and Restrictions

The JES2 Offset Table (module SIBRSLV2) and JES3 Offset Table (module SIBRSLV3) are no longer required for MVS 4.3 and above systems, with the following restrictions:

- If the SVAA subsystem is initialized ***before*** JES initialization, the name of the JES (Primary Subsystem) must be “JES2” or “JES3” otherwise the SIBRSLV2 or SIBRSLV3 offset tables are required.

  If the SVAA subsystem is initialized ***after*** JES initialization, the JES Offset Tables are not required and the name of the JES can be anything that is allowed by IBM naming conventions.

- If either or both the SIBRSLV2 and SIBRSLV3 offset tables are present, they must be assembled at the correct level for the current system.

Note: If a JES Offset Table is present and does not match current system levels, a SIB2940E message is issued. If a JES Offset Table is required but is not present, a SIB2919E message is issued. No message is issued that indicates that a JES Offset Table is successfully bypassed, or is present and is in use.

The relationships between operating system environments and offset table requirements are summarized in the following table.

Table 1-3. *JES Offset Table requirements by environment*

Environment	JES2 Offset Table required?	JES3 Offset Table required?
All systems JES2 and MVS/ESA 4.3 or above	No	No
All systems JES2, some below MVS/ESA 4.3	Yes—only for JES2 systems below MVS/ESA 4.3	No
Mixed JES2 and JES3 systems, all systems at MVS/ESA 4.3 or above	No	No
Mixed JES2 and JES3 systems, some JES2 and JES3 systems below MVS/ESA 4.3	Yes—only for JES2 systems below MVS/ESA 4.3	Yes—only for JES3 systems below MVS/ESA 4.3
All systems JES3 and MVS/ESA 4.3 or above	No	No

Installation Process Simplification

Subject to the above applicability and restrictions, the following SVAA installation tasks are not required:

- SMP/E installation of the TSIB311 function for JES2 and/or the TSIB312 function for JES3
- SMP/E installation of the dummy base and TSIB311 and/or TSIB312 functions for Multi-Level Operating System Support (MLSS)
- SMP/E installation of the Offset Table Support Usermods

Locating and Eliminating SIBRSLV2 and SIBRSLV3

To fully realize the benefits of JES Offset Table elimination it is necessary to delete any copies of modules SIBRSLV2 and SIBRSLV3 from the following locations:

- The linklist concatenation (then refresh LLA using a F LLA,REFRESH command, and when refreshed, follow with a SETPROG LNKLST,UPDATE,ASID=01 command to update the Master Scheduler address space)
- Any library in the STEPLIB concatenation used to initialize the SVAA subsystem (JCL using SIBMVSS)
- Any library in the STEPLIB concatenation used to initialize the SVAA address space (JCL using SIBMAIN.)

Multi-level Operating System Support (MLSS)

The JES Offset Table functions (TSIB311 and TSIB312), use JES and DFP macros to create non-executable load modules (SIBRSLV2 and SIBRSLV3) that contain information needed for the proper operation of SVAA. If any of the IBM macros used for this information change (by new release, PTF, or APAR), these tables may not be usable and may have to be reinstalled.

If you need to run SVAA on multiple operating systems using different levels of OS/390, DFP, and/or JES2/JES3, it may be necessary to install TSIB311 or TSIB312 separately for each of those operating systems. If this applies to you, follow the instructions in “Step 6: Installing Multi-level Operating System Support” on page 2-26 for running the “M” set of jobs.

Note: It may not be necessary to install the JES Offset Tables. Refer to “JES Offset Table Elimination” on page 1-6 before running the “M” set of jobs.

Step 1: Getting Started

You can proceed with the SMP/E installation of SVAA before the SVA hardware is installed. However, before you can test SVAA subcommands that access the SVA subsystem, you must do two things, one of which requires the SVA hardware:

Step 1a: Define the SVA Subsystem to OS/390

For complete instructions on how to define the SVA subsystem devices to OS/390, refer to the *V2Xf Shared Virtual Array: Planning* guide.

Step 1b: Perform the Minimum SVA Configuration

Note: To accomplish this configuration, your SVA hardware must be installed.

A minimum SVA configuration must be completed at the Local Operator Panel (LOP) or Detached Operator Panel (DOP) before SVAA can communicate with the SVA subsystem.

The minimum configuration requires:

- Forming at least one production array.
- Defining at least one subsystem identifier (SSID) within the valid SSID ranges. The SSID must be *unique* among all your DASD subsystem IDs.
- Defining at least one privileged Extended Control and Monitoring (ECAM) device (described in Appendix A, “SVAA Communication Devices” in this manual).
- Configuring and enabling at least one channel that is accessible to the OS/390 system.

Each SVA subsystem must have a *unique* subsystem name.

See the *V2Xf Shared Virtual Array: Operation and Recovery* manual for Local Operator Panel or Detached Operator Panel configuration procedures.

Chapter 2. Installing SVAA

The SMP/E installation process consists of the following steps:

Step 2	Loading the Installation JCL (from the Maintenance Tape)
Step 3	Building the SMP/E Database
Step 4	Allocating and Defining the SVAA Data Sets
Step 5	Installing the SVAA Functions (from the Product Tape)
Step 6	Installing Multi-level Operating System Support
Step 7	Installing Usermods

Two tapes are required in the SVAA installation process: the SVAA Product Tape and the SVAA Maintenance Tape. The two tapes are described on the following page.

SVAA Product Tape Contents

The SVAA Version 3.1 Product Tape is a 3480 standard-label tape with the volume serial number SIB310. Table 2-1 lists the tape's contents.

Table 2-1. *SVAA Product Tape contents*

File No.	Data Set Name	Description
1	SMPMCS	SVAA SMP/E control statements
2	SSIB310.F1	SVAA JCLIN
3	SSIB310.F2	SVAA load modules
4	SSIB310.F3	SVAA macros, REXX, ISPF dialogs
5	TSIB311.F1	JES2 Offset Table JCLIN
6	TSIB311.F2	JES2 Offset Table source
7	TSIB312.F1	JES3 Offset Table JCLIN
8	TSIB312.F2	JES3 Offset Table source
9	SSNI210.F1	SVAA LLAPI JCLIN
10	SSN1210.F2	SVAA LLAPI load modules
11	SSKP124.F1	Common Parser JCLIN
12	SSKP124.F2	Common Parser load modules
13	ASAR65D.F1	SAS/C resident library JCLIN
14	ASAR65D.F2	SAS/C resident library load modules
15	ASAT65D.F1	SAS/C all-resident library JCLIN
16	ASAT65D.F2	SAS/C all-resident library load modules
17	SSAT65D.F1	SAS/C transient library JCLIN
18	SSAT65D.F2	SAS/C transient library load modules
19	SSJC110.F1	SVAA Server load modules
20	SMPEJCL	SMP/E installation JCL

SVAA Maintenance Tape Contents

The SVAA Maintenance Tape is a 3480 non-label tape. Table 2-2 lists the tape's contents.

Table 2-2. *SVAA Maintenance Tape contents*

File No.	Data Set Format	Description
1	SMPPTFIN	All PTFs issued since Base tape
2	IEBGENER	Print file of all PTF cover letters
3	IEBGENER	Print file of summary info for all PTFs
4	SMPPTFIN	SMP/E external hold statements
5	IEBUPDTE	Current installation JCL

Step 2: Loading the Installation JCL

To install SVAA, you must use the current installation JCL, which is in:
file 5 on the SVAA Maintenance Tape. (**Do not** use file 20 on the SVAA Product Tape.)

Modify the following job to load the installation JCL file from the Maintenance Tape to a data set on your system:

- For *mnttap*, enter the volume serial number that appears on the external label of the Maintenance Tape.
- For *tapeunit*, enter the unit number of the drive on which the tape is mounted.
- Change the SYSUT2 **DSN**, **UNIT**, and **BLKSIZE** fields as needed.

```
//jobname   JOB    'accounting info'
//*
//STEP1     EXEC   PGM=IEBUPDTE,PARM=NEW
//SYSPRINT  DD     SYSOUT=*
//SYSIN     DD     DSN=INSTJCL,DISP=OLD,
//                 VOL=SER=mnttap,
//                 UNIT=tapeunit,
//                 LABEL=(5,NL,EXPDT=98000),
//                 DCB=(RECFM=FB,LRECL=80,BLKSIZE=7200)
//*
//SYSUT2    DD     DSN=data.set.name,DISP=(,CATLG),
//                 UNIT=sysallda,SPACE=(TRK,(10,2,10)),
//                 DCB=(RECFM=FB,LRECL=80,BLKSIZE=23440)
```

Figure 2-1. *Load JCL*

JCL members $$NOTES and $$TOC are shown below.

$$NOTES: Notes for Using SVAA Installation JCL

This library contains sample JCL for installing the SVAA functions and Maintenance PTFs with SMP/E. Member $$TOC, the table of contents, lists the function of each member.

Follow the instructions in this chapter and in the documentation provided in each member before executing each job. Change the lowercase variables in each job to valid uppercase values. Execute the jobs in the order given.

There is an extensive amount of editing necessary to use the JCL. An optional edit macro (member $EDJCL) is provided to help with the editing. Follow the instructions in the edit macro for its use.

For existing IXFP customers, the SVAA $EDJCL macros differ significantly from the previous IXFP $IXEDIT macros. Do NOT use the IXFP $IXEDIT1 or $IXEDIT2 macro to make SVAA installation JCL changes.

Instructions within each job contain either notes, or a list of changes to make. Anything within a note will not be changed by the $EDJCL edit macro and must be considered separately before running the job. Notes will always begin with **Note:**.

$$TOC: SVAA Installation JCL Table of Contents

$$NOTES	Notes on using the installation JCL.
$$TOC	Table of contents.
$EDJCL	Edit macro used to modify installation JCL (optional).
$EDRUN	REXX EXEC to invoke $EDJCL against all members of the SVAA Installation JCL pds (optional).
$METHODS	Description of Installation Methods 1 and 2.
$NEW	Changes to Installation JCL for current Maintenance level.

SMP/E Database Build

C1CSIBLD	Allocate and initialize the SMP/E data sets. (Installation Method 1 only).
C2SMPLOG	Allocate and initialize the SMP/E log data sets.
C3ZONES1	Define zones and zone options. (Installation Method 1 only).
C4ZONES2	Define zones and zone options. (Installation Method 2 only).
C5DDDEFS	Define the required DDDEFs for the global zone. (Installation Method 1 only).
C6DDDEFS	Define the DDDEFs for the target and distribution zones.

Function Installation

D1ALLOC	Allocate and define the SVAA data sets.
D2ALLOC	Allocate and define the SAS/C data sets.
D3ALLOC	Allocate and define the Common Parser data sets.
D4ALLOC	Define the DDDEFs for the operating system.
I1RCVFUN	Receive the SVAA and support functions.
I2RCVPTF	Receive the SVAA PTFs and HOLDDATA.
I3HOLD	List the HOLDDATA for the SVAA SYSMODs (optional).
I4APPLY	Apply the SVAA functions and PTFs.
I5ACCEPT	Accept the SVAA functions and PTFs.
I6PTFAPP	Apply any PTFs not previously applied.
I7PTFACC	Accept any PTFs not previously accepted.
J0PATHS	Create the SVAA Server directories on USS.
J1DDDEFS	Allocate and define the SVAA Server DDDEFs.
J2APPLY	Apply the SVAA Server function.
J3ACCEPT	Accept the SVAA Server function.

Multi-Level Operating System Support

M0NOTE	Important note.
M1CSI	Update the SVAA global CSI and create the new zones.
M2ALLOC	Allocate SVAA data sets and define DDDEFs.
M3APPLY	Apply JES Offset Table support function(s).
M4ACCEPT	Accept JES Offset Table support function(s).
M5PTFAPP	Apply PTFs to the JES2/3 function.
M6PTFACC	Accept PTFs to the JES2/3 function.

Maintenance Installation

P1RCV	Receive the Maintenance PTFs.
P2APPLY	Apply the Maintenance PTFs.
P3ACCEPT	Accept the Maintenance PTFs.

Step 3: Configuring the SMP/E Database

Configuring the SMP/E database consists of the following tasks:

- Defining either a new SMP/E CSI for SVAA, or new target and distribution zones in an existing SMP/E CSI.
- Defining the global zone requirements.
- Defining the SYSLIB concatenation.
- Defining the SVAA DDDEFs.

For each SMP/E task in this section, you can use the Installation JCL library member, the SMP/E Sysmod Management dialogs, or, where available, the printed sample JCL to modify the SMP/E database.

Installation Method Options

SVAA must be installed in its own SMP/E CSI, or in its own target and distribution zone in an existing SMP/E CSI, because of the following limiting factors:

- Because of significant functional repackaging of SAS/C and SVAA, **you cannot install SVAA in the same SMP/E target and distribution zone as IXFP or other products that use different levels of SAS/C.**
- Because SAS/C modules are provided with SVAA, **you cannot install SVAA in the same SMP/E target and distribution zone that contains program products that make use of the SAS/C runtime and/or transient libraries.**
- There is a possibility of name collision between SVAA module names and the names of modules owned by other vendors' products. SMP/E requires that each element name be unique.

There are two methods of installing the SVAA functions:

Method 1 Install SVAA in its own newly allocated SMP/E CSI.

Method 2 Install SVAA in an existing SMP/E CSI but in new SMP/E target and distribution zones.

Certain jobs used to build the SMP/E database will need to be modified or not run for Installation Method 2. The following table shows the jobs that are affected:

Table 2-3. *Installation Methods*

Job Name	Description	Method 1 (new CSI)	Method 2 (existing CSI)
C1CSIBLD	Allocate and initialize SMP/E data sets	run	do not run
C2SMPLOG	Allocate the SMP/E log data sets	run	conditional
C3ZONES1	Define zones and zone options for Method 1	run	do not run
C4ZONES2	Define zones and zone options for Method 2	do not run	run
C5DDDEFS	Define the required DDDEFs for the global zone for Method 1	run	do not run
C6DDDEFS	Define the DDDEFs for the target and distribution zones for Methods 1 and 2	run	run

Defining the SMP/E CSI

Create the SMP/E CSI

JCL member **C1CSIBLD** defines and initializes the SMP/E CSI and allocates the SMPPTS, SMPSTS, and SMPSCDS data sets. Do not run this job if you are using Installation Method 2 (installing to an existing CSI).

Allocate the SMPLOG Data Sets

JCL member **C2SMPLOG** allocates the SMPLOG and SMPLOGA data sets.

If you are using Installation Method 1 you must allocate the log data sets.

If you are using Installation Method 2 you have the option of using either the same SMPLOG and SMPLOGA data sets as used by your current global zone, or using new log data sets. You can use job C2SMPLOG to allocate new log data sets.

If you choose to use new (different) log data sets, you need to add a DD statement for the SMPLOG and SMPLOGA data sets each time you execute SMP/E in batch.

GLOBAL Zone Requirements

Several parameters within the SMP/E global zone must be set at certain or minimum values to ensure adequate space for the SMP/E Receive, Apply, and Accept processing.

Note: To set the parameters described below, run:

- JCL member **C3ZONES1** if you are installing with Method 1, or
- JCL member **C4ZONES2** if you are installing with Method 2.

Linkage Editor Parameters

Run JCL member **C3ZONES1** or **C4ZONES2**. The NAME parameter can be omitted. This will cause SMP/E to choose the default linkage editor. When the DFP level supports the binder, IEWBLINK is the default link-edit utility. When the DFP level does not support the binder, IEWL is the default.

X37 Error Recovery

Run JCL member **C3ZONES1** or **C4ZONES2**. Update the error recovery parameter in your global zone options with RETRYDDN(ALL). This will cause the Error Recovery utility to attempt to recover from all X37 abends when the Apply or Accept job includes the RETRY(YES) parameter.

SMPTLIB DDDEF

Run JCL member **C3ZONES1** or **C4ZONES2**. The SMPTLIB DDDEF parameters must provide adequate space for the SMP/E Receive process. This is defined in two locations within the SMP/E global zone.

Verify that the SMPTLIB space parameters in your global zone OPTIONS DSSPACE/DSPREFIX entry have the following minimum allocations:

PRIMARY TRACKS:	250
SECONDARY TRACKS:	150
DIRECTORY BLOCKS:	250

Verify that the SMPTLIB space parameters in your global zone DDDEF entry have the following minimum allocations:

SPACE UNITS:	TRACKS
PRIMARY SPACE:	250
SECONDARY SPACE:	150
DIR:	250

NOPURGE SMP/E Global Zone Option

Run JCL member **C3ZONES1** or **C4ZONES2**.

You should use the **NOPURGE** SMP/E global zone option when you run the Accept for TSIB311 and/or TSIB312. If you do not use this option you will need to re-receive the function if it should need to be reinstalled. NOPURGE specifies that after SMP/E accepts SYSMODs, it should not delete the associated global zone SYSMOD entries, HOLDDATA entries, SMPPTS MCS entries, or SMPTLIB data sets.

Global Zone DDDEFs

For Method 1 only, run installation JCL member **C5DDDEFS** to define the required DDDEFs for the global zone.

Target and Distribution Zone DDDEFs

For both Method 1 and Method 2, run installation JCL member **C6DDDEFS** to define the DDDEFs for the target and distribution zones.

Step 4: Allocating and Defining the SVAA Data Sets

See installation JCL members **D1ALLOC**, **D2ALLOC**, **D3ALLOC**, and **D4ALLOC**.

D1ALLOC, **D2ALLOC**, and **D3ALLOC** create the data sets and add DDDEF entries for each data set in the target and distribution zone of the CSI.

Allocating Data Sets

The following tables describe the attributes and the actual number of used directory blocks and space allowances for the SVAA target and distribution data sets. The JCL in jobs D1ALLOC, D2ALLOC, and D3ALLOC give the recommended space allowances and directory blocks, which are approximately double the figures for actual space used. Space estimates are given in 3380 tracks and in blocks.

The block sizes listed are the same as those on the Product tape. These are not necessarily the recommended block sizes. See the key to the **Recommended Block Sizes** in Table 2-4 on page 2-9 and Table 2-5 on page 2-10. If you change the block size, you may want to change the number of blocks. The values given in the tables are the same as those in JCL members D1ALLOC, D2ALLOC, and D3ALLOC.

Defining SVAA Data Sets

Run the **D1ALLOC**, **D2ALLOC**, and **D3ALLOC** jobs. If you do not wish to use DDDEFs for the SVAA data sets, you will need to include a DD statement for each SVAA data set within your SMP/E PROC or JCL for any SVAA SMP/E tasks to be done.

Update your target and distribution zone DDDEFs with the SVAA data set names. Change "hlq" to an appropriate high level data set name qualifier, and correct the name of the target and distribution zones in the SET BOUNDARY statements. The SYS1.PARMLIB DDDEFs are needed if you install the JES Offset Table functions.

Table 2-4. *Distribution Data Sets*

Data Set Name	DSORG	RECFM	LRECL	Block Size	Used Blocks	Used 3390 Tracks	Used Dir Blocks	Note	Description
hlq.ASACLINK	PO	U	0	15476	284	95	28	-	SAS/C transient
hlq.ASAROAM	PO	U	0	6144	66	8	12	-	SAS/C resident
hlq.ASAROBM	PO	U	0	6144	291	33	56	-	SAS/C resident
hlq.ASAROMM	PO	U	0	6144	358	42	63	-	SAS/C resident
hlq.ASAROSM	PO	U	0	6144	406	46	76	-	SAS/C resident
hlq.ASAROTM	PO	U	0	6144	260	29	31	-	SAS/C resident
hlq.ASIBASM	PO	FB	80	3120	1	1	1	-	SVAA assembler source
hlq.ASIBCLIB	PO	FB	80	3120	241	16	3	-	SVAA REXX EXECS
hlq.ASIBLOAD	PO	U	0	6144	1477	165	148	-	SVAA modules
hlq.ASIBMAC	PO	FB	80	3120	32	16	1	-	SVAA assembler macros
hlq.ASIBMLIB	PO	FB	80	3120	49	4	2	-	SVAA ISPF messages
hlq.ASIBPLIB	PO	FB	80	3120	585	37	21	-	SVAA ISPF panels
hlq.ASIBSAMP	PO	FB	80	3120	301	19	4	-	SVAA sample
hlq.ASIBSAS	PO	FB	80	3120	559	35	6	-	SVAA SAS programs
hlq.ASIBSLIB	PO	FB	80	3120	30	2	1	-	SVAA ISPF skeletons
hlq.ASIBTABL	PO	FB	80	3120	2	1	1	-	SVAA ISPF table
hlq.ASIBTLIB	PO	FB	80	3120	2	1	1	-	SVAA ISPF table
hlq.ASJCHFS	PO	V	8192	8196	1432	238	1	-	SVAA server
hlq.ASKPRTNS	PO	U	0	6144	56	7	6	-	Common Parser 1.2.4

Note: The distribution library for SIBLINK and SIBRTNS is ASIBLOAD.

Note: It is recommended that the **block size** for each data set be modeled after a similar system data set on your system.

Table 2-5. *Target Data Sets*

Data Set Name	DSORG	RECFM	LRECL	Block Size	Used Blocks	Used 3390 Tracks	Used Dir Blocks	Note	Description
hlq.SACLINK	PO	U	0	23476	190	95	28	2	SAS/C transient library
hlq.SIBASM	PO	FB	80	3120	1	1	1	-	SVAA assembler source
hlq.SIBCLIB	PO	FB	80	3120	241	16	3	4	SVAA REXX EXECS
hlq.SIBLINK	PO	U	0	23476	46	23	12	2	SVAA load modules
hlq.SIBLOAD	PO	U	0	23476	616	308	26	2	SVAA load modules
hlq.SIBMAC	PO	FB	80	3120	32	13	1	3	SVAA assembler macros
hlq.SIBMLIB	PO	FB	80	3120	49	4	2	1	SVAA ISPF messages
hlq.SIBPLIB	PO	FB	80	3120	585	37	21	1	SVAA ISPF panels
hlq.SIBRTNS	PO	U	0	6144	1289	144	134	-	SVAA routines library
hlq.SIBSAMP	PO	FB	80	3120	301	19	4	-	SVAA sample library
hlq.SIBSAS	PO	FB	80	3120	559	35	6	-	SVAA SAS programs
hlq.SIBSLIB	PO	FB	80	3120	30	2	1	1	SVAA ISPF skeletons
hlq.SIBTABL	PO	FB	80	3120	2	1	1	1	SVAA ISPF table
hlq.SIBTLIB	PO	FB	80	3120	2	1	1	1	SVAA ISPF table
hlq.SSAROMOD	PO	U	0	6144	1382	154	237	-	SAS/C resident library
hlq.SKPRTNS	PO	U	0	6144	55	7	6	-	Common Parser 1.2.4
hlq.STKLOAD	PO	U	0	23476	12	6	1	2	Common Load library

Note: Recommended Block Sizes

It is recommended that the block size for each data set be modeled after a similar system data set on your system. Use the value in the "Note" column in the above table to find the model data set name.

Note 1: Same as ISPF's ISPPLIB, ISPMLIB, ISPTLIB, ISPSLIB, etc.
Note 2: Same as SYS1.LINKLIB.
Note 3: Same as SYS1.MACLIB.
Note 4: Same as CLIST or SYSEXEC Library. If your current SYSPROC data sets are allocated with RECFM=VB you will need to convert the SIBCLIB library to a second data set that is variable blocked before concatenating with your SYSPROC data sets. The original SIBCLIB target data set should remain fixed block for use by SMP/E.

SYSLIB Concatenation

The last step, before installing the SVAA functions, is to verify the SYSLIB concatenation.

The **D4ALLOC** job defines DDDEFs for the operating system including the SYSLIB macro data sets and SYS1.PARMLIB.

Before submitting this job, review the section “Multi-level Operating System Support (MLSS)” on page 1-7 or JCL member M0NOTE, and review “JES Offset Table Elimination” on page 1-6.

Follow either Scenario 1 or Scenario 2, depending upon whether you are installing the JES Offset Table functions TSIB311 and/or TSIB312.

Scenario 1: You are NOT installing JES Offset Table functions

In the target and distribution zones in the SET BOUNDARY statements in the **D4ALLOC** job:

1. Remove the PARMLIB and the jesmac DDDEFs from the target and distribution zones below.
2. Remove the jesmac DDDEFs from the SYSLIB concatenations in the target and distribution zones below.

Scenario 2: You ARE installing JES Offset Table functions

In the target and distribution zones in the SET BOUNDARY statements in the **D4ALLOC** job:

1. With OS/390, IBM has made changes to the macro library names for JES2 and JES3. Determine which JES2 or JES3 macro library you are using and make the indicated changes manually. Also, verify the correct unit and volser.
   - For JES2, change jesmac to HASPSRC or SHASMAC.
   - For JES3, change jesmac to AJES3MAC or AIATMAC.

   If you have both JES2 and JES3, duplicate the jesmac DDDEFs (2) for the second JES macro library.
2. Verify that the SYS1.PARMLIB data set has the correct high level qualifier, volume, and unit.

   If you already have SYS1.PARMLIB defined to these zones you may not want to redefine it. The ADD will keep it from being defined unintentionally.
3. Verify *mtshlq*.SMPMTS for the correct unit and volser. This is the SMPMTS used by the OS/390 operating system. It is required for the assembly of the JES2 or JES3 Offset Table load modules, which use IBM macros that may have been applied but not yet accepted and therefore do not have an associated target library. SVAA does not have its own SMPMTS data set.

In ALL cases:

- Verify that the following system data sets have the correct high level qualifier, volume, and unit:

  AGENLIB, AMACLIB, AMODGEN, MACLIB, and MODGEN

- Change the SYSLIB DDDEFs to match your requirements. Don't change the ADD to a REP unless you intend to completely replace your SYSLIB concatenation.

  Each REP DDDEF statement will get a return code 04 the first time this job is executed.

The SYSLIB Concatenation:

```
mtshlq.SMPMTS
SYS1.SHASMAC   (for TSIB311 - JES2 Offset Table function)
SYS1.AIATMAC   (for TSIB312 - JES3 Offset Table function)
SYS1.AGENLIB
SYS1.AMACLIB
SYS1.AMODGEN
SYS1.MACLIB
hlq1.SIBMAC
```

Step 5: Installing the SVAA Functions and PTFs

The SVAA function installation consists of the following tasks:

- Receiving the SVAA functions (from the SVAA Product Tape)
- Receiving the SVAA Maintenance PTFs and HOLDDATA (from the SVAA Maintenance Tape)
- Listing and resolving HOLDDATA
- Applying the SVAA functions
- Accepting the SVAA functions
- Applying any PTFs not previously installed
- Accepting any PTFs not previously installed
- Creating the SVAA Server directories
- Defining the SVAA Server path names
- Applying the SVAA Server function
- Accepting the SVAA Server function

For each step in the installation process you can use the Installation JCL library member, the SMP/E Sysmod Management dialogs, or the printed sample JCL to install the SVAA programs into the appropriate SMP/E zone.

SVAA SMP/E Function IDs

The SVAA software product is packaged in standard SMP/E format along with selected components of the SAS/C compiler. SVAA Version 3.1 and SAS/C have the following SMP/E function IDs (FMIDs):

SSIB310	SVAA Base function
TSIB311	JES2 Offset Table function
TSIB312	JES3 Offset Table function
SSNI210	SVAA LLAPI function
SSJC110	SVAA Server function
SSKP124	Common Parser 1.2.4 Base function
ASAR65D	SAS/C 6.5.0 resident library base function
SSAT65D	SAS/C 6.5.0 transient library base function
ASAT65D	SAS/C 6.5.0 all-resident library base function

Step 5a: Receiving the SVAA Functions

Use the sample JCL in **I1RCVFUN** to receive the SVAA functions from the Product Tape.

Review the notes in the sample JCL.

The SVAA Server function (SSJC110) has the following software prerequisites:

- OS/390 Version 2 Release 10 or higher
- SMP/E Version 1 Release 8.1 or OS/390 SMP/E

If you are not at these software levels, comment out the SSJC110 function in the I1RCVFUN JCL.

```
//jobname   JOB    'accounting info'
//*
//          EXEC   smpeproc
//*
//SMPPTFIN  DD     DISP=(OLD,PASS),
//                 VOL=(,RETAIN,SER=SIB310),
//                 UNIT=tapeunit,
//                 LABEL=(1,SL,EXPDT=98000),
//                 DSN=SMPMCS
//SMPCNTL   DD     *

 SET BOUNDARY ( GLOBAL ) .

 RECEIVE SELECT (  SSIB310     /* SVAA Base function         */
                   SSNI210     /* SVAA LLAPI function        */
                   SSJC110     /* SVAA Server function       */
                   TSIB311     /* JES2 Offset Table function */
                   TSIB312     /* JES3 Offset Table function */
                   SSKP124     /* Common Parser 1.2.4        */
                   ASAR65D     /* SAS/C 6.5.0 Resident       */
                   SSAT65D     /* SAS/C 6.5.0 Transient      */
                   ASAT65D     /* SAS/C 6.5.0 All-Resident   */
               )
         SYSMODS
         .
```

Figure 2-2. *JCL member I1RCVFUN: Receiving the SVAA functions*

When the I1RCVFUN job has completed successfully, proceed to “Step 5b: Receiving the Maintenance PTFs and HOLDDATA” on page 2-15 (I2RCVPTF).

Step 5b: Receiving the Maintenance PTFs and HOLDDATA

Use the sample JCL in **I2RCVPTF** to receive the PTFs and HOLDDATA from the Maintenance tape.

```
//jobname   JOB    'accounting info'
//*
//          EXEC   smpeproc
//*
//SMPPTFIN  DD     DSN=PTF,DISP=SHR,
//                 VOL=(,RETAIN,SER=mnttap),
//                 UNIT=tapeunit,
//                 LABEL=(1,NL,EXPDT=98000),
//                 DCB=(RECFM=FB,LRECL=80,BLKSIZE=7200)
//SMPHOLD   DD     DSN=HOLDDATA,DISP=SHR,
//                 VOL=(,RETAIN,SER=mnttap),
//                 UNIT=AFF=SMPPTFIN,
//                 LABEL=(4,NL,EXPDT=98000),
//                 DCB=(RECFM=FB,LRECL=80,BLKSIZE=7200)
//SMPCNTL   DD     *

 SET BOUNDARY ( GLOBAL ) .

 RECEIVE SYSMODS             /* Receive all SYSMODs and    */
         HOLDDATA            /* any exception data.  List  */
         LIST                /* the MCS for each SYSMOD.   */
         .
```

Figure 2-3. *JCL member I2RCVPTF: Receiving the PTFs and HOLDDATA*

When the I2RCVPTF job has completed successfully, proceed to “Step 5c: Listing the HOLDDATA for the SVAA SYSMODs” on page 2-16 (I3HOLD).

Step 5c: Listing the HOLDDATA for the SVAA SYSMODs

Use the sample JCL in the **I3HOLD** job to list all of the SMP/E HOLDDATA for the SVAA SYSMODs, including any that were previously received. This step is optional.

Some PTFs may have special requirements that must be met before they can be installed. These PTFs are prevented from installing with the use of SMP/E HOLDDATA. Before running the Apply job, review the output from this job, and follow the instructions given for each held PTF. Once the requirements are met, you can bypass the hold condition for that held PTF in the Apply job. Do not bypass HOLDERROR conditions. Refer to the *SMP/E Reference* manual for complete instructions on using the BYPASS parameter.

```
//jobname   JOB    'accounting info'
//*
//          EXEC   smpeproc
//*
//SMPCNTL   DD  *

 SET BOUNDARY ( GLOBAL ) .

 LIST    HOLDDATA             /* List all HOLDDATA for      */
         FORFMID (            /* these function IDs:        */
                   SSIB310    /* SVAA Base function         */
                   SSNI210    /* SVAA LLAPI function        */
                   SSJC110    /* SVAA Server function       */
                   TSIB311    /* JES2 Offset Table function */
                   TSIB312    /* JES3 Offset Table function */
                   SSKP124    /* Common Parser 1.2.4        */
                   SSAT65D    /* SAS/C 6.5.0 Transient      */
                   ASAR65D    /* SAS/C 6.5.0 Resident       */
                   ASAT65D    /* SAS/C 6.5.0 All-Resident   */
                 )
         .
```

Figure 2-4. *JCL member I3HOLD: Listing the HOLDDATA for the SVAA SYSMODs*

When the I3HOLD job has completed successfully, proceed to “Step 5d: Applying the SVAA Functions” on page 2-17.

Step 5d: Applying the SVAA Functions

Use the sample JCL in **I4APPLY** to install the SVAA functions for installation Methods 1 and 2 into the appropriate target libraries.

Review the notes in the sample JCL.

Before submitting this job, specify the name of your target zone in the SET BOUNDARY statement.

Add BYPASS options as needed after examining the SMP/E HOLDDATA. Refer to “Step 5c: Listing the HOLDDATA for the SVAA SYSMODs” on page 2-16 for further information on the BYPASS parameter.

Note: If it is not necessary to install the JES Offset Tables (refer to “JES Offset Table Elimination” on page 1-6), then leave the comment tags in place for the TSIB311 and TSIB312 lines or delete these lines from the JCL.

If it is necessary to install the JES Offset Tables, then move the leftmost comment tag (/*) to the right of each JES Offset Table function (TSIB311 and/or TSIB312) to be installed.

```
//jobname   JOB    'accounting info'
//*
//          EXEC   smpeproc
//*
//SMPLOG    DD     DISP=SHR,DSN=loghlq.SMPLOG
//SMPLOGA   DD     DISP=SHR,DSN=loghlq.SMPLOGA
//SMPCNTL   DD     *

 SET BOUNDARY ( target ) .     /* Change to your target zone */

 APPLY   SELECT (  SSIB310     /* SVAA Base function         */
                   SSNI210     /* SVAA LLAPI function        */
        /*         TSIB311        JES2 Offset Table function */
        /*         TSIB312        JES3 Offset Table function */
                   SSKP124     /* Common Parser 1.2.4        */
                   SSAT65D     /* SAS/C 6.5.0 Transient      */
                   ASAR65D     /* SAS/C 6.5.0 Resident       */
                   ASAT65D     /* SAS/C 6.5.0 All-Resident   */
                )
        GROUPEXTEND
        NOJCLINREPORT
        RETRY ( YES )
        .

//* BYPASS ( bypass options )
//* BYPASS ( HOLDDATA (ACTION DOC) )
```

Figure 2-5. *JCL member I4APPLY: Applying the SVAA functions*

Note: Expect return code 04 with Binder message IEW2454W or SMP/E message GIM23903W for link-edit processing to the SIBRTNS and SKPRTNS libraries.

When the I4APPLY job has completed successfully, proceed to “Step 5e: Accepting the SVAA Functions” on page 2-18.

Step 5e: Accepting the SVAA Functions

Use the sample JCL in **I5ACCEPT** to accept the SVAA functions. The ACCEPT CHECK option can be used as often as necessary before the actual ACCEPT process to identify SMP/E processing problems. All SMP/E-detected problems must be resolved before the SVAA functions can be successfully accepted.

Before submitting this job, specify the name of your distribution zone in the SET BOUNDARY statement.

If you applied the TSIB311 or TSIB312 function(s) in the I4APPLY step, you must accept the function(s) in this step. Move the leftmost comment tag (/*) to the right of each JES Offset Table function (TSIB311 and/or TSIB312) to be installed.

You may want to use the **NOPURGE** SMP/E global zone option when you run the Accept for TSIB311 and/or TSIB312 (see job **C3ZONES1**). If you do not use this option you will need to re-receive the function if it should need to be reinstalled. NOPURGE indicates that after SMP/E accepts SYSMODs, it should not delete the associated global zone SYSMOD entries, HOLDDATA entries, SMPPTS MCS entries, or SMPTLIB data sets.

Note: You may see the following message during Accept processing of the TSIB311 or TSIB312 functions:

```
GIM24701 SMP/E COULD NOT OBTAIN LINK-EDIT PARAMETERS FOR LOAD
MODULE aaaaaaaa IN THE bbbbbbbb LIBRARY FOR SYSMOD ccccccc.
```

This message is normal when SMP/E accepts elements into the distribution libraries for the first time.

```
//jobname   JOB    'accounting info'
//*
//          EXEC   smpeproc
//*
//SMPCNTL   DD     *

 SET BOUNDARY ( dlib ) .       /* Change to your DLIB zone   */

 ACCEPT  SELECT (  SSIB310     /* SVAA Base function         */
                   SSNI210     /* SVAA LLAPI function        */
        /*         TSIB311        JES2 Offset Table function */
        /*         TSIB312        JES3 Offset Table function */
                   SSKP124     /* Common Parser 1.2.4        */
                   SSAT65D     /* SAS/C 6.5.0 Transient      */
                   ASAR65D     /* SAS/C 6.5.0 Resident       */
                   ASAT65D     /* SAS/C 6.5.0 All-Resident   */
                )
        GROUPEXTEND
        NOJCLINREPORT
        RETRY ( YES )
        .
```

Figure 2-6. *JCL member I5ACCEPT: Accepting the SVAA functions*

Step 5f: Applying Remaining Maintenance PTFs

Use the sample JCL in job **I6PTFAPP** to apply any Maintenance PTFs not applied using GROUPEXTEND in the I4APPLY job.

If you applied the TSIB311 or TSIB312 function(s) in the I4APPLY step, you must apply any other associated Maintenance PTFs in this step. Move the leftmost comment tag (/*) to the right of each JES Offset Table function (TSIB311 and/or TSIB312) to be installed.

Add BYPASS options as needed after examining the SMP/E HOLDDATA. Refer to "Step 5c: Listing the HOLDDATA for the SVAA SYSMODs" on page 2-16 for further information about the BYPASS parameter.

```
//jobname   JOB    'accounting info'
//*
//          EXEC   smpeproc
//*
//SMPLOG    DD     DISP=SHR,DSN=loghlq.SMPLOG
//SMPLOGA   DD     DISP=SHR,DSN=loghlq.SMPLOGA
//SMPCNTL   DD     *

 SET BOUNDARY ( target ) .

 APPLY   FORFMID (            /* For these function IDs:    */
                 SSIB310      /* SVAA Base function         */
                 SSNI210      /* SVAA LLAPI function        */
         /*      TSIB311         JES2 Offset Table function */
         /*      TSIB312         JES3 Offset Table function */
                 SSKP124      /* Common Parser 1.2.4        */
                 SSAT65D      /* SAS/C 6.5.0 Transient      */
                 ASAR65D      /* SAS/C 6.5.0 Resident       */
                 ASAT65D      /* SAS/C 6.5.0 All-Resident   */
               )
         GROUPEXTEND
         RETRY ( YES )
         COMPRESS( ALL )
         .
```

Figure 2-7. *JCL member I6PTFAPP: Applying remaining PTFs*

Step 5g: Accepting Remaining Maintenance PTFs

Use the sample JCL in job **I7PTFACC** to accept any Maintenance PTFs not accepted using GROUPEXTEND in the I5ACCEPT job.

If you accepted the TSIB311 or TSIB312 function(s) in the I5ACCEPT step, you must accept any other associated Maintenance PTFs in this step. Move the leftmost comment tag (/*) to the right of each JES Offset Table function (TSIB311 and/or TSIB312) to be installed.

Add BYPASS options as needed. Use the same BYPASS options used in the I6PTFAPP job.

```
//jobname   JOB    'accounting info'
//*
//          EXEC   smpeproc
//*
//SMPLOG    DD     DISP=SHR,DSN=loghlq.SMPLOG
//SMPLOGA   DD     DISP=SHR,DSN=loghlq.SMPLOGA
//SMPCNTL   DD     *

 SET BOUNDARY ( dlib ) .

 ACCEPT  FORFMID (
                   SSIB310     /* SVAA Base function         */
                   SSNI210     /* SVAA LLAPI function        */
        /*         TSIB311        JES2 Offset Table function */
        /*         TSIB312        JES3 Offset Table function */
                   SSKP124     /* Common Parser 1.2.4        */
                   SSAT65D     /* SAS/C 6.5.0 Transient      */
                   ASAR65D     /* SAS/C 6.5.0 Resident       */
                   ASAT65D     /* SAS/C 6.5.0 All-Resident   */
                 )
        GROUPEXTEND
        RETRY ( YES )
        .
```

Figure 2-8. *JCL member I7PTFACC: Accepting remaining PTFs*

Note: When the Accept job has completed successfully, **you have not yet Applied and Accepted the SVAA Server function maintenance (FMID SSJC110).** The jobs to Apply and Accept maintenance for the SVAA Server must be executed after Step 5k (see page 2-25).

Step 5h: Creating the SVAA Server Directories

The **J0PATHS** member creates the SVAA Server directories on OS/390 UNIX Systems Services (USS).

Note: The following software prerequisites are necessary to Receive, Apply, and Accept the SVAA Server function:

- OS/390 Version 2 Release 10
- SMP/E Version 1 Release 8.1 or OS/390 SMP/E

1. Your systems administrator, responsible for UNIX Systems Services, must provide the path name that will be used for the SVAA Server directories. The Server function installs into three low level subdirectories: */bin, /install*, and */lib*. (Review this member, and member J1DDDEFS with the USS systems administrator).

   The directory:

   - cannot be an NFS (Network File System)-defined path
   - must be a local HFS
   - cannot be shared

   The path name is also defined in SVAA parmlib member SIBSFC*xx*. The value of SIBHOME in SIBSFC*xx* and the */user-defined_pathname* defined here must match. SIBSFC*xx* is described in Step 9 on page 3-11.

2. The J0PATHS job will execute the BPXBATCH program to:

   a. Create the */user-defined_pathname* directory.

   b. Change the permissions necessary to give read and write access to these directories to userids that submit SMP/E Apply and Accept jobs. Additionally, give read and execute permissions to userids that execute files installed in the SVAA */bin, /install*, and */lib* subdirectories.

      The userids must contain an OMVS segment to allow OMVS access. An OMVS segment contains OS/390 UNIX information about the user.

   c. Create the SVAA subdirectory names added to the */<user-defined_pathname>* path (remember that all UNIX commands are case sensitive).

   Output of BPXBATCH steps can be found in:

   ```
   /tmp/SVAAinstall.J0PATHSout1
   /tmp/SVAAinstall.J0PATHSout2
   /tmp/SVAAinstall.J0PATHSerr1
   /tmp/SVAAinstall.J0PATHSerr2
   ```

   Instream control statements for STDENV DD must start in column one (1).

```
//jobname   JOB    'accounting info'
//*
//STEP1   EXEC PGM=BPXBATCH,
// PARM='SH cd /;$ZA $ZD;cd $ZD;$ZA $ZI $ZB $ZL;pwd;ls -lR;printenv'
//*
//STDENV   DD *
ZA=mkdir
ZD=/user-defined_pathname
ZI=install
ZB=bin
ZL=lib
//STDOUT   DD PATH='/tmp/SVAAinstall.J0PATHSout1',
//            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=SIRWXU
//STDERR   DD PATH='/tmp/SVAAinstall.J0PATHSerr1',
//            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=SIRWXU
//*
//STEP2   EXEC PGM=BPXBATCH,COND=(0,LT),
// PARM='SH cd $ZD;$ZA $ZP $ZI $ZB $ZL;pwd;ls -lR;printenv'
//*
//STDENV   DD *
ZA=chmod
ZD=/user-defined_pathname
ZP=ppp
ZI=install
ZB=bin
ZL=lib
//STDOUT   DD PATH='/tmp/SVAAinstall.J0PATHSout2',
//            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=SIRWXU
//STDERR   DD PATH='/tmp/SVAAinstall.J0PATHSerr2',
//            PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=SIRWXU
//
```

Figure 2-9. *JCL member J0PATHS: Creating the SVAA Server directories on UNIX Systems Services*

Continue with job J1DDDEFS.

Step 5i: Defining the SVAA Server Path Names

Use the sample JCL in member **J1DDDEFS** to allocate the SVAA Server DLIB data set and define the SVAA Server USS path names.

Change */<user-defined_pathname>* to the OS/390 USS path name where your HFS files will reside.

```
//jobname   JOB    'accounting info'
//*
//STEP1     EXEC   smpeproc
//*
//ASJCHFS   DD     DISP=(,CATLG),
//          DSN=hlq1.ASJCHFS,
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//          UNIT=iuuu,
//          VOL=SER=ivvvvv,
//          SPACE=(3120,(2288,260,20))
//*
//SMPCNTL   DD     *

 SET BOUNDARY ( dlib ) .       /* Change to your DLIB zone   */
 UCLIN .

 REP        DDDEF    ( ASJCHFS )
            DATASET  ( hlq1.ASJCHFS )
            SHR
            .
 ENDUCL .

 SET BOUNDARY ( target ) .     /* Change to your target zone */

 UCLIN .

 REP        DDDEF    ( SJCBIN )
            PATH     ( '/<user-defined_pathname>/bin/' )
            .
 REP        DDDEF    ( SJCINS  )
            PATH     ( '/<user-defined_pathname>/install/' )
            .
 REP        DDDEF    ( SJCLIB )
            PATH     ( '/<user-defined_pathname>/lib/' )
            .
 ENDUCL .
```

Figure 2-10. *JCL member J1DDDEFS: Defining the SVAA Server path names*

Step 5j: Applying the SVAA Server Function

Use the sample JCL in member **J2APPLY** to apply the SVAA Server function. This job must be executed on the operating system where the *user-defined_pathname* subdirectories */bin, /install*, and */lib* local HFS resides.

```
//jobname   JOB    'accounting info'
//*
//STEP1     EXEC   smpeproc
//SJCBIN DD PATH='/<user-defined_pathname>/bin/'
//SJCINS DD PATH='/<user-defined_pathname>/install/'
//SJCLIB DD PATH='/<user-defined_pathname>/lib/'
//SMPCNTL  DD  *

 SET BOUNDARY ( target ) .     /* Change to your target zone */

 APPLY   SELECT (  SSJC110     /* SVAA Server function      */
                )
         GROUPEXTEND
         NOJCLINREPORT
         RETRY ( YES )
         .
```

Figure 2-11. *JCL member J2APPLY: Applying the SVAA Server function*

Step 5k: Accepting the SVAA Server Function

Use the sample JCL in member **J3ACCEPT** to accept the SVAA Server function. This job must be executed on the operating system where the *user-defined_pathname* subdirectories */bin, /install*, and */lib* local HFS resides.

```
//jobname   JOB    'accounting info'
//*
//STEP1     EXEC   smpeproc
//SJCBIN DD PATH='/<user-defined_pathname>/bin/'
//SJCINS DD PATH='/<user-defined_pathname>/install/'
//SJCLIB DD PATH='/<user-defined_pathname>/lib/'
//SMPCNTL  DD  *

 SET BOUNDARY ( dlib ) .       /* Change to your DLIB zone   */

 ACCEPT  SELECT (  SSJC110     /* SVAA Server function       */
               )
        NOJCLINREPORT
        RETRY ( YES )
        .
```

Figure 2-12. *JCL member J3ACCEPT: Accepting the SVAA Server function*

When the Accept job has completed successfully,

- Go to pages 6-3 and 6-4 and execute the P2APPLY and P3ACCEPT jobs. After they have run, return to this spot to determine your next step.
- if you will be running SVAA on more than one level of the OS/390 operating system, proceed to “Step 6: Installing Multi-level Operating System Support.”
- otherwise, go to “Step 7: Installing Usermods” on page 2-29.

Step 6: Installing Multi-level Operating System Support

The JES Offset Table functions (TSIB311 and TSIB312) use JES and DFP macros to create non-executable load modules (SIBRSLV2 and SIBRSLV3) that contain information needed for the proper operation of SVAA. If any of the IBM macros used for this information change (by new release, PTF, or APAR), these tables may not be usable and may have to be reinstalled.

Note: It may not be necessary to install the JES Offset Tables. Refer to "JES Offset Table Elimination" on page 1-6 before running the "M" set of jobs.

If you need to run SVAA on multiple operating systems using different levels of OS/390, DFP, and/or JES2/JES3, *and* you have determined that it is necessary to install TSIB311 or TSIB312 separately for each of those operating systems, follow the instructions for providing multi-level operating system support (MLSS) contained in the following sections.

Installation JCL

For each task in this section, you can use the Installation JCL library member, the SMP/E Sysmod Management dialogs, or a printed sample JCL provided previously.

If you are using the installation JCL, you will need to repeat the 'M' series of jobs (jobs M1 - M6) for each level of the operating system where SVAA will run.

Location of the Target and Distribution Zones

See installation JCL member **M1CSI**. The recommended method is to install the JES functions, TSIB311 or TSIB312 (or both), in a separate target and distribution zone for each level of your operating system. The location of these target and distribution zones is up to you. They can share the same global zone with the SVAA Base function, or they can be installed in the OS/390 target and distribution zones for each system.

Base Function ID Definitions

See installation JCL member **M1CSI**. Because TSIB311 and TSIB312 are dependent functions of SSIB310, it is necessary to create a dummy FMID for SSIB310 on each system. This will cause the Base function to **appear** to be installed in the target and distribution zones where the multi-level system support is to be installed.

System PARMLIB

See installation JCL member **M1CSI**. The PARMLIB DDDEF for SYS1.PARMLIB is required in each SMP/E zone where you install the JES Offset Table functions (TSIB311 and/or TSIB312). Use the **unit and volume parameters** to ensure that the data set for the desired level of the operating system is used. SYS1.PARMLIB member GIMOPCDE may be needed during assembly of offset tables SIBRSLV2 or SIBRSLV3. The member is not modified by SVAA.

SYSLIB Concatenation

See installation JCL members **M1CSI** and **M2ALLOC**. If you have both JES2 and JES3, duplicate the jesmac DDDEF for the second JES macro library in job M1CSI and include both ddnames in each SYSLIB DDDEF concatenation in job M2ALLOC.

You must define an individual DDDEF and a SYSLIB concatenation DDDEF for the following data sets. The SIBMAC DDDEF must point to the same SIBMAC data set used by the SVAA Base function, SSIB310. When defining the DDDEFs for the "SYS1" data sets, use the **unit and volume parameters** to ensure that the data sets for the desired level of the operating system are used.

- mtshlq.SMPMTS
- SYS1.MACLIB
- SYS1.AMODGEN
- SYS1.SHASMAC (for JES2 OS/390 systems)
- SYS1.HASPSRC (for JES2 non-OS/390 systems)
- SYS1.AIATMAC (for JES3 OS/390 systems)
- SYS1.AJES3MAC (for JES3 systems)
- hlq.SIBMAC

Note: Change mtshlq.SMPMTS to the SMPMTS data set name used by the OS/390 operating system. The SMPMTS is required and uses IBM macros that may have been applied but not yet accepted and therefore do not have an associated target library. SVAA does not have its own SMPMTS data set.

Data Set Allocations and DDDEFs

JCL member **M2ALLOC** creates the following SVAA data sets in the target and distribution zones where you are installing TSIB311 and/or TSIB312.

Note: These are different copies of these data sets from those allocated with the Base function.

- hlq.ASIBLOAD
- hlq.ASIBASM
- hlq.SIBASM
- hlq.SIBLINK

You must allocate each data set and define its DDDEF in each target and distribution zone where a JES Offset Table function is to be installed.

Installing the MLSS Functions with SMP/E

This section explains the requirements for the SMP/E Receive, Apply, and Accept processing.

Receiving the Multi-level Support Functions

These functions were received in job I1RCVFUN in Step 5a.

Applying the Multi-level Support Functions

Use JCL member **M3APPLY** to apply the multi-level support functions. Before submitting the job, specify the name of your target zone in the SET BOUNDARY statement and remove all FMID selections except TSIB311 and/or TSIB312 on the APPLY statement.

Accepting the Multi-level Support Functions

Use JCL member **M4ACCEPT** to accept the multi-level support functions. Before submitting the job, specify the name of the distribution zone in the SET BOUNDARY statement and remove all FMID selections except TSIB311 and/or TSIB312 on the ACCEPT statement.

Applying PTFs to the TSIB311 or TSIB312 Dependent Function

You can use JCL member **M5PTFAPP** to apply PTFs to the JES dependent function.

Accepting PTFs to the TSIB311 or TSIB312 Dependent Function

You can use JCL member **M6PTFACC** to accept PTFs to the JES dependent function.

Execution Considerations

If the JES Offset Tables (SIBRSLV2 or SIBRSLV3) are required, each different version must reside in an authorized link list library that is only used by the level of the operating system for which it was built and must be in link list libraries that are not shared by the JES2 and JES3 systems.

Available Usermods

In support of the JES Offset Tables, three optional usermods have been provided to cause a regression report from SMP/E if any of the required IBM macros have been updated. Refer to “Step 7a: Installing the JES Offset Table Support Usermods” on page 2-29 for additional information.

Additional Operating Systems

Repeat steps “Location of the Target and Distribution Zones” on page 2-26 through “Installing the MLSS Functions with SMP/E” on page 2-27 (jobs M1 - M6) for each level of the operating system where SVAA will execute.

After successful installation of the multi-level system support, proceed to “Step 7: Installing Usermods” on page 2-29.

Step 7: Installing Usermods

Step 7a: Installing the JES Offset Table Support Usermods

Note: It may not be necessary to install the JES Offset Table Support Usermods. Refer to “JES Offset Table Elimination” on page 1-6 before running the “M” set of jobs.

A set of documentation and sample JCL are located in the SIBSAMP library to supply three optional SMP/E dummy usermods that support the SVAA JES Offset Tables (SIBRSLV2 and/or SIBRSLV3). Installing these dummy usermods causes SMP/E to generate a regression report for the usermods if any of the required IBM elements are changed by a PTF or APAR. The following members are provided:

SIBOTDOC	Documentation for JES Offset Tables sample usermods.
SIBOTULD	Sample JCL to unload the macro entries from the operating system target zone.
SIBOTUM1	Sample dummy usermod for the required DFP macros.
SIBOTUM2	Sample dummy usermod for the required JES2 macros.
SIBOTUM3	Sample dummy usermod for the required JES3 macros.

Step 7b: Installing the SVAA SIBCTUM Usermod

SVAA provides a copy of the SAS/C 6.5.0 transient run-time library in the *hlq.*SACLINK data set. It is possible that other vendors also include the transient library as part of their product. If you have more than one copy of the SAS/C transient run-time library, you may be unable to place the *hlq.*SACLINK data set in the system link list or to use the standard CTRANS ddname for this data set.

A usermod is required to allow the use of a ddname other than CTRANS to load the SAS/C 6.5.0 transient library routines. This is done by changing the SAS/C CTRANS ddname to SIBCTRAN in all SVAA load modules containing the L$CMAINO csect.

JCL and documentation to install the usermod is located in the SIBSAMP library member SIBCTUM.

The usermod can be implemented without an IPL by following the steps documented in the sample member.

Chapter 3. Customizing for SVAA

Step 8	Customizing SYS1.PARMLIB for SVAA
Step 9	Defining parmlib members for the SVAA started task
Step 10	Defining SIBSTK*xx*, which is used in customizing the TSO and batch environments
Step 11	Verifying library requirements
Step 12	Creating SVAA started-task JCL
Step 13	Creating SVAASRVR started-task JCL
Step 14	Customizing the TSO environment
Step 15	Defining security resources
Step 16	Customizing SVAA User Exits
Step 17	Customizing the logging facility

Step 8: Customizing SYS1.PARMLIB for SVAA

To run SVAA on your system, you must customize SYS1.PARMLIB members by performing some or all of the following steps:

- Customizing LNKLST*xx* or PROG*xx*
- Customizing IEAAPF*xx* or PROG*xx*
- Customizing IECIOS*xx*
- Customizing IEFSSN*xx*
- Customizing SMFPRM*xx*
- Customizing MPFLST*xx*

See the IBM *OS/390 Initialization and Tuning Reference* for the correct syntax for each parmlib member.

SIBSAMP Library

Samples of the SYS1.PARMLIB members are included in the SIBSAMP library installed with SMP/E. This library is a collection of sample SVAA jobs, parameter files, and user exits that help you initialize, customize, and test SVAA. Each member includes comments that describe its purpose and explain any necessary modifications you must make. A complete list of the contents of SIBSAMP is provided in Appendix B, “Contents of SIBSAMP.”

Step 8a: Customizing the LNKLSTxx or PROGxx Member

A copy of the following data set must be in the link list concatenation (either the PROG*xx* member for OS/390 2.4 or above, or the LNKLST*xx* member for any level of OS/390):

hlq.**SIBLINK**

To improve performance, it is recommended that copies of the following data sets also be in the system link list concatenation:

hlq.**SIBLOAD**
hlq.**STKLOAD**

Sample members, named LNKLSTIX and PROGIX, are included in the SIBSAMP library.

Note: Do not place the SMP/E target data sets (SIBLINK and/or SIBLOAD) in the system link list. As with all SMP/E target data sets, this can cause system problems when applying maintenance to the data sets. Instead, place copies of these data sets in the link list. The copies should have zero secondary extents.

Step 8b: Customizing the IEAAPFxx or PROGxx Member

You must include the following data sets in one of the authorized program lists in SYS1.PARMLIB (either the PROG*xx* member for OS/390 2.4 or above, or the IEAAPF*xx* member for any level of OS/390):

hlq.**SACLINK**
hlq.**SIBLINK**
hlq.**SIBLOAD**
hlq.**STKLOAD**

Sample members, named IEAAPFIX and PROGIX, are included in the SIBSAMP library.

Note: If a library is in the link list concatenation but is not APF-authorized, referring to the library in a JOBLIB or STEPLIB statement causes the library to be considered unauthorized for the duration of the job or step. Therefore, if you will ever access any of the above three data sets using a JOBLIB or STEPLIB statement, the data sets must be in an APF list, even if all three are in the link list and the link list is automatically authorized (the LNKAUTH=APFTAB parameter of IEASYS00 is *not* specified).

Step 8c: Customizing the IECIOSxx Member

For support of SVA's internal error-recovery procedures, the proper MIH (missing-interrupt handler) timeout value must be set for all connected devices on each SVA subsystem, regardless of whether SVAA is installed. (For each SVA subsystem there is a minimum MIH timeout value, which you can obtain from your StorageTek CSE.)

You need to set the MIH timeout value in IECIOS*xx* if:

- you do not initialize the SVAA subsystem at IPL time, and
- you do not allow SVAA to set this value automatically.

SVAA can automatically set the proper MIH timeout value if you omit the **MIH** parameter (or specify **MIH(S)**) in the subsystem definition contained in the IEFSSN*xx* member of SYS1.PARMLIB (see “Step 8d: Customizing the IEFSSNxx Member” on page 3-4).

Note: If you have applications on your system that detect missing interrupts independently of the system MIH setting, you ***must*** modify those applications to allow at least the minimum MIH timeout value (obtained from your CSE) for all SVA devices.

To Customize the IECIOSxx Member

If you use the **MIH(W)** parameter in the subsystem definition contained in the IEFSSN*xx*, you need to modify the MIH parameter in the IECIOS*xx* member of SYS1.PARMLIB. This ensures that the required value is in effect for subsequent IPLs.

Sample MIH parameter (for a timeout value of 5 minutes):

```
MIH TIME=05:00,DEV=(0100-01FF)
```

You can determine the current setting of the MIH timeout value by issuing the OS/390 operator command:

```
D IOS,MIH,DEV=(mmmm-nnnn)
```

where **mmmm-nnnn** is the range of device numbers of your SVA devices.

You can also change the timeout value from the system console using the SETIOS command. For more information, see *OS/390 Operations: System Commands*.

If you dynamically add SVA devices between IPLs, you must reset the timeout value for the devices *after* they have been added.

A sample IECIOS*xx* member, named IECIOSIX, is included in the SIBSAMP library.

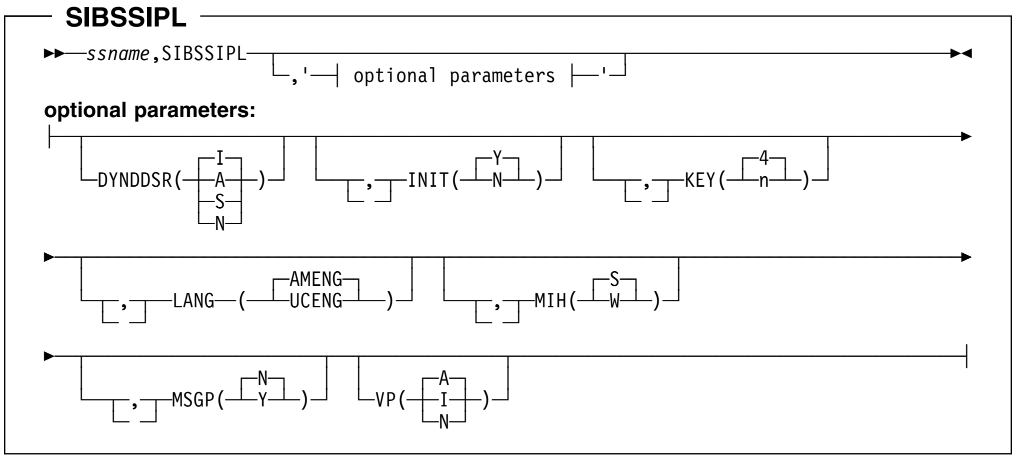
Step 8d: Customizing the IEFSSNxx Member

Each SVAA subsystem must be defined to OS/390 as a secondary subsystem. Subsystem definitions are contained in SYS1.PARMLIB member IEFSSN*xx* (known as the Subsystem Name Table or SSN). Some examples are provided on page 3-7.

A sample IEFSSN*xx* member, named IEFSSNIX, is included in the SIBSAMP library.

Note: SVAA does not support the dynamic subsystem SETSSI commands that were introduced with MVS/ESA. However, SETSSI can still be used to dynamically add an SSN definition, as in “Step 18: Enabling Parmlib Changes” on page 4-2.

Syntax



Notes:

1. The VP (volume preferencing) parameter is specifically for SnapShot. It is ignored if SnapShot in not installed.
2. You can use either a blank or a comma to separate optional parameters.
3. Any sequence of optional parameters containing a blank or comma must be enclosed in single quotation marks.

Parameters

The SVAA entries in the IEFSSN*xx* member can have the following parameters.

ssname

Required

This parameter specifies the name of the SVAA subsystem (*not* the SVA subsystem).

The SSN entry is required in the subsystem name table of each OS/390 operating system under which SVAA will execute. The SVAA subsystem name does not have to be unique across operating systems; the same name (or a different name) can be used on each OS/390 system.

Values:	*ssname* contains up to 4 characters and must begin with an alphabetic or special character (#, @, or $). The remaining characters (if any) can be alphanumeric or any of the same special characters.
Default value:	None

SIBSSIPL

Required

This keyword specifies that SIBSSIPL is the name of the subsystem initialization routine.

Values:	SIBSSIPL (no other value is allowed)
Default value:	None

DYNDDSR

Optional

This parameter controls the initialization of Dynamic DDSR.

If you initialize SVAA during IPL with Dynamic DDSR active (that is, using **DYNDDSR(A)**), Dynamic DDSR begins immediately to process *all* volumes on *all* installed SVA subsystems; you cannot limit DDSR processing to specific volumes or subsystems when you activate Dynamic DDSR at initialization time.

However, if you do not want Dynamic DDSR to access all volumes, specify **DYNDDSR(I)**. Then, after SVAA is initialized, issue a RELEASE DYNAMICDATA subcommand that specifies the volumes you want to be processed.

Values:

A	Dynamic DDSR is to be initialized and activated.
S	Dynamic DDSR is to be initialized but only activated in simulation mode; no actual space release will be performed.
I	Dynamic DDSR is to be initialized but not activated until a RELEASE DYNAMICDATA(STATUS(ACTIVE)) subcommand is issued.
N	Dynamic DDSR is not to be initialized and cannot be activated until the SVAA subsystem is restarted with **A, I,** or **S** specified in DYNDDSR.

Default value: **I**

INIT

Optional

This parameter indicates whether subsystem initialization is to be performed during IPL.

Values:

Y	Perform full subsystem initialization, including allocation and initialization of data structures, loading of global service routines, and device mapping. **Note:** Performing device mapping can lengthen the IPL process. See Appendix D, “Device Mapping” for more information.
N	Do not perform subsystem initialization. (For information about initializing SVAA with SIBMVSS at a later time, see “Step 19: Initializing the SVAA Subsystem with SIBMVSS” on page 4-4.)

Default value: `Y`

KEY

Optional

This parameter identifies the protection key in which the subsystem is to run. The SVAA subsystem can be initialized with a problem program protection key of 1 to 7. Once initialized, the key value for the SVAA subsystem cannot be changed.

Values: 1 to 7

SVAA will not function properly if you specify a problem program protection key of 8 to 15.

Default value: 4

LANG

Optional

This parameter indicates whether SVAA subsystem messages routed to the operator console are to appear in mixed case or all uppercase.

Values:

`AMENG`	Display all messages in "American" (mixed case) English.
`UCENG`	Display all messages in uppercase English.

Default value: `AMENG`

MIH

Optional

This parameter specifies how SVAA is to handle incorrect settings of the DASD MIH (missing-interrupt handler) timeout value. Step 8c (page 3-3) describes how to verify that the MIH timeout value is set correctly for all SVA devices.

Values:

`S`	SVAA is to set the MIH timeout interval to the recommended value for all SVA devices. When setting MIH values for devices, SVAA displays the SIB2972I message once—for the first SVA device only.
`W`	SVAA is *not* to change the MIH timeout interval for any SVA device, but is to display the warning message SIB2971W for each SVA device for which the MIH timeout value is insufficient.

Default value: `S`

MSGP

Optional

This parameter indicates whether the subsystem name is to be prefixed in the display of SVAA messages.

To support automated console operations programs, the subsystem name normally does not prefix the message identifier.

Values:

`Y`	Display the subsystem name as a prefix to SVAA messages.
`N`	Suppress the display of the subsystem name as a prefix to SVAA messages.

Default value: `N`

VP

Optional

This parameter specifies status for the volume preferencing intercept.

Note: The VP parameter is ignored if SnapShot is not installed.

Values:

`A`	SVAA installs volume preferencing and makes it active.
`I`	SVAA installs volume preferencing, but leaves it inactive (volume preferencing can be made active later with a SET VOLPREF subcommand).
`N`	Volume preferencing is not to be installed and cannot be activated with a SET VOLPREF subcommand. To activate volume preferencing later, you must reinitialize the SVAA subsystem specifying **`VP(A)`** or **`VP(I)`**. (In the *SVA SnapShot for OS/390 Installation, Customization, and Maintenance* guide, see the step called “Initializing Volume Preferencing with SIBMVSS.”)

Default value: `A`

Examples of Valid Subsystem Definitions

```
SVAA,SIBSSIPL
SVAA,SIBSSIPL,'INIT(Y) MSGP(Y)'
SVAA,SIBSSIPL,'DYNDDSR(A) MIH(W)'
SVAA,SIBSSIPL,'INIT(Y) LANG(UCENG)'
SVAA,SIBSSIPL,'DYNDDSR(N),INIT(Y)'
```

Step 8e: Customizing the SMFPRMxx Member

If you want SMF to collect SVAA data, you must identify the SMF record type and subtypes for SVAA in the SMFPRM*xx* member of SYS1.PARMLIB.

SVAA SMF records have six different subtypes:

- Record subtypes 1 thru 4 are used for Reporter data. The record formats of these four subtypes are documented in the *SVAA for OS/390 Reporting* manual.
- Record subtype 5 is for data written when space is released by Interval or Dynamic DDSR. The format of subtype 5 records is documented in the *SVAA for OS/390 Configuration and Administration* manual.
- Record subtype 7 is for space utilization data. The format of subtype 7 records is documented in the *SVAA for OS/390 Reporting* manual.

A sample SMFPRM*xx* member, named SMFPRMIX, is included in the SVAA SIBSAMP data set. (This member contains the SUBPARM syntax but, as explained in steps 2 and 3 below, you also may need to include the SYS and SUBSYS parameters.)

Syntax

```
SYS(options)
SUBSYS(STC,options)
SUBPARM(ssname(smftype[,subtypes]))
```

To have SMF collect SVAA data, you must:

1. In the SUBPARM parameter, select the record type to be used for SVAA data and specify the record subtypes for the data you want to collect.
2. Include the selected SVAA record type in the TYPE option of the SYS parameter—if the TYPE option is present. (If the TYPE option is omitted, ***all*** record types are enabled by default.)
3. If the SUBSYS(STC,...) parameter (for started tasks) is present in SMFPRM*xx*, and if it includes the TYPE option, it must include the SVAA record type.

SYS Parameter

The options for the SYS parameter are described in the IBM *OS/390 Initialization and Tuning Reference*. The TYPE and NOTYPE options both affect SVAA data collection for SMF. For SVAA data to be collected:

- If the TYPE option is specified, the SVAA SMF record type (selected in the SUBPARM parameter) must be included as one of its values.
- If the NOTYPE option is specified, the SVAA record type must *not* be included as one of its values.

SUBSYS Parameter

The SUBSYS(STC,...) parameter is optional in the SMFPRM*xx* member. The options available with SUBSYS(STC,...) are identical to those used with the SYS parameter.

However, if the SUBSYS(STC,...) parameter is present in SMFPRM*xx*, the values entered for the options in SUBSYS(STC,...) override the values specified for started-task options in the SYS parameter.

SUBPARM Parameter

The SUBPARM parameter enables you to specify the SVAA SMF record type and subtypes to be written.

If the TYPE or NOTYPE option is specified in the SYS and SUBSYS(STC,...) parameters, the SVAA record type (*smftype*) specified in the SUBPARM parameter must match the corresponding record type specified in the SYS and SUBSYS(STC,...) parameters.

ssname

Required

ssname specifies the SVAA subsystem name.

Values:	The name must be the *ssname* defined in the IEFSSN*xx* member (see page 3-4).
Default value	None

smftype

Required

smftype specifies the SMF record type to be used for SVAA data.

Values:	An integer ranging from 128 to 255
Default value:	None

subtype

Optional

subtype specifies the SVAA record subtypes that are to be written (or *not* written) for this SVAA subsystem.

Values: One or more SVAA record subtypes, with or without minus signs:

1, 2, 3, 4, SRP	Write Reporter SMF records. (Entering any one, or a combination, of the values causes all four Reporter subtypes to be written.)
-1, -2, -3, -4, NOSRP	Do not write Reporter SMF records. (Entering any one, or a combination, of the values causes all four Reporter subtypes to be written.)
5, DDSR	Write DDSR SMF records.
-5, NODDSR	Do not write DDSR SMF records.
7	Write space utilization SMF records.
-7	Do not write space utilization SMF records.

Default values: -1, -5, -7

Example of SMFPRMxx Member

In the following SMF entry example, 250 is the record type selected for SVAA data. Reporter, DDSR, and space utilization records are to be written. (All four Reporter record subtypes are indicated by including the **2**.)

```
SYS(TYPE(0:104,250))
SUBSYS(STC,TYPE(0:55,80:104,250))
SUBPARM(SVAA(250,2,5,7))
```

Figure 3-1. *SMFPRMxx parmlib member example*

Step 8f: Customizing the MPFLSTxx Member

SVAA messages that appear on the operator console in response to SVAA operator commands do not contain Descriptor Code 5 (Immediate Command Response). Some automated operations software packages require Descriptor Code 5 on a message to process the message as a command response.

SIBMPF is an MPF exit that resides in the SIBLINK library. The module's purpose is to turn on Descriptor Code 5 for messages for which it is invoked. The MPF facility provided by OS/390 allows the user to control which messages are processed by an MPF exit. By properly coding the MPFLST*xx* member of SYS1.PARMLIB, any SVAA console message can appear with Descriptor Code 5.

Example of MPFLSTxx Member

A sample member illustrating the correct use of SIBMPF is included in the SIBSAMP library in member MPFLSTIX. For more information about the MPFLST*xx* member of SYS1.PARMLIB, see the IBM *OS/390 Initialization and Tuning Reference*.

Example:

```
SIB0654I,AUTO(YES),SUP(NO),USEREXIT(SIBMPF)
SIB0655I,AUTO(YES),SUP(NO),USEREXIT(SIBMPF)
SIB0656I,AUTO(YES),SUP(NO),USEREXIT(SIBMPF)
```

Once the MPFLST*xx* member has been coded, either issue the SET MPF operator command, or IPL your system to activate the definition. For more information about the SET MPF command, see the *OS/390 System Commands* manual.

Step 9: Defining Parmlib Members for the SVAA Started Task

Parmlib statements enable you to specify various parameters related to SVAA operation. This step involves defining one or more of the following parmlib members for the SVAA started task:

- SIBSYS*xx* — has pointers to the following five members
- SIBDSR*xx* — controls Deleted Data Space Release (DDSR)
- SIBSFC*xx* — manages the SVAA Server
- SIBSIM*xx* — controls processing of SIM messages that are unique to SVA
- SIBSRP*xx* — controls Subsystem Reporting Program (SRP) initialization
- SIBVOP*xx* — enables the entry of operator commands that control and display the status of channels and devices

Parameter members for the SVAA started task, and for the applications that run under it, may be placed either in the system parameter library, SYS1.PARMLIB, or in a data set identified by an STKPARMS DD statement in the started-task JCL.

Note: Sample parmlib statements in this manual illustrate syntax. Refer to the *SVAA for OS/390 Configuration and Administration* and *SVAA for OS/390 Reporting* manuals for detailed information about subcommands.

Step 9a: SIBSYSxx Member

The SIBSYS*xx* parmlib member enables you to specify which SVAA components are to run in the SVAA address space. Keyword values specified in this member are used to locate other parmlib members that contain control information for SVAA components.

The syntax rules are:

- There can be only one entry per line
- There must be a comma after each entry except the last
- Each comment line begins with `/*` and ends with `*/`

```
/*  Sample parmlib member SIBSYSXX  */
DSR=00,
SFC=00,
SIM=00,
SRP=00,
VOP=00
```

Figure 3-2. *Sample SIBSYSxx parmlib member*

The example in Figure 3-2 is the SIBSYSXX member in the SIBSAMP data set. The statements in the example are interpreted as follows:

DSR=xx

Required—if you want to start the SIBDSR*xx* subtask

Identifies the SIBDSR*xx* parmlib member that is to be used for DDSR startup parameters (see “Step 9b: SIBDSRxx Member” on page 3-13).

Values: *xx*—two alphanumeric characters specifying: start the DDSR subtask using the startup parameters in SIBDSR*xx*

(blank)—start the DSR subtask but do not pass the SIBDSR*xx* parmlib member

SFC=xx

Required—if you want to start the SIBSFC*xx* subtask

Identifies the SIBSFC*xx* parmlib member that is to be used for SVAA Server startup parameters (see "Step 9c: SIBSFCxx Member" on page 3-14).

Values: *xx*—two alphanumeric characters specifying: start the SFC subtask using the startup parameters in SIBSFC*xx*

(blank)—start the SFC subtask but do not pass the SIBSFC*xx* parmlib member

SIM=xx

Required—if you want to start the SIBSIM*xx* subtask

Identifies the SIBSIM*xx* parmlib member that is to be used for SIM (service information message) startup parameters (see "Step 9d: SIBSIMxx Member" on page 3-14).

Values: *xx*—two alphanumeric characters specifying: start the SIM subtask using the startup parameters in SIBSIM*xx*

(blank)—start SIM processing but do not pass the SIBSIM*xx* parmlib member

SRP=xx

Required—if you want to start the SIBSRP*xx* subtask

Identifies the SIBSRP*xx* parmlib member that is to be used for Reporter startup parameters (see "Step 9e: SIBSRPxx Member" on page 3-15).

Values: *xx*—two alphanumeric characters specifying: start the SRP subtask using the startup parameters in SIBSRP*xx*

(blank)—start data collection but do not pass the SIBSRP*xx* parmlib member

VOP=xx

Required—if you want to start the SIBVOP*xx* subtask

Identifies the SIBVOP*xx* parmlib member that is to be used for SVA virtual operator panel (VOP) startup parameters (see "Step 9f: SIBVOPxx Member" on page 3-16).

Values: *xx*—two alphanumeric characters specifying: start the VOP subtask using the startup parameters in SIBVOP*xx*

(blank)—start the VOP subtask but do not pass the SIBVOP*xx* parmlib member

Example

```
DSR=,
SIM=00,
SRP=01
```

In this example, (1) the DDSR subtask is started without any user-specified parameters, (2) the SIM subtask is started using member SIBSIM00, (3) the Reporter subtask is started using the startup parameters in member SIBSRP01, and (4) the VOP subtask is not started.

Step 9b: SIBDSRxx Member

The SIBDSR*xx* parmlib member provides a means of activating and controlling Interval or Dynamic DDSR.

The example in Figure 3-3 is from the SIBDSRXX member of the SIBSAMP library. The **RELEASE INTERVALDATA** subcommand in the example defines two Interval DDSR tasks for SVA subsystem ICEBERG1: one to run at 4-hour intervals, the other at 24-hour intervals.

```
/*  Perform Interval DDSR for subsystem ICEBERG1  */
RELEASE INTERVALDATA(-
     RELID(ICE1HOT)-
     INT(4H)-
     RESLIM(5S)-
     DEFER(YES)-
     SUBSYS(ICEBERG1)-
     VOL(ICE001:ICE029))
RELEASE INTERVALDATA(-
     RELID(ICE1COOL)-
     INT(24H)-
     RESLIM(5S)-
     DEFER(YES)-
     SUBSYS(ICEBERG1)-
     VOL(ICE030:ICE070))
```

Figure 3-3. *Sample SIBDSRxx parmlib member (Interval DDSR)*

The example in Figure 3-4 is from the SIBDSR06 member of the SIBSAMP library. The **RELEASE DYNAMICDATA** subcommand in the example requests that SVAA activate Dynamic DDSR for a specific set of functional volumes.

```
/* Activate Dynamic DDSR for subsystem 'ICE001' */
RELEASE DYNAMICDATA (STAT(ACT) SUBSYS(ICE001) VOL(PROJ4:PROJ7))
```

Figure 3-4. *Sample SIBDSRxx parmlib member (Dynamic DDSR)*

Note: The SIBDSR*xx* members above are only examples; you can enter any DDSR, SET, or QUERY subcommand in SIBDSR*xx*. For detailed information about these subcommands, see the *SVAA for OS/390 Configuration and Administration* and *SVAA for OS/390 Reporting* manuals.

For more examples of SIBDSR*xx*, see SIBDSR01 through SIBDSR05 in the SIBSAMP data set.

Step 9c: SIBSFCxx Member

The SIBSFC*xx* parmlib member defines all of the variable information required to initialize the SVAA Server. The SET subcommands shown in the sample SIBSFC*xx* (below) are described in the *SVAA for OS/390 Configuration and Administration* manual.

The example in Figure 3-5 is from the SIBSFCXX member of the SIBSAMP library.

```
/*  Establish the SVAA Server environment variables      */
SET CASE(mixed)                      /* Maintain case              */
SET DEBUG(OFF)                       /* Turn Debugging off         */
SET ECAMDEV(100 200 300)             /* Subsystems available to SVAC */
SET CONFIGPATH(/svaa1.1/install)     /* Location of config.dat file  */
SET JAVA_HOME(/usr/lpp/java/J1.1)    /* Java install directory       */
SET LOGFILE(/svaa1.1/SVAA-ServerLog) /* Location and name of the     */
                                     /*  SVAA Server Log file        */
SET SERVERPROC(SVAASRVR)             /* Name of the SVAA Server      */
                                     /*  Proc to be executed when    */
                                     /*  SFC is initialized          */
SET SERVERNAME(SVAA-Server)          /* Unique name for SVAA Server  */
SET SIBHOME(/svaa1.1)                /* SVAA install directory       */
SET SIBPORT(22753)                   /* Port SVAA Server will use    */
                                     /*  to communicate with clients */
```

Figure 3-5. *Sample SIBSFCxx parmlib member*

Note: If you plan to produce point-in-time reports, you will want to add a SET WEBPORT subcommand to this parmlib member.

Step 9d: SIBSIMxx Member

The SIBSIM*xx* parmlib member enables you to control SIM processing—the process of translating SVA-specific SIM messages into messages more meaningful to users.

The example in Figure 3-6 is the SIBSIMXX member in the SIBSAMP data set. The **`SET ECAMDEVICE`** subcommand identifies the devices over which ECAM I/O operations are to be performed. (See Appendix A, "SVAA Communication Devices" for more information on ECAM devices.)

```
/*  Service Information Message Processing  */
SET ECAMDEVICE(200)
```

Figure 3-6. *Sample SIBSIMxx parmlib member*

The SIBSIM*xx* parmlib member must contain at least one SET ECAMDEVICE subcommand for each installed SVA subsystem.

Step 9e: SIBSRPxx Member

The SIBSRP*xx* parmlib member allows you to control initialization of Reporter data collection.

In the example in Figure 3-7, data collection occurs with the same synchronization time and at the same intervals as specified for RMF data collection in member ERBRMF00 in SYS1.PARMLIB or the data set allocated to the STKPARMS DD statement.

```
/*  SRP Performance Tracking Data Collection  */
SET CASE(U)
SET CMDE(ON)
SET ECAMDEVICE(310)
INITIALIZE MAINLOG(SMF)
/*  Collect Performance Tracking Data at 15-minute Intervals  */
COLLECT PTDATA(-
     RMF(00))
```

Figure 3-7. *Sample SIBSRPxx parmlib member*

Note: In most cases, the SIBSRP*xx* member must at least contain a SET ECAMDEVICE subcommand.

The example in Figure 3-8 is from the SIBSRPXX member of the SIBSAMP data set. In this example, data is collected for subsystem ICEBERG1 only.

```
/*  SRP Performance Tracking Data Collection  */
SET CASE(U)
SET CMDE(ON)
SET ECAMDEVICE(310)
INITIALIZE MAINLOG(SMF)
/*  Collect Performance Tracking Data on 15-minute Intervals  */
COLLECT PTDATA(-
     TIMES(0000:2400)-
     SYNC(5M)-
     SUBSYS(ICEBERG1)-
     INTERVAL(15M))
```

Figure 3-8. *Sample SIBSRPxx parmlib member*

In addition to the subcommands to control data collection, the examples in Figure 3-7 and Figure 3-8 include other subcommands. `SET CASE(U)` causes all input characters to be translated automatically to uppercase. `SET CMDE(ON)` specifies that messages are to be displayed at the destination specified by the SET DESTINATION subcommand.

These are only examples; you can enter any Reporter subcommands and any SET and QUERY subcommands in the SIBSRP*xx* parmlib member. See the the *SVAA for OS/390 Reporting* manual for descriptions of Reporter subcommands and the SVAA SET and QUERY subcommands.

Step 9f: SIBVOPxx Member

The SIBVOP*xx* parmlib member enables you to control whether SVAA virtual operator panel (VOP) commands can be issued from the system console.

The example in Figure 3-9 is the SIBVOPXX member in the SIBSAMP data set. The **SET ECAMDEVICE** subcommand identifies the devices over which ECAM I/O operations are to be performed. (See Appendix A, “SVAA Communication Devices” for more information on ECAM devices.)

```
/*  Console Command Processing  */
SET ECAMDEVICE(200)
```

Figure 3-9. *Sample SIBVOPxx parmlib member*

Figure 3-9 is only an example of a SIBVOP*xx* member; you can include any of the SIBVOP subcommands described in the *SVAA for OS/390 Configuration and Administration* manual or the *SVAA for OS/390 Reporting* manual.

Step 10: Defining SIBSTKxx Member

The SIBSTK*xx* parmlib member enables you to issue SVAA subcommands to set up your TSO or batch environment. The default SIBSTK*xx* parmlib member name is SIBSTK00. The SIBSTK*xx* member can be allocated in TSO or SIBBATCH to the STKPARMS file.

A sample SIBSTK*xx* parmlib member is shown in Figure 3-10. This example is in the SIBSTKXX member provided in the SIBSAMP data set.

```
/*  Configuration Parameters for SVAA  */
SET CASE(U)
SET CMDE(ON)
SET ECAMDEVICE(200)
SET DEST(OUTF(SVAA.RPT.LISTING))
```

Figure 3-10. *Sample SIBSTKxx parmlib member*

The SIBSTK*xx* member shown in Figure 3-10 is only an example; you may enter any valid SVAA subcommand in SIBSTK*xx*. In this example:

- **`SET CASE(U)`** causes all input characters to be translated to uppercase.
- **`SET CMDE(ON)`** specifies that messages are to be displayed at the destination specified by the OUTMSG parameter of the SET DEST subcommand.
- **`SET ECAMDEVICE(200)`** designates device **`200`** as an ECAM communications device.
- The **`SET DEST(OUTF ...)`** subcommand specifies the file to which report output is to be directed.

All SVAA subcommands are documented in the *SVAA for OS/390 Configuration and Administration* and *SVAA for OS/390 Reporting* manuals.

Note: In most cases, the SIBSTK*xx* member must at least contain a SET ECAMDEVICE subcommand.

Specifying an Alternate SIBSTKxx Parmlib Member

To specify an alternate SIBSTK*xx* parmlib member for SIBADMIN, use this command:

```
SIBADMIN =PARMLIB=membername
```

For SIBBATCH, use the following statement in the JCL:

```
// <jobcard>
//STEP1  EXEC   PGM=SIBBATCH,PARM='=PARMLIB=membername'
```

where *membername* can have up to 8 characters.

In ISPF sessions, you can specify an alternate SIBSTK*xx* parmlib member for initialization by using the SVAA SESSION PROFILE panel (described in the *SVAA for OS/390 Configuration and Administration* manual and the *SVAA for OS/390 Reporting* manual). Once you have made the changes, exit from and re-enter SVAA to put the changes into effect.

Step 11: Verifying Library Requirements

This section discusses special considerations and requirements for the SVAA load module libraries:

- SIBLINK
- SIBLOAD
- STKLOAD
- SACLINK

For information about allocation parameters, see the "Allocating Data Sets" section in Chapter 2.

Step 11a: Verifying SIBLINK Library Requirements

The *hlq*.SIBLINK load module library:

- Must be in one of the authorized program lists in SYS1.PARMLIB (IEAAPF*xx* or PROG*xx*)
- Must be included in the system link list (LNKLST*xx* or PROG*xx*)
- Should be allocated with no secondary extents
- Should not be used as the SMP/E target library (*hlq*.SIBLINK should be a copy of the target)

Step 11b: Verifying SIBLOAD Library Requirements

The *hlq*.SIBLOAD load module library must be included in one of the authorized program lists in SYS1.PARMLIB (IEAAPF*xx* or PROG*xx*).

If *hlq*.SIBLOAD is included in the system link list (LNKLST*xx* or PROG*xx*), the *hlq*.SIBLOAD load module library:

- Should be allocated with no secondary extents
- Should not be used as the SMP/E target library (*hlq*.SIBLOAD should be a copy of the target)

If *hlq*.SIBLOAD is not included in the system link list, the *hlq*.SIBLOAD load module library:

- Must exist in a STEPLIB DD in:
  - the started-task JCL
  - SIBMVSS jobs
  - SIBBATCH jobs
  - the TSO logon proc
- Must, for ISPF access, be allocated (before ISPF is invoked) to either:
  - the STEPLIB ddname in a TSO logon procedure, or
  - the ISPLLIB ddname in a TSO logon exec, provided that the ISPLLIB ddname has not already been used by the TSO logon procedure.

  (You will do this in "Step 14b: Allocating SVAA Data Sets for ISPF").

Step 11c: Verifying STKLOAD Library Requirements

The *hlq*.STKLOAD load module library must be included in one of the authorized program lists in SYS1.PARMLIB (IEAAPF*xx* or PROG*xx*).

If *hlq*.STKLOAD is included in the system link list (LNKLST*xx* or PROG*xx*), the *hlq*.STKLOAD load module library:

- Should be allocated with no secondary extents
- Should not be used as the SMP/E target library (*hlq*.STKLOAD should be a copy of the target)

If *hlq*.STKLOAD is not included in the system link list, the *hlq*.STKLOAD load module library:

- Must exist in a STEPLIB DD in:
  - the started-task JCL
  - SIBMVSS jobs
  - SIBBATCH jobs
  - the TSO logon proc
- Must, for ISPF access, be allocated (before ISPF is invoked) to either:
  - the STEPLIB ddname in a TSO logon procedure, or
  - the ISPLLIB ddname in a TSO logon exec, provided that the ISPLLIB ddname has not already been used by the TSO logon procedure.

  (You will do this in "Step 14b: Allocating SVAA Data Sets for ISPF").

Step 11d: Verifying SACLINK Library Requirements

SVAA provides a copy of the SAS/C transient run-time library in the *hlq*.SACLINK data set. However, if the system link list already contains a copy of the SAS/C run-time library from IBM or another vendor, you cannot place the *hlq*.SACLINK data set in the link list.

Follow these rules regarding the *hlq*.SACLINK data set:

- You must include the *hlq*.SACLINK load module library in one of the authorized program lists in SYS1.PARMLIB (IEAAPF*xx* or PROG*xx*)—as in Step 8b.
- You must use the SIBCTRAN ddname for all allocations for the *hlq*.SACLINK data set.

  To use the SIBCTRAN ddname you must install the SIBCTUM usermod (see the sample in SIBSAMP). This SAS/C-supplied usermod allows you to use a ddname other than CTRANS to load transient library routines.
- Ensure that the *hlq*.SACLINK data set using the SIBCTRAN ddname is allocated for the following tasks:
  - SVAA started-task JCL (refer to “Step 12: Creating the SVAA Started-Task JCL” on page 3-21)
  - TSO logon procs, EXECs or CLISTs (refer to “Step 14: Customizing the TSO Environment” on page 3-26)
  - TSO batch jobs which execute SIBADMIN (refer to “Step 14: Customizing the TSO Environment” on page 3-26)
  - SIBBATCH JCL (refer to “Step 23: Verifying SVAA under SIBBATCH” on page 4-10)
  - ISPF (refer to “Step 14b: Allocating SVAA Data Sets for ISPF” on page 3-27)
- *hlq*.SACLINK must exist in a STEPLIB DD in SIBMVSS jobs.

Step 12: Creating the SVAA Started-Task JCL

A system-cataloged procedure library (PROCLIB) must contain a procedure for the SVAA started task. The SVAA task must be associated with a userid that has read/write access to the */user-defined_pathname* directories defined in installation JCL member J0PATHS.

Note: This same userid must contain an OMVS segment to allow OMVS access.

The following sample JCL contains a cataloged procedure to start the SVAA started task. This JCL is included in the SVAAJCL member of the SIBSAMP data set.

```
//ssname  PROC MEMBER=xx
//*
//*  Sample JCL to start the SVAA started task.
//*
//ssname  EXEC PGM=SIBMAIN,
//             PARM='&MEMBER',TIME=1440,REGION=4M
//*
//*  SVAA Load Library                      If not in link list
//STEPLIB  DD  DISP=SHR,DSN=hlq.SIBLOAD
//         DD  DISP=SHR,DSN=hlq.STKLOAD
//*
//*  SVAA Parameter Libraries               If not in SYS1.PARMLIB
//STKPARMS DD  DISP=SHR,DSN=hlq.sibparms
//         DD  DISP=SHR,DSN=hlq.rmfparms
//*
//*  Standard SYSOUT
//SYSPRINT DD  SYSOUT=A
//SYSTERM  DD  SYSOUT=A
//*
//*  C Transient Runtime Libraries
//*
//*  If the hlq.SACLINK data set is not in the link list, use the
//*  SIBCTRAN or CTRANS ddname.  To use SIBCTRAN, see member
//*  SIBCTUM in the SIBSAMP library.
//*
//*SIBCTRAN DD  DISP=SHR,DSN=hlq.SACLINK
//*CTRANS   DD  DISP=SHR,DSN=hlq.SACLINK
//*
//*  SRP and DDSR Message Log
//SIBLOG   DD  SYSOUT=A
//*
//outfile1 DD  DISP=SHR,DSN=outfile-name-1
//outfile2 DD  DISP=SHR,DSN=outfile-name-2
//*
//*  Dump data set - optional
//SYSMDUMP DD  DISP=SHR,DSN=hlq.dumpds
//*
```

Figure 3-11. *Sample SVAA started-task JCL*

The sample started-task JCL in Figure 3-11 includes some statements that require modification:

hlq

Replace *hlq* with your high-level qualifier; follow TSO data-set naming conventions.

ssname

Required

Replace *ssname* with the name of the subsystem in which SVAA is running. The name must be the same as the *ssname* defined in “Customizing the IEFSSNxx Member” on page 3-4. This must also be the member name in SYS1.PROCLIB for this started task.

MEMBER=xx

Required

Replace *xx* with the suffix of the SIBSYS*xx* member of the SYS1.PARMLIB data set (or the data set allocated to the STKPARMS DD statement) that is to be used during SVAA started-task initialization. It must be two alphanumeric characters. For example, if member SIBSYS01 is to be used, specify **MEMBER=01**.

STEPLIB

Conditional—required unless both SIBLOAD and STKLOAD are included in the system link list.

If an entry for the *hlq*.SIBLOAD data set is not in the link list, it must be named in the STEPLIB. Similarly, if an entry for the *hlq*.STKLOAD data set is not in the link list, it must be named in the STEPLIB. Refer to “Step 11: Verifying Library Requirements” for more information about these data sets.

STKPARMS

Conditional—required unless SVAA parmlib members are placed in SYS1.PARMLIB.

The first DD statement in the sample specifies the SVAA parameter library. The ddname must be STKPARMS. The data set name may be any name you want (following TSO naming conventions), and the data sets should be allocated with the same DCB attributes as SYS1.PARMLIB.

The second (concatenated) DD statement in the sample enables you to specify a data set in which you can place an ERBRMF*nn* member to contain RMF synchronization information. This enables the SVAA started task to access the RMF information without having access to SYS1.PARMLIB.

Note: The information in the *hlq.rmfparms* data set can be included in the *hlq.sibparms* data set.

SIBCTRAN or CTRANS

Conditional

If an entry for the *hlq*.SACLINK data set is not in the link list, use the SIBCTRAN ddname. You must install the SIBCTUM usermod to use the SIBCTRAN ddname (see sample in SIBSAMP). This SAS-supplied usermod allows the use of a ddname other than CTRANS to load transient library routines. You can also use the CTRANS ddname or a STEPLIB ddname.

SIBLOG

Optional

Specifies the log file to be used for messages issued by applications executing under the SVAA address space. If this statement is not included, application messages are written to the OS/390 system log.

You can use any data set name for the log file (following standard TSO data set naming conventions). Do not specify DCB parameters for the file; they are set up automatically.

outfile1, outfile2

Optional

Specify the SRP data-collection output files that are specified on the OUTFILE or ALTERNATEOUTFILE parameters of the SVAA INITIALIZE MAINLOG or COLLECT ORDATA subcommands (or with the SET DESTINATION subcommand). Follow TSO naming conventions when specifying the ddname and data set name. Refer to the *SVAA for OS/390 Reporting* manual for more information.

SYSMDUMP

Optional

Specifies where the SVAA started task is to write any dumps that it may take. Allocate this data set using the same DCB parameters as in SYS1.DUMP01.

Execution

Use the following commands to start, stop, or modify the SVAA started task (the SVAA address space).

Starting the SVAA Started Task

Start the SVAA address space at an OS/390 console by entering:

```
S ssname[,MEMBER=nn]
```

Stopping the SVAA Started Task

Terminate SVAA by issuing the OS/390 STOP (**P**) command from the system console:

```
P ssname
```

Terminate SVAA with the STOP (**P**) command, if possible, rather than with a CANCEL or FORCE command, to avoid problems with the SVAA started task.

Modifying the SVAA Started Task

Issue a subcommand to the SVAA started task by entering:

```
F ssname,subcommand
```

where *subcommand* is any of the following:

General commands:

```
SEND DSR
SEND SFC
SEND SIM
SEND SRP
SEND VOP
START DSR
START SFC
START SIM
START SRP
START VOP
STOP DSR
STOP SFC
STOP SIM
STOP SRP
STOP VOP
```

DDSR subcommands:

```
DISPLAY RELEASE
RELEASE DYNAMICDATA
RELEASE INTERVALDATA
RESUME RELEASE
START RELEASE
STOP RELEASE
SUSPEND RELEASE
```

VOP subcommands:

```
ATTN DEVICE
DISPLAY CHANNEL
DISPLAY DEVICE
DISPLAY NCL
VARY CHANNEL
VARY DEVICE
```

SVAA Subsystem Reporting subcommands:

```
COLLECT ORDATA
COLLECT PTDATA
DISPLAY COLLECTION
SET COLLECTION
START COLLECTION
```

All of these subcommands are documented in either the *SVAA for OS/390 Configuration and Administration* manual or the *SVAA for OS/390 Reporting* manual.

Step 13: Creating the SVAASRVR Started-Task JCL

A system-cataloged procedure library (PROCLIB) must contain a procedure for the SVAASRVR started task. The SVAASRVR task must be associated with a userid that has read/write access to the */user-defined_pathname* directories defined in installation JCL member J0PATHS.

Note: Additionally, this same userid must contain an OMVS segment to allow OMVS access.

This proc is executed as a result of the SFC initialization. The following sample JCL will initialize the SVAA Server. A copy of this JCL is included in the SVAASRVR member of the SIBSAMP data set.

```
//SVAASRVR PROC
//*
//*  Sample JCL to initialize the SVAA Server
//*
//*  Execute BPXBATCH to remove the STDOUT/STDIN/STDERR
//*  files created by SFC initialization.
//*  Change SSN1 to the SSN of the initializing subsystem.
//*
//S1 EXEC PGM=BPXBATCH,
//        PARM='SH rm /tmp/SIBstd*.SSN1'
//*
//*  Initialize the SVAA Server
//*
//S2 EXEC PGM=SIBSFCCM,REGION=OK
//*
```

Figure 3-12. *Sample SVAASRVR started-task JCL*

The first step (S1) in this proc erases the STDIN, STDOUT and STDERR data sets created as a result of SFC initialization.

The second step (S2) executes the SIBSFCCM program, which initializes the SVAA Server and provides communication between the SVAA Server and the SVAA address space.

Step 14: Customizing the TSO Environment

Step 14a: Allocating SVAA Data Sets for TSO

Before you can execute SVAA from TSO, you must allocate the required data sets to your TSO session.

Load Module Libraries

The data sets required for SVAA to run under TSO are the SVAA load module libraries *hlq*.SIBLINK, *hlq*.SIBLOAD, *hlq*.STKLOAD, and *hlq*.SACLINK. Requirements for the load module libraries are in “Step 11: Verifying Library Requirements.”

SVAA Parameter Library

The SVAA parameter library, STKPARMS, contains startup parameters used by SVAA. Refer to “Step 10: Defining SIBSTKxx Member” on page 3-17 for more information about this data set. Allocate a partitioned data set with a record length of 80 to the STKPARMS ddname.

DDNAME	Data Set Name
STKPARMS	*pvt*.STKPARMS

Command Library

Include a private CLIST or REXX EXEC data set in the standard TSO concatenation for either SYSPROC or SYSEXEC:

This library is used for the PROFSIBA CLIST or REXX EXEC and for SIBADMIN MACRO commands (both optional).

DDNAME	Data Set Name
SYSPROC	*pvt*.CLIST

Sample Logon Proc

A sample logon PROC called SIBTSOP is included in the SIBSAMP library.

Step 14b: Allocating SVAA Data Sets for ISPF

You can allocate the SVAA data sets in any of these ways:

- In a TSO logon PROC (see Figure 3-14 on page 3-29 for an example)
- In a TSO logon exec (see Figure 3-15 on page 3-30 for an example)
- With a CLIST or REXX exec (see Figure 3-16 on page 3-31 and Figure 3-17 on page 3-32 for examples)

You may need to concatenate the SVAA data sets with other data sets having the same ddname.

Load Module Libraries

Requirements for the load module libraries are in "Step 11: Verifying Library Requirements." If the SVAA load module libraries are not included in the system link list or STEPLIB DD statement in your TSO logon PROC, they must be allocated to the standard ISPF concatenations:

DDNAME	Data Set Name
ISPLLIB	*hlq*.SIBLOAD
	hlq.STKLOAD
CTRANS or SIBCTRAN	*hlq*.SACLINK

CTRANS is the default ddname, but if the job in SIBSAMP member SIBCTUM has been run, you must use SIBCTRAN.

Note: You *cannot* access the SVAA load library as a user load library via LIBDEF.

Command Library

Include the SVAA REXX exec data set in the standard TSO concatenation for either SYSPROC or SYSEXEC:

DDNAME	Data Set Name
SYSPROC	*hlq*.SIBCLIB

ISPF Data Sets

Include the following SVAA data sets in the standard ISPF concatenations, or allocate them separately to the SVAA-defined ddnames.

ISPF DDNAME	SVAA DDNAME	Data Set Name	
ISPPLIB	SIBPLIB	*hlq*.SIBPLIB	
ISPMLIB	SIBMLIB	*hlq*.SIBMLIB	
ISPSLIB	SIBSLIB	*hlq*.SIBSLIB	
ISPTLIB	SIBTLIB	*pvt*.SIBTABL	(See Note 1 below)
		hlq.SIBTLIB	
-------	SIBTABL	*pvt*.SIBTABL	(See Note 2 below)

Notes:

1. **The data set SIBTABL *must* be in front of the SVAA SIBTLIB data set in the *xxx*TLIB concatenation** (where *xxx* is either ISP or SIB).

One way to accomplish this is to include these statements in your TSO logon clist:

```
ALLOC FI(xxxTLIB )  DA( 'pvt.SIBTABL'                   +
                        'hlq.SIBTLIB'                   +
                        'SYS1.ISRTLIB'                  +
                        'SYS1.ISPTLIB'      )  SHR REUSE
ALLOC FI(SIBTABL )  DA( 'pvt.SIBTABL'       )  SHR REUSE
```

Figure 3-13. *Position of SIBTABL and SIBTLIB data set statements*

2. SIBTABL is a private data set that is not to be shared with other users. Notice that it is to be allocated under two ddnames, *xxx*TLIB and SIBTABL.

   The SIBTABL DD statement must refer to a single data set (concatenations are not allowed for table output libraries).

   Use the same attributes for the *pvt*.SIBTABL data set that you use for the *hlq*.SIBTLIB data set (see Table 2-5 on page 2-10).

SVAA Parameter Library

Use of the SVAA parameter library, STKPARMS, is optional. It contains startup parameters used by SVAA under both ISPF and native TSO. Refer to “Step 10: Defining SIBSTKxx Member” on page 3-17 for more information about this data set. Allocate a partitioned data set with record length 80 to the STKPARMS ddname.

DDNAME	Data Set Name
STKPARMS	*pvt.sibparms*

ISP vs SIB DDnames

It is possible to allocate SVAA data sets using the ISP*xxxx* ddnames or the SIB*xxxx* ddnames before invoking the SVAA dialog. If the SVAA ddnames (SIB*xxxx*) are used for the allocations, they will precede the ISPF allocations in the search order. If the SIB*xxxx* allocation is not found, the SVAA dialog components are expected to be in the standard ISPF (ISP*xxxx*) concatenations.

Examples

The sample execs on the following pages show four ways of allocating the SVAA data sets for ISPF.

Sample TSO Logon Proc

The sample TSO logon proc in Figure 3-14 allocates the SVAA TSO and ISPF libraries. A copy of this exec, called SIBTSOP, is included in SIBSAMP.

```
//SIBTSOP  EXEC PGM=IKJEFT01,DYNAMNBR=75
//*********************************************************************
//*
//* Sample TSO logon proc to allocate SVAA TSO and ISPF libraries
//*
//*********************************************************************
//* Make the following changes:
//*
//* -  Use the STEPLIB ddname for hlq.SIBLOAD and hlq.STKLOAD if
//*    these data sets are not in the system link list.
//* -  Use the STEPLIB ddname for hlq.SIBLOAD if this data set
//*    is not in the system link list.
//*
//* -  If the hlq.SACLINK data set is not in the link list, use the
//*    SIBCTRAN or the CTRANS ddname.  To use SIBCTRAN, see member
//*    SIBCTUM in the SIBSAMP library.
//*
//* -  Change the data set names below to match your installation.
//*
//*********************************************************************
//*STEPLIB  DD  DISP=SHR,DSN=hlq.SIBLOAD
//*         DD  DISP=SHR,DSN=hlq.STKLOAD
//*SIBCTRAN DD  DISP=SHR,DSN=hlq.SACLINK
//*CTRANS   DD  DISP=SHR,DSN=hlq.SACLINK
//*
//SYSPROC  DD DISP=SHR,DSN=hlq.SIBCLIB
//         DD DISP=SHR,DSN=SYS1.ISPEXEC
//         DD DISP=SHR,DSN=SYS1.ISRCLIB
//         DD DISP=SHR,DSN=SYS1.TSO.CLIST
//SYSEXEC  DD DISP=SHR,DSN=hlq.SIBCLIB
//         DD DISP=SHR,DSN=SYS1.ISPEXEC
//         DD DISP=SHR,DSN=SYS1.ISRCLIB
//         DD DISP=SHR,DSN=SYS1.TSO.REXX
//ISPCLIB  DD DISP=SHR,DSN=SYS1.ISRCLIB
//ISPLLIB  DD DISP=SHR,DSN=hlq.SIBLOAD
//         DD DISP=SHR,DSN=hlq.STKLOAD
//         DD DISP=SHR,DSN=your.load.library
//ISPPLIB  DD DISP=SHR,hlq.SIBPLIB
//         DD DISP=SHR,DSN=SYS1.ISRPnnn
//         DD DISP=SHR,DSN=SYS1.ISPPnnn
//ISPMLIB  DD DISP=SHR,DSN=hlq.SIBMLIB
//         DD DISP=SHR,DSN=SYS1.ISRMnnn
//         DD DISP=SHR,DSN=SYS1.ISPMnnn
//ISPSLIB  DD DISP=SHR,DSN=hlq.SIBSLIB
//         DD DISP=SHR,DSN=SYS1.ISRSnnn
//         DD DISP=SHR,DSN=SYS1.ISPSLIB
//ISPTLIB  DD DISP=SHR,DSN=pvt.SIBTABL
//         DD DISP=SHR,DSN=hlq.SIBTLIB
//         DD DISP=SHR,DSN=SYS1.ISRTLIB
//         DD DISP=SHR,DSN=SYS1.ISPTnnn
//ISPTABL  DD DISP=SHR,DSN=pvt.SIBTABL
//*
//ISRLDUMY DD DUMMY
//SYSHELP  DD DISP=SHR,DSN=SYS1.HELP
//SYSPRINT DD TERM=TS,SYSOUT=*
//SYSIN    DD TERM=TS
```

Figure 3-14. *Sample TSO logon proc that allocates SVAA libraries*

Sample TSO Logon Exec

The sample logon exec in Figure 3-15 preallocates the ISPF data sets using TSO ALLOCATE commands. A copy of this exec, called SIBISPF, is included in SIBSAMP.

```
/* REXX :  Sample TSO logon exec */

/* Make the following changes:
**
** -  Remove the SIBLOAD and STKLOAD data sets from the ISPLLIB ddname
**    if these data sets are in the system link list or are allocated
**    to the STEPLIB ddname of the TSO logon proc.
**
** -  If the hlq.SACLINK data set is not in the system link list, or
**    is not being allocated in the TSO logon procedure, use the
**    SIBCTRAN or the CTRANS ddname.  To use SIBCTRAN, see member
**    SIBCTUM in the SIBSAMP library.
**
** -  Change the data set names below to match your installation.
*/
 "ALLOC FI(ISPLLIB )  DA( 'hlq.SIBLOAD'                  " ,
 "                        'hlq.STKLOAD'                  " ,
 "                        'your.load.library' )  SHR REUSE"

 "ALLOC FI(SIBCTRAN)  DA( 'hlq.SACLINK'       )  SHR REUSE"

 "ALLOC FI(SYSEXEC )  DA( 'hlq.SIBCLIB'                  " ,
 "                        'SYS1.ISPEXEC'                 " ,
 "                        'SYS1.ISRCLIB'                 " ,
 "                        'SYS1.TSO.CLIST'    )  SHR REUSE"

 "ALLOC FI(SYSPROC )  DA( 'hlq.SIBCLIB'                  " ,
 "                        'SYS1.ISPEXEC'                 " ,
 "                        'SYS1.ISRCLIB'                 " ,
 "                        'SYS1.TSO.REXX'     )  SHR REUSE"

 "ALLOC FI(ISPPLIB )  DA( 'hlq.SIBPLIB'                  " ,
 "                        'SYS1.ISRPnnn'                 " ,
 "                        'SYS1.ISPPnnn'      )  SHR REUSE"

 "ALLOC FI(ISPMLIB )  DA( 'hlq.SIBMLIB'                  " ,
 "                        'SYS1.ISRMnnn'                 " ,
 "                        'SYS1.ISPMnnn'      )  SHR REUSE"

 "ALLOC FI(ISPSLIB )  DA( 'hlq.SIBSLIB'                  " ,
 "                        'SYS1.ISRSnnn'                 " ,
 "                        'SYS1.ISPSLIB'      )  SHR REUSE"

 "ALLOC FI(ISPTLIB )  DA( 'pvt.SIBTABL'                  " ,
 "                        'hlq.SIBTLIB'                  " ,
 "                        'SYS1.ISRTLIB'                 " ,
 "                        'SYS1.ISPTnnn'      )  SHR REUSE"

 "ALLOC FI(SIBTABL )  DA( 'pvt.SIBTABL'       )  SHR REUSE"

 EXIT
```

Figure 3-15. *Sample TSO logon exec to preallocate ISPF data sets*

Sample REXX Exec Using LIBDEF Commands

The sample REXX exec in Figure 3-16 preallocates the SVAA ISPF libraries with LIBDEF commands. A copy of this exec, called SIBLDEF, is included in the SIBSAMP data set.

```
/* Sample REXX EXEC to preallocate the SVAA ISPF libraries */
/* using ISPxxxx ddnames                                    */

/*
** Allocate the SVAA application level EXEC library.
** (Change the data set name below to match your installation.)
*/
ADDRESS TSO
"ALTLIB ACTIVATE APPLICATION(EXEC) DSNAME('hlq.SIBCLIB')"

/*
** Allocate the SVAA application level ISPF libraries.
** (Change the data set names below to match your installation.)
*/
ADDRESS ISPEXEC
"LIBDEF ISPPLIB DATASET ID('hlq.SIBPLIB') COND"
"LIBDEF ISPMLIB DATASET ID('hlq.SIBMLIB') COND"
"LIBDEF ISPSLIB DATASET ID('hlq.SIBSLIB') COND"

/*
** Allocate the SVAA table library.
** (Change the data set names below to match your installation.)
*/
"LIBDEF ISPTLIB DATASET ID('pvt.SIBTABL'
                           'hlq.SIBTLIB') COND"
/*
** Allocate a private SVAA table output library.
** (Change the data set name below to match your installation.)
*/
"LIBDEF ISPTABL DATASET ID('pvt.SIBTABL') COND"

EXIT
```

Figure 3-16. *Sample exec to preallocate ISPF libraries using ISPxxxx ddnames*

Sample REXX Exec Using TSO ALLOCATE Commands

The sample REXX exec in Figure 3-17 preallocates the SVAA ISPF libraries using TSO ALLOCATE commands. A copy of this exec, called SIBALLOC, is included in the SIBSAMP data set.

```
/* Sample REXX EXEC to preallocate the SVAA ISPF libraries */
/* using SIBxxxx ddnames                                    */

/*
** Allocate the SVAA application level EXEC library.
** (Change the data set name below to match your installation.)
*/
ADDRESS TSO
"ALTLIB ACTIVATE APPLICATION(EXEC) DSNAME('hlq.SIBCLIB')"

/*
** Allocate the SVAA application level ISPF libraries.
** (Change the data set names below to match your installation.)
*/
"ALLOCATE FI(SIBPLIB) DA('hlq.SIBPLIB') SHR"
"ALLOCATE FI(SIBMLIB) DA('hlq.SIBMLIB') SHR"
"ALLOCATE FI(SIBSLIB) DA('hlq.SIBSLIB') SHR"

/*
** Allocate the SVAA table library.
** (Change the data set names below to match your installation.)
*/
"ALLOCATE FI(SIBTLIB) DA('pvt.SIBTABL'
                         'hlq.SIBTLIB') SHR"

/*
** Allocate a private SVAA table output library.
** (Change the data set name below to match your installation.)
*/
"ALLOCATE FI(SIBTABL) DA('pvt.SIBTABL') SHR"

EXIT
```

Figure 3-17. *Sample exec to preallocate ISPF libraries using SIBxxxx ddnames*

Step 14c: Setting up to Run SVAA under ISPF

You can invoke the SVAA dialog from ISPF with a CLIST or REXX exec. The following CLIST invokes the SVAA dialog.

```
PROC 0 /* SVAA MAIN MENU */

 ISPEXEC SELECT CMD(SIBREMMO) NOCHECK

EXIT
```

Figure 3-18. *CLIST to call SVAA primary panel exec*

If you prefer, you can add an SVAA selection to an ISPF menu. To add SVAA to the ISPF primary panel so that you can access SVAA functions from ISPF, edit the ISR@PRIM member of the ISPPLIB data set. To this member, add the highlighted (boldface) lines shown in the example in Figure 3-19. You can also execute SVAA from a secondary panel.

```
%------------------------  SAMPLE PRIMARY OPTION MENU  -----------------------
%OPTION  ===>_ZCMD                                                             +
%                                                          +USERID   - &ZUSER
%   0 +ISPF PARMS  - Specify terminal and user parameters   +TIME     - &ZTIME
%   1 +COMMANDS    - Create/change command table            +TERMINAL - &ZTERM
%   2 +ISPPREP     - Preprocessed panel utility             +PF KEYS  - &ZKEYS
%   3 +ISPDTLC     - ISPF DTL Conversion Utility
%   4 +.           - (Description for option 4)
%   5 +.           - (Description for option 5)
%   7 +DIALOG TEST - Perform dialog testing
%   I +SVAA        - Shared Virtual Array Administrator
%   T +TUTORIAL    - Display information about this application
%   X +EXIT        - Terminate ISPF using list/log defaults
%
+Enter%END+command to terminate application.
%
+5684-043 (C) COPYRIGHT IBM CORP. 1980, 1990
)INIT
  .HELP = ISP00003      /* Help for this panel                  */
  &ZPRIM = YES          /* This is a primary option menu        */
  &ZHTOP = ISP00003     /* Tutorial table of contents for this appl*/
  &ZHINDEX = ISP91000   /* Tutorial index - 1st page for this appl */
  VPUT(ZHTOP,ZHINDEX) PROFILE
)PROC
  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
                0,'PANEL(ISPOPTA)'
                1,'PANEL(ISPUCMA)'
                2,'PGM(ISPPREP) NEWAPPL'
                3,'CMD(%ISPDTLC)'
                7,'PGM(ISPYXDR) PARM(ISP) NOCHECK'
            /*******************************************************/
            /*                                                     */
            /* Add other applications here.                        */
            /*                                                     */
            /*******************************************************/
                I,'CMD(SIBREMMO) NOCHECK'
                T,'PGM(ISPTUTOR) PARM(ISP00000)'
              ' ',' '
                X,'EXIT'
                *,'?' )
  &ZTRAIL = .TRAIL
)END
```

Figure 3-19. *Sample ISPF primary panel*

Step 14d: Customizing the TSO Session

You can set SVAA options for TSO sessions with the SVAA subcommands: SET, DISPLAY, and QUERY. (See *SVAA for OS/390 Configuration and Administration* or *SVAA for OS/390 Reporting* for subcommand syntax.)

SVAA subcommands to customize TSO can be invoked from either:

- the SIBSTK*xx* parmlib member (described in Step 10), or
- a REXX EXEC or TSO CLIST member named PROFSIBA.

Important!

If the SIBSTK*xx* parmlib member or the PROFSIBA REXX EXEC or CLIST exists, it is executed when SIBADMIN is invoked. One of these *should* contain a SET ECAMDEVICE subcommand specifying at least one device per SVA subsystem. Statements in PROFSIBA override those in SIBSTK*xx*, except that the SET ECAMDEVICE subcommands in *both* sources are used.

The example in Figure 3-20 is in the PROFSIBA member of SIBSAMP. For a REXX EXEC, copy this member to a library allocated to SYSPROC or SYSEXEC and add or modify as many statements as you wish.

```
/* REXX profile macro for SIBADMIN */
address SIB
"set case(mixed)"
"set ecamdev(100)"
exit
```

Figure 3-20. *Sample REXX EXEC for customizing SIBADMIN*

The example in Figure 3-21 is in the PROFSIBC member of SIBSAMP. For a CLIST, copy this member to a library allocated to SYSPROC, rename it PROFSIBA, and add or modify as many statements as you wish.

```
PROC 1 PROGRAM
CONTROL NOMSG
DATA
SET CASE(U)
SET ECAMDEV(200)
ENDDATA
EXIT
```

Figure 3-21. *Sample CLIST for customizing SIBADMIN*

Note: With the SVAA ISPF panels, you can specify additional options for your SVAA session with the SVAA SESSION PROFILE panel (described in *SVAA for OS/390 Configuration and Administration* and *SVAA for OS/390 Reporting*).

Step 15: Defining Security Resources

To install and use SVAA you must activate certain security classes and define resources to your security system. The tables in the following sections summarize the security resources checked by SVAA. You may already have defined some of these resources; others you may have to modify or add. See the documentation for your security system for information about activating a class and defining a resource.

Note: If you do not have a security system installed and if security is required, you must code the SIBIOATX and SIBSNDAX security exits (Step 16 on page 3-36).

Checking ECAM-Level Security Resources

Table 3-1 summarizes the ECAM-level resources checked by SVAA.

For the first two rows in the table, the values shown for Class, Resource name, and Access are the defaults; the SIBIOATX and SIBSNDAX security exits may change them.

Table 3-1. *ECAM-level security resources checked by SVAA*

Class	Resource name	Access	When used
FACILITY	STGADMIN. IDC.LISTDATA	READ	Before sending an unrestricted control message to a DASD subsystem (default action). (Appendix A, “SVAA Communication Devices” describes unrestricted messages.) Before sending a command to the reporting or SIM tasks in the SVAA address space from a TSO user or batch job (default action).
FACILITY	STGADMIN. IDC.SETCACHE	UPDATE	Before sending a restricted control message to a DASD subsystem (default action). (Appendix A, “SVAA Communication Devices” describes restricted messages.) Before sending a command to the VOP or DSR tasks in the SVAA address space (default action).
FACILITY	IXFP.FORCE. DELETE.FUNC.DEV	ALTER	When deleting an Alias device, or when deleting a functional device that has back-end storage and that: 1) is not defined to this host operating system, *or* 2) is disabled, *or* 3) has a volser that cannot be accessed. In these cases, you must have RACF FACILITY class ALTER authority for the generalized resource IXFP.FORCE.DELETE.FUNC.DEV. Since Alias devices are always considered to not be defined to the host, the deletion of Aliases will always use the RACF FACILITY class and the generalized resource IXFP.FORCE.DELETE.FUNC.DEV profile for security checking. (See the description of the FORCE parameter of the DELETE DEVICE subcommand in the *SVAA for OS/390 Configuration and Administration* manual.) Existing IXFP security mechanisms, including RACF resource names and security exits, will work without changes in SVAA.

Warning: SVAA rejects any command that requires “UPDATE” authority when the names of the security resources that the command uses are not defined in your security system.

Checking MVS- or User-Level Security Resources

Table 3-2 summarizes the OS/390- or user-level security resources checked by SVAA.

Table 3-2. *MVS- or user-level security resources checked by SVAA*

Class	Resource name	Access	When used
DASDVOL	Volser	ALTER	When deleting a functional device that is defined to this host operating system *and* for which the volser and VTOC can be accessed *and* on which SVAA determines user data exists. If the DASDVOL class is not active on this host operating system, before you attempt to delete the device you must either: 1) disable the device (use the ALTER DEVICE command and the CKDENA(NO) parameter), 2) vary all paths offline to the device on this host operating system, or 3) implement the SIBIOATX user exit, otherwise the delete will fail with a SIB1164E message indicating that you have insufficient authority to access the subsystem.
FACILITY	STGADMIN.ANT. PPRC.COMMANDS	READ	SVAA SAF services first checks the user's authority against this resource prior to issuing any PPRCOPY subcommand.
FACILITY	STGADMIN.ANT. PPRC.CQUERY	READ	If the PPRC QUERY subcommand is issued but was not permitted with the above resource, SVAA SAF services attempts a second check for authorization to issue the QUERY subcommand against this resource.

Access to Logging Files

Unless you are using SMF for data collection, SVAA stores collected subsystem data in a user-defined file. You must arrange for your site security administrator to grant update authority to the SVAA started task for access to the output files.

Step 16: Customizing SVAA User Exits

This step briefly describes the SVAA user exits that are invoked during SVAA processing. Detailed information about the user exits, including the SVAA default actions, appears in Appendix E, “SVAA User Exits.”

Note: Ordinarily, you need to install the SIBIOATX and SIBSNDAX exits only if you have no security system or if the resources are not defined. However, if you are not satisfied with the default actions of SVAA, you can customize the SIBIOATX and SIBSNDX exits.

You can enable and/or customize SVAA user exits now or at any time after this point. If you decide to enable them at a later time, you *will not* have to reinitialize your OS/390 system or refresh SVAA. If, however, you ***modify*** a user exit that you were already using, you ***must*** refresh SVAA. (Instructions for refreshing SVAA are provided in Chapter 5.)

Any user exit (listed in Table 3-3) that you want to use must be in a link list, STEPLIB, or JOBLIB data set, and it must be in an APF-authorized library. The first time a user exit is called, it is loaded into a common storage area and remains there as long as the SVAA subsystem remains on your OS/390 system.

The SIBSAMP data set contains the sample SIBIOATX, SIBSNDAX, and SIBSRPSW exit routines with their default actions. The mapping macros for the data areas are in the SIBMAC data set.

Table 3-3. *Summary of SVAA user exits and functions*

User Exit Name	Function	Description
SIBIOATX	Authorize subsystem I/O	Verifies a user's authority: • to send a particular ECAM message to a subsystem (security definition required). • to use the DELETE DEVICE subcommand (security definition required). If your OS/390 system does not have a security product that supports SAF-type requests, you must customize and install the SIBIOATX user exit.
	Authorize PPRC subcommands	Verifies a user's authority to execute a particular PPRC subcommand (security definition required). If your OS/390 system does not include a security product that supports SAF-type requests, you must customize and install the SIBOATX user exit.
SIBSNDAX	Authorize sending commands	Verifies a user's authority to send a command to a subtask in the SVAA address space. If you need to communicate with the SVAA address space from TSO or a batch job, you must customize and install the SIBSNDAX user exit.
SIBSRPSW	Handle full logging file	Invoked by the Reporter data collection task before closing a logging file, switching logging files, or when a logging file becomes full. If you wish to use the main and alternate logging files rather than SMF, you must customize and install the sample SIBSRPSW routine, as described in “Customizing the SIBSRPSW Exit” on page E-8.

Step 17: Customizing the Logging Facility

To be able to view log messages at the OS/390 system console, the administrator for the OS/390 server must make appropriate changes to the syslogd configuration on UNIX Systems Services (USS). Messages written to the OS/390 system log are written to the *user* logging facility and have *emerg, alert, crit*, and *err* severities. Normally, this is done by adding the following line to the /etc/syslog.conf file:

```
user.*/dev/console
```

However, you should check the documentation for syslogd configuration for your specific installation.

Chapter 4. Verifying SVAA Installation

You can initialize the SVAA subsystem either at IPL time or with a SIBMVSS batch job after IPL is complete. The factors to consider are these:

- If you initialize the SVAA subsystem when you IPL, system initialization is slower, but Dynamic DDSR (if you are using it) can be in effect as soon as SVAA initialization is complete.

  Because OS/390 subsystem initialization is serialized, your IPL time is lengthened by the amount of time required to initialize the SVAA subsystem.

- If you initialize the SVAA subsystem later by running a SIBMVSS batch job, subsystem initialization is faster, but Dynamic DDSR (if you are using it) is not in effect until SIBMVSS is complete.

Step 18	Enabling parmlib changes
Step 19	Initializing SVAA with SIBMVSS
Step 20	Verifying the SVAA initialization
Step 21	Verifying SVAA under TSO
Step 22	Verifying SVAA under ISPF
Step 23	Verifying SVAA under SIBBATCH

After installing the Shared Virtual Array Console (SVAC):

Step 24	Verifying SVAA using SVAC

Step 18: Enabling Parmlib Changes

PROGxx or LNKLSTxx

(See Step 8a)

After making your SVAA changes to the PROG*xx* member,

- use the SET PROG command to enable dynamically ***all*** PROG*xx* changes, or
- issue individual SETPROG commands to dynamically create a new link list (DEFINE), add the new SVAA libraries (ADD), delete any IXFP libraries (DELETE), activate the new link list (ACTIVATE), and update the Master Scheduler address space to use the new link list (UPDATE).

Remember that if you use the LNKLST*xx* member to control the link list, you can't dynamically change the names of the libraries used; therefore, using the PROG*xx* member is recommended. Refer to the IBM *MVS System Commands* manual for the exact syntax and further information.

For example:

```
SET PROG=xx
```

Or:

```
SETPROG LNKLST,DEFINE,NAME=newlnklstname,COPYFROM=currentlnklstname
SETPROG LNKLST,ADD,DSN=hlq.SACLINK,VOL=volser
     .
     .
SETPROG LNKLST,DELETE,DSN=ixfp.SACLINK
     .
     .
SETPROG LNKLST,ACTIVATE,NAME=newlnklstname
SETPROG LNKLST,UPDATE,ASID=01
```

PROGxx or IEAAPFxx

(See Step 8b)

After making your SVAA changes to the PROG*xx* member,

- use the SET PROG command to enable dynamically ***all*** PROG*xx* changes, or
- issue individual SETPROG commands to dynamically APF authorize each SVAA library.

Remember that if you use the IEAAPF*xx* member to control APF authorization, an IPL is required; therefore, using the PROG*xx* member is recommended. Refer to the IBM *MVS System Commands* manual for the exact syntax and further information.

For example:

```
SET PROG=xx
```

Or:

```
SETPROG APF,ADD,DSN=hlq.SACLINK,VOL=volser
```

IECIOSxx

(See Step 8c)

Use the SET IOS command to enable dynamically your SVAA changes to the IECIOS*xx* member. Refer to the IBM *MVS System Commands* manual for the exact syntax and further information.

For example:

```
SET IOS=xx
```

IEFSSNxx

(See Step 8d)

After making your SVAA changes to the IEFSSN*xx* member, use the SETSSI command with only the SUBNAME parameter to dynamically add the subsystem without an IPL. Note that, because SVAA is not a dynamic subsystem, you cannot specify parameters such as ACTIVATE and DEACTIVATE. Refer to the IBM *MVS System Commands* manual for the exact syntax and further information.

For example:

```
SETSSI ADD,SUBNAME=name
```

SMFPRMxx

(See Step 8e)

Use the SET SMF command to enable dynamically your SVAA changes to the SMFPRM*xx* member. Refer to the IBM *MVS System Commands* manual for the exact syntax and further information.

For example:

```
SET SMF=xx
```

Step 19: Initializing the SVAA Subsystem with SIBMVSS

By executing the SIBMVSS program—which supports the ALTER SSNAME command—you can initialize, reinitialize, or terminate either:

- the SVAA subsystem, or
- one or more SVAA resource initialization modules (RIMs).

You can invoke SIBMVSS in batch mode or through a started-task procedure; you *cannot* invoke SIBMVSS under TSO.

Note: Initializing SVAA with SIBMVSS performs the same tasks that are invoked during IPL by specifying INIT(Y) in IEFSSN*xx*—including full subsystem initialization, allocation and initialization of data structures, loading of global service routines, and device mapping. For more information about device mapping, see Appendix D, “Device Mapping.”

An example of JCL to invoke the ALTER SSNAME command in batch mode is shown in Figure 4-1. (This sample JCL is in the SVAASS member of the SIBSAMP library.)

```
//jobname   JOB (job-accounting-info)
//*
//* SVAA subsystem JCL
//* Initialize, reinit, or terminate the subsystem
//*
//        EXEC PGM=SIBMVSS,TIME=5
//*
//STEPLIB  DD  DISP=SHR,DSN=hlq.SIBLOAD     If not in link list
//         DD  DISP=SHR,DSN=hlq.STKLOAD     If not in link list
//         DD  DISP=SHR,DSN=hlq.SACLINK     If not in link list
//*
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
ALTER SSNAME(NAME(SVAA) INIT PARAM('INIT(Y),DYNDDSR(A)'))
/*
```

Figure 4-1. *Sample JCL to invoke ALTER SSNAME command*

Notes:

1. Include the STEPLIB DD statement if the *hlq*.SIBLOAD data set, *hlq*.STKLOAD data set, or *hlq*.SACLINK data set is not in the link list.
2. If you are executing SIBMVSS as a PROC and the PROC name is the same as the subsystem name specified in IEFSSN*xx*, you must specify **'S** *procname***,SUB=JES2'** to start the task.

Syntax for ALTER SSNAME command

ALTER SSNAME

```
►►──ALTER SSNAME──(──NAME(ssname)──┬─INITialize───┬──┬──────────────────────────┬──────────────►
                                   ├─REINITialize─┤  │        ┌─────────────┐   │
                                   └─TERMINATE────┘  └─RIM(──▼─┬─PC──────┬─┴─)─┘
                                                               ├─DMQ─────┤
                                                               ├─ENF─────┤
                                                               ├─DYNDDSR─┤
                                                               └─VP──────┘

►──┬─────────────────────────────────┬──┬──────────────┬──┬───────┬──)──────────────►◄
   └─PARAMeter(startup-parameters)───┘  └─DeinstalleXits─┘  └─FORCE─┘
```

Note: The VP option for the RIM parameter is specifically for SnapShot. (VP is ignored if SnapShot is not installed.)

Parameters

NAME (ssname)

Required

This parameter specifies the SVAA subsystem for which you want to invoke the ALTER SSNAME command.

Note: OS/390 must have been enabled previously with the SVAA subsystem definition in the IEFSSN*xx* parmlib member. (See “Step 18: Enabling Parmlib Changes” on page 4-2.)

Values:	*ssname* is the SVAA subsystem name (from SYS1.PARMLIB(IEFSSN*xx*)).
Default value:	None

INIT | REINIT | TERMINATE

Required

This parameter specifies the action to be taken on the SVAA subsystem or one or more resource initialization modules.

Values:	**INITialize** initializes the SVAA subsystem for the first time.
	REINITialize reinitializes the SVAA subsystem after you have made some changes, such as to DDSR.
	TERMINATE terminates SVAA. You must first terminate any associated SVAA started task.
Default value:	None

RIM

Optional

This parameter specifies one or more SVAA resource initialization modules (RIMs) on which the command is to act. When you use this parameter, the ALTER SSNAME command affects only the specified RIM components of the SVAA subsystem—not the entire subsystem.

Note: You can specify these values in any combination from none (the default) to all four.

Values:	**PC** performs RIM processing for the SVAA OS/390-authorized services functions. **DMQ** performs RIM processing for device mapping services. **ENF** performs RIM processing for event notification (e.g., vary online). **DYNDDSR** performs RIM processing for Dynamic DDSR. The status of Dynamic DDSR is set to Inactive (the default). **VP** (for SnapShot only) performs RIM processing for volume preferencing. The status of volume preferencing is set to Active (the default). (VP is ignored if SnapShot is not installed.)
Default value:	None

PARAMETER (startup-parameters)

Optional—but these startup parameters are ignored if the RIM ***parameter is specified.***

This parameter specifies startup parameters to be used in the initialization of SVAA. (Note that the ALTER SSNAME command does not read the IEFSSN*xx* parmlib member for initialization parameters.) If no parameters are specified, the SVAA default parameters are used.

Abbreviation:	**PARAM**
Values:	Any of the optional startup parameters (**DYNDDSR, INIT, KEY, LANG, MIH,** and **MSGP**) listed under "Customizing the IEFSSNxx Member" on page 3-4. **Note:** A sequence of subparameters entered with the PARAM parameter must be enclosed within single quotes if the text string contains blanks.

DEINSTALLEXITS

Optional

If you have SnapShot installed, this parameter removes the residual volume preferencing control of the DADSM intercept vector for IGGDAC02.

Note: SVAA termination does not require the DX parameter unless you are using products that require hook removal (for example, SAMS:Allocate). Also, for the hook removal to be successful, any product installed after SVAA must be removed first.

Abbreviation: **DX**

FORCE

Optional—not recommended

If FORCE is specified and SVAA encounters an error in executing the command, SVAA continues to try to execute the command.

Examples of ALTER SSNAME Command Syntax

```
ALTER SSNAME (NAME(SS1) INIT PARAM(MSGP(Y)))
ALTER SSNAME (NAME(SS2) INIT PARAM('DYNDDSR(A) MIH(W)'))
ALTER SSNAME (NAME(SS3) INIT PARAM('LANG(UCENG)'))
ALTER SSNAME (NAME(SS3) REINIT RIM(PC DYNDDSR))
ALTER SSNAME (NAME(SS4) TERMINATE DX)
```

Step 20: Verifying SVAA Initialization

After you have initialized your SVAA subsystem, you can issue the following command from the OS/390 system console:

D GRS,RES=(SYSZSVAA,*)

This command requests GRS (Global Resource Serialization) to list all resources with the major name of SYSZSVAA (SVAA uses this authorized resource name internally to register active SVAA subsystems; the name should not be used by other programs).

You receive output similar to the following:

```
ISG020I 12.05.21 GRS STATUS 478
S=SYSTEM  SYSZSVAA     SVAA          T W
SYSNAME        JOBNAME         ASID    TCBADDR   EXC/SHR  OWN/WAIT
MVS9      *MASTER*          0001       007FF778 EXCLUSIVE   OWN
```

The IBM *MVS/ESA System Messages* manual for message ISG020I describes the fields as:

```
S=scope   qname       rname          flag
SYSNAME        JOBNAME         ASID    TCBADDR   EXC/SHR  OWN/WAIT
sysname   jobname           asid       tcbaddr  exc/shr  own/wait
```

Notice the highlighted (bold) fields:

- The qname is SYSZSVAA, the authorized qname used by SVAA.
- The rname is a hexadecimal string containing the name of the initialized SVAA subsystem, as well as some other data for internal SVAA use.
- The jobname is *MASTER*, because the SVAA subsystem is initialized within the MASTER address space. In this case, the SVAA subsystem is active.
- If you issue this command when your SVAA started task is available, you receive two sets of messages. The second set has the name of your SVAA started task as the jobname.

If you issue this command when the subsystem is not available, you receive the following message:

```
ISG020I 12.05.21 GRS STATUS 504
NO REQUESTORS FOR RESOURCE SYSZSVAA *
```

Note: SVAA uses SIBSVAA for the non-authorized major name.

Step 21: Verifying SVAA under TSO

If your SVA hardware is ***not installed***, you will receive a message from either of the following tests indicating that SVAA is unable to locate any SVA subsystems.

Batch Test

To verify that your installed SVAA software runs in batch mode, run Step 2 of the **IVP2** member of the SIBSAMP data set after you have finished installing SVAA.

Modify the appropriate statements in the JCL provided in IVP2 and submit the job. In the JCL, change the data set name on the STEPLIB statement to the name you used during SVAA installation.

```
//ivp2     JOB 'accounting-info'
//*****************************************************************
//* Step 2 executes SVAA in batch TSO
//*****************************************************************
//STEP2  EXEC  PGM=IKJEFT01,DYNAMNBR=50
//*STEPLIB  DD  DSN=hlq.SIBLOAD,DISP=SHR
//*         DD  DSN=hlq.STKLOAD,DISP=SHR
//*SIBCTRAN DD  DSN=hlq.SACLINK,DISP=SHR
//*CTRANS   DD  DSN=hlq.SACLINK,DISP=SHR
//*STKPARMS DD  DSN=hlq.sibparms,DISP=SHR
//*
//SYSTSPRT DD  SYSOUT=(,)
//SYSTSIN  DD  *
SIBADMIN
 QUERY VERSION
 QUERY SSNAME
 QUERY SUBSYSTEM(ICEBERG)
 QUERY SUBSYSTEM(NONICEBERG)
END
/*
```

Figure 4-2. *JCL for verifying SVAA under TSO*

Online Test

To verify SVAA online, logon to TSO with your new logon proc which allocates the TSO data sets.

Before running this test, add a SET ECAMDEVICE subcommand to SIBSTK*xx* in your STKPARMS data set.

From **TSO**, enter:

```
SIBADMIN QUERY SUBSYSTEM(ICEBERG)
```

Step 22: Verifying SVAA under ISPF

Once you have logged onto TSO and have allocated your SVAA data sets, you are ready to begin using the SVAA panels. (Remember that you need a SET ECAMDEVICE subcommand in either PROFSIBA or SIBSTK00 if you want to access your SVA subsystem).

In the following steps, you move through a short sequence of panels to the SUBSYSTEM CONFIGURATION panel to see the current status of your SVA subsystem configuration.

1. Enter ISPF and go through the panels to get to SVAA, as defined for your installation. SVAA displays a copyright panel and then the SVAA main menu, as shown in Figure 4-3.

```
                    *** SHARED VIRTUAL ARRAY ADMINISTRATOR ***
OPTION  ===>

   0  OPTIONS       - Specify user parameters
   1  CONFIGURATION - Configure and administer subsystems
   2  SELECTION     - Maintain device selection lists
   3  REPORT        - Perform subsystem data collection and reporting
   X  EXIT          - Terminate processing


Enter END command to terminate processing
```

Figure 4-3. *SVAA main menu (SIBMM00)*

2. Select option **1** (type **1** and press **Enter**).

   SVAA displays the SUBSYSTEM ADMINISTRATION panel.

3. Select option **2** to browse the current configuration information.

   SVAA displays the SUBSYSTEM COMPONENT SELECTION panel.

4. Select option **1** to display subsystem characteristics.

   At this point, if you have defined an ECAM device for more than one SVA subsystem, SVAA displays the SUBSYSTEM SELECTION panel. (If this occurs, select a subsystem by typing **S** to the left of a subsystem name and press **Enter**.)

   SVAA displays the SUBSYSTEM CONFIGURATION panel. (This panel is the end of this sequence.)

5. Enter the ISPF **end** command to exit this panel and each succeeding panel until you exit SVAA.

If these steps have gone smoothly, you are ready to run SVAA under ISPF.

For more information about using the panels, see the SVAA Panels chapter in the *SVAA for OS/390 Configuration and Administration* manual.

Step 23: Verifying SVAA under SIBBATCH

The SIBBATCH program provides another way to execute SVAA subcommands in batch mode. The following step involves modifying some sample JCL and running a SIBBATCH job.

To verify your SVAA software with SIBBATCH, run Step 1 of the IVP2 member of the SIBSAMP data set after you have finished installing SVAA.

Modify the appropriate statements in the JCL provided in the IVP2 member and submit the job. In the JCL, change the data set name on the STEPLIB statement to the name you used during SVAA installation.

The SIBLOAD and SACLINK data sets are needed if they are not in the link list concatenation. Refer to “Step 11: Verifying Library Requirements” for more information about these data sets.

Note: If your SVA hardware is ***not installed***, you will receive a message from this test indicating that SVAA is unable to locate any SVA subsystems.

```
//ivp2     JOB 'accounting-info'
//*****************************************************************
//* Step 1 executes the SVAA batch program
//*****************************************************************
//STEP1  EXEC  PGM=SIBBATCH
//*STEPLIB  DD  DSN=hlq.SIBLOAD,DISP=SHR
//*         DD  DSN=hlq.STKLOAD,DISP=SHR
//*SIBCTRAN DD  DSN=hlq.SACLINK,DISP=SHR
//*CTRANS   DD  DSN=hlq.SACLINK,DISP=SHR
//*STKPARMS DD  DSN=hlq.sibparms,DISP=SHR
//*
//SYSTERM  DD  SYSOUT=(,)
//SYSPRINT DD  SYSOUT=(,)
//SYSIN    DD  *
 QUERY VERSION
 QUERY SSNAME
 QUERY SUBSYSTEM(ICEBERG)
 QUERY SUBSYSTEM(NONICEBERG)
/*
```

Figure 4-4. *JCL for verifying SVAA in batch*

Step 24: Verifying SVAA using Shared Virtual Array Console

To verify that you can run SVAA using the SVA Console:

1. Install SVA Console 1.1.0 using the instructions in the *SVA Console Quick Start Guide*.
2. Start the SVA Console program.

   Note: Before proceeding, wait for the message:
   SIB7011I SVAA *server* **initialization complete.**
3. Under **Connect**, select **Server Software**.
4. Enter either the Host Name or the IP Address.
5. Enter either:
   - the Server Software Name, as specified in SIBSFCxx with the SET SERVERNAME subcommand, or
   - the TCP/IP Port, as specified in SIBSFCxx with the SET SIBPORT subcommand.

     See the *SVAA for OS/390 Configuration and Administration* manual for subcommand documentation.
6. When you have successfully connected to SVAA, you will see a subsystem name for each ECAM device that has been defined in SIBSFCxx.

Chapter 5. Refreshing SVAA

Once SVAA has been completely installed and is running, you may be able to refresh elements changed by a PTF without an IPL. Listed below are four installation methods followed by suggestions on when to use each method.

Installation methods

1. IPL with the new elements in the link list.
2. Recycle the SVAA subsystem or a specific SVAA RIM.
   - Terminate any SIBADMIN, SIBBATCH or SVAA ISPF sessions.
   - Stop the SVAA started task.
   - Terminate the SVAA subsystem using SIBMVSS.
   - Copy elements to their libraries.
   - Refresh LLA.
   - Initialize the SVAA subsystem using SIBMVSS.
   - Start the SVAA started task.
3. Recycle the SVAA started task, batch and online users.
   - Terminate any SIBADMIN, SIBBATCH or SVAA ISPF sessions.
   - Stop the SVAA started task.
   - Copy element to library.
   - Refresh LLA (if the library is in the link list).
   - Start the SVAA started task.
4. Recycle the SIBADMIN, SIBBATCH and SVAA ISPF users.
   - Terminate any SIBADMIN, SIBBATCH and SVAA ISPF sessions.
   - Copy elements to libraries.
   - Restart SIBADMIN, SIBBATCH and SVAA ISPF sessions.

When to use which method

- SIBLINK Elements:
  - If you can afford an IPL, this is always the first choice, especially if a large amount of maintenance has been installed.

    Note: An IPL is necessary to implement PTFs containing modules SIBRDDSR, SIBPCNS0, and SIBPCALT.
  - If you need to avoid an IPL and you are not certain that a particular RIM element has been modified, you can use SIBMVSS to recycle the subsystem (installation method 2).

  - If you are certain that the change affects only a particular RIM element, you can recycle a subsystem RIM with SIBMVSS (installation method 2).

- SIBLOAD, SACLINK, or STKLOAD Elements:
  - These elements can be refreshed by recycling the SVAA started task (installation method 3).
- TSO and ISPF elements:
  - For updates to libraries SIBCLIB, SIBPLIB, SIBMLIB, SIBTLIB, and SIBSLIB recycle the TSO or ISPF session (installation method 4).

Any unusual requirements for refreshing SVAA following the installation of a PTF will be documented in the PTF cover letter and the PTF will have a ++HOLD for ACTION.

Chapter 6. Installing the Maintenance PTFs

This chapter describes the procedure for installing the PTFs from the SVAA Maintenance Tape.

The tape is a 3480 non-label tape containing the files listed in Table 6-1.

Table 6-1. *Maintenance Tape contents*

File No.	Data Set Format	Description
1	SMPPTFIN	All PTFs issued since Base tape
2	IEBGENER	Print file of all PTF cover letters
3	IEBGENER	Print file of summary info for all PTFs
4	SMPPTFIN	SMP/E external hold statements
5	IEBUPDTE	Current installation JCL

If you have not already done so, contact StorageTek Software Support for any PTFs available since your Maintenance tape was created. Instructions for contacting customer support are located in the StorageTek publication: *Requesting Help from Software Support*.

Current installation JCL is in file 5 of the Maintenance tape. See “Step 2: Loading the Installation JCL” on page 2-3 for JCL to load the file to disk. You can use the Installation JCL members, the SMP/E SYSMOD Management dialogs, or the printed sample JCL to complete the SMP/E installation of the maintenance PTFs. This consists of the following tasks:

- Receiving the Maintenance PTFs
- Resolving HOLDDATA
- Applying the Maintenance PTFs
- Accepting the Maintenance PTFs

Receiving the Maintenance PTFs

Use the sample JCL in **P1RCV** to receive the Maintenance PTFs and HOLDDATA from the Maintenance tape. Obtain the actual volume serial number from the tape cartridge external label.

```
//jobname   JOB    'accounting info'
//*
//          EXEC   smpeproc
//SMPPTFIN  DD     DSN=PTF,DISP=SHR,
//                 VOL=(,RETAIN,SER=mnttap),
//                 UNIT=3480,
//                 LABEL=(1,NL,EXPDT=98000),
//                 DCB=(RECFM=FB,LRECL=80,BLKSIZE=7200)
//SMPHOLD   DD     DSN=HOLDDATA,DISP=SHR,
//                 VOL=(,RETAIN,SER=mnttap),
//                 UNIT=AFF=SMPPTFIN,
//                 LABEL=(4,NL,EXPDT=98000),
//                 DCB=(RECFM=FB,LRECL=80,BLKSIZE=7200)
//SMPCNTL   DD     *

 SET BOUNDARY ( GLOBAL ) .

 RECEIVE SYSMODS         /* Receive all SYSMODs          */
         HOLDDATA        /* and any exception data.      */
         .               /*                              */

 LIST    SYSMODS         /*  List the cover letter       */
         MCS             /*  for each SYSMOD in          */
         .               /*  this zone.                  */
```

Figure 6-1. *JCL member P1RCV: Receiving the Maintenance PTFs*

Resolving HOLDDATA

Some PTFs may have special requirements that must be met before they can be installed. These PTFs are prevented from being installed by the use of SMP/E HOLDDATA. Before running the Apply job, review the output from the Receive job for HOLDDATA, and follow the instructions given for each held PTF. Once the requirements are met, you can bypass the hold condition for that held PTF in the Apply job. Do not bypass HOLDERROR conditions. Refer to the *SMP/E Reference* manual for complete instructions on using the BYPASS parameter.

Applying the Maintenance PTFs

Use the sample JCL member **P2APPLY** to apply the Maintenance PTFs. Use the APPLY CHECK option as often as necessary before the actual APPLY process to identify SMP/E processing problems. Before submitting this job, change the BOUNDARY, FORFMID, and BYPASS parameters as needed.

If you are not using MLSS, remove the comment tags (/* and */) from the JES Offset Table function (TSIB311 and/or TSIB312) that you are installing.

If you are using MLSS, you can use JCL member M5PTFAPP to apply PTFs for the TSIB311 or TSIB312 Offset Table function.

Ensure that none of your SVAA SMP/E target load module libraries are in use by SVAA or in the system link list (LNKLSTxx). This job could cause those libraries to be compressed while in use, with unpredictable results.

```
//jobname   JOB    'accounting info'
//*
//          EXEC   smpeproc
//SMPCNTL   DD     *

 SET BOUNDARY ( target ) .     /* Change to your target zone */

 APPLY   FORFMID(
                   SSIB310     /* SVAA Base function         */
                   SSNI210     /* SVAA LLAPI function        */
                   SSJC110     /* SVAA server                */
         /*        TSIB311        JES2 Offset Table function */
         /*        TSIB312        JES3 Offset Table function */
                   SSKP124     /* Common Parser 1.2.4        */
                   ASAR65D     /* SAS/C 6.5.0 Resident       */
                   SSAT65D     /* SAS/C 6.5.0 Transient      */
                   ASAT65D     /* SAS/C 6.5.0 All-Resident   */
              )
         GROUPEXTEND
         BYPASS( bypass options )
         RETRY( YES )
         COMPRESS( ALL )
         .
```

Figure 6-2. *JCL member P2APPLY: Applying the Maintenance PTFs*

Note: Expect return code 04 with Binder message IEW2454W or SMP/E message GIM23903W for link-edit processing to the SIBRTNS and SKPRTNS libraries.

Accepting the Maintenance PTFs

Use the sample JCL member **P3ACCEPT** to accept the Maintenance PTFs. Use the same BYPASS parameters that were used in the APPLY job. Use the ACCEPT CHECK option as often as necessary before the actual ACCEPT process to identify SMP/E processing problems. Before submitting this job, change the BOUNDARY, FORFMID, and BYPASS parameters as needed.

If you are not using MLSS, remove the comment tags (/* and */) from the JES Offset Table function (TSIB311 and/or TSIB312) that you are installing.

```
//jobname   JOB    'accounting info'
//*
//          EXEC   smpeproc
//SMPCNTL   DD     *

 SET BOUNDARY ( dlib ) .       /* Change to your DLIB zone   */

 ACCEPT  FORFMID(
                SSIB310     /* SVAA Base function         */
                SSNI210     /* SVAA LLAPI function        */
                SSJC110     /* SVAA server                */
        /*      TSIB311        JES2 Offset Table function */
        /*      TSIB312        JES3 Offset Table function */
                SSKP124     /* Common Parser 1.2.4        */
                ASAR65D     /* SAS/C 6.5.0 Resident       */
                SSAT65D     /* SAS/C 6.5.0 Transient      */
                ASAT65D     /* SAS/C 6.5.0 All-Resident   */
             )
        GROUPEXTEND
        BYPASS( bypass options )
        RETRY( YES )
        COMPRESS( ALL )
        .
```

Figure 6-3. *JCL member P3ACCEPT: Accepting the Maintenance PTFs*

Appendix A. SVAA Communication Devices

SVAA uses a communication protocol called Extended Control and Monitoring (ECAM) to send messages to, and receive messages from, an SVA subsystem. The SVA subsystem recognizes an ECAM request and processes it in the control unit.

Note: Non-SVA subsystems reject SVA ECAM requests.

Defining ECAM Communications Devices

Most SVAA subcommands require a designated communications (ECAM) device. You designate an ECAM device with the SET ECAMDEVICE subcommand, which can be entered in several ways:

- In the SIBSTK*xx* member of the data set allocated to STKPARMS
- In the PROFSIBA member in SYSPROC
- In SIBBATCH or SIBADMIN under TSO batch
- After entering SIBADMIN without any subcommands
- With the SIBADMIN MACRO command

For SVAA under ISPF, you must have a SET ECAMDEVICE subcommand in either PROFSIBA or SIBSTK*xx*.

Note: Statements in PROFSIBA override those in SIBSTK*xx*, except for SET ECAMDEVICE subcommands, which are cumulative.

If PPRC is in use:

- Do not issue a SET ECAMDEVICE subcommand against a PPRC primary or secondary volume; i.e., do not try to define a PPRC volume as an ECAM device.
- Do not issue a TSO CESTPAIR command (or an ICKDSF PPRCOPY ESTPAIR command) against an ECAM device; i.e., do not attempt to establish a PPRC pair that includes an ECAM device.

Defining a Privileged ECAM Device

To send ECAM messages that modify an SVA subsystem, SVAA requires that you define at least one functional device in the subsystem as a privileged ECAM device.

To define a privileged ECAM device, access the Functional Device Configuration screen at the Local Operator Panel (LOP) or the Detached Operator Panel (DOP) and set the ECAM field to **Y** for at least one functional device. You may want to designate two devices as privileged ECAM devices to ensure one device as a backup. See the *V2Xf Shared Virtual Array: Operation and Recovery* manual for detailed instructions on configuring a functional device.

Message Types

There are three types of ECAM messages:

Category 1-restricted
Messages sent to alter subsystem characteristics or to start a drain. These messages do not have to have the affected device as a destination.

Category 1-unrestricted
Messages sent for Reporter purposes—*not* to alter subsystem characteristics. These messages may have any SVA device as a destination.

Category 2
These messages, which include DDSR requests, must be sent with the affected device as the destination. Any device can be a destination for Category 2 messages.

Table A-1 explains the definitions required for these message types.

Table A-1. *ECAM message types*

Message type	**Device defined as privileged ECAM**	**Device defined for communications**
Category 1-restricted	Required	Required
Category 1-unrestricted	Not required	Required
Category 2	Not required	Not required

Note: A device is defined as a privileged ECAM device by one of:

- Set ECAM to **Y** on the Functional Device Configuration screen at the LOP or DOP.
- DEFINE DEVICE subcommand (from SIB prompt) using the PRIVILEGEDECAM(YES) option
- Privileged ECAM set to **Y** on the SVAA Update Functional Device panel

Note: A device is defined as an SVAA communications device by the SET ECAMDEVICE subcommand.

Appendix B. Contents of SIBSAMP

The following is a list and brief description of sample SVAA jobs, user exits and parameter files you can find in the SIBSAMP data set. The names at the left are also the member names in the SIBSAMP data set.

ALTCHAN	Sample SIBBATCH job that alters the characteristics of several channels.
ALTDEV	Sample SIBBATCH job that alters the characteristics of functional devices.
ALTSUBSY	Sample SIBBATCH job that alters the characteristics of the SVA subsystem.
ATTNDEV	Sample SIBBATCH job that performs the ATTENTION DEVICE function.
DEFDEV	Sample SIBBATCH job that defines 16 functional devices and assigns an FDID range.
DISPDEV	Sample SIBBATCH job that displays all functional devices on all subsystems.
DISPDRIV	Sample SIBBATCH job that displays the status of all physical drives on all subsystems.
DRAINDRV	Sample SIBBATCH job that starts a drain.
FORMARRY	Sample SIBBATCH job that forms an array in the Production partition.
IEAAPFIX	Sample parameter file to authorize the SVAA load libraries.
IECIOSIX	Sample parameter file to set the SVA MIH timeout value.
IEFSSNIX	Sample parameter file to define an SVAA subsystem.
IVP1	Install verification JCL to ensure SVAA is installed properly.
IVP2	Install verification JCL to ensure SVAA can communicate with an installed SVA subsystem.
LISTDEV	Sample SIBBATCH job that lists all functional devices with FDIDs that match a specified FDID pattern.
LNKLSTIX	Sample parmlib member to add the SVAA load module libraries to the system link list.
MPFLSTIX	Sample MPFLSTIXxx member to specify MPF exit processing of selected SVAA console messages.
PROFSIBA	Sample REXX statements for PROFSIBA to customize the SIBADMIN environment.
PROFSIBC	Sample CLIST statements to customize the SIBADMIN environment. Before using this member, rename it PROFSIBA.
PROGIX	Sample parameter file to authorize the SVAA load libraries.
REPORTEV	Sample SIBBATCH job that produces a detailed drain event report.
RPTPERF	Sample job to generate performance reports.
RPTSPACE	Sample job to generate space utilization reports.

SIBALLOC	Sample REXX EXEC to preallocate the SVAA ISPF libraries using TSO ALLOCATE commands and SIBxxxx ddnames.
SIBCEMSR	Sample SAS program (that invokes SAS/GRAPH) to produce the cache effectiveness monthly summary graph.
SIBCMDAX	Sample user exit invoked by SVAA to verify the user's authority to issue a SnapShot subcommand.
SIBCTUM	Sample usermod to use the SIBCTRAN ddname rather than CTRANS to load the SAS/C transient library routines.
SIBDMINC	Sample source code for a Device Inclusion Table. See member SIBETUM for sample usermod for installing a Device Exclusion Table. SIBETUM can also be used to install a Device Inclusion Table.
SIBDPDSR	Sample SAS program (that invokes SAS/GRAPH) to produce the device performance daily summary graph.
SIBDPWSR	Sample SAS program (that invokes SAS/GRAPH) to produce the device performance weekly summary graph.
SIBDSRXX	Sample parameter file for Interval DDSR.
SIBDSR01	Sample parameter file that performs Interval DDSR one time for all devices.
SIBDSR02	Sample parameter file that modifies an Interval DDSR task.
SIBDSR03	Sample parameter file that suspends all Interval DDSR tasks and reactivates them in simulate mode.
SIBDSR04	Sample parameter file that starts or restarts Dynamic DDSR with simulate, trace and debug on.
SIBDSR05	Sample parameter file that suspends the Dynamic DDSR task and reactivates it in simulate mode.
SIBDSR06	Sample parameter file that activates Dynamic DDSR for a given subsystem.
SIBETUM	Sample usermod to implement a Device Exclusion Table.
SIBIOATX	SVAA subsystem I/O authorization user exit.
SIBISPF	Sample TSO logon EXEC to preallocate the SVAA ISPF data sets using TSO ALLOCATE commands.
SIBISPFU	Sample REXX program for converting the SVAA ISPF dialog to uppercase.
SIBLDEF	Sample REXX EXEC to preallocate the SVAA ISPF libraries using LIBDEF and ALTLIB commands.
SIBOFFLD	Sample JCL for a procedure to copy a full logging file to another data set.
SIBOT*	Set of documentation and sample JCL to provide SMP/E dummy usermods that support the SVAA offset tables (SIBRSLV2 and/or SIBRSLV3). Installing these dummy usermods causes SMP/E to issue a regression message for the usermods if any of the required elements are changed by a PTF or APAR.
SIBOTDOC	Documentation for sample usermods for the SVAA offset tables.

SIBOTULD	Sample JCL to unload the macro entries for the operating system target zone.
SIBOTUM1	Sample dummy usermod for the required DFP macros.
SIBOTUM2	Sample dummy usermod for the required JES2 macros.
SIBOTUM3	Sample dummy usermod for the required JES3 macros.
SIBSFCXX	Sample SIBSFC parmlib member to establish Server Framework Component (SFC) environment variables.
SIBSGR01	Sample SAS program (that invokes SAS/GRAPH) to produce the average service time and average transfer size graph from Reporter Data Extract output.
SIBSGR02	Sample SAS program (that invokes SAS/GRAPH) to produce the average service time and cache hit percent graph from Reporter Data Extract output.
SIBSGR03	Sample SAS program (that invokes SAS/GRAPH) to produce the average service time and data throughput graph from Reporter Data Extract output.
SIBSGR04	Sample SAS program (that invokes SAS/GRAPH) to produce the average service time and I/O rate graph from Reporter Data Extract output.
SIBSGR05	Sample SAS program (that invokes SAS/GRAPH) to produce the average service time graph for the twenty-five busiest functional devices from Reporter Data Extract output.
SIBSGR06	Sample SAS program (that invokes SAS/GRAPH) to produce the I/O service time at maximum I/O rate graph from Reporter Data Extract output.
SIBSGR07	Sample SAS program (that invokes SAS/GRAPH) to produce the SVA free space analysis graph from Reporter Data Extract output.
SIBSIMXX	Sample Service Information Message (SIM) processing parameters.
SIBSNDAX	SVAA subsystem send command authorization user exit.
SIBSRPSW	SVAA subsystem file-switch user exit.
SIBSRPXX	Sample Reporter parameters that collect performance tracking data.
SIBSTKXX	Sample configuration parameters for SVAA.
SIBSUIR	Sample SAS program (that invokes SAS/GRAPH) to produce the space-utilization interval graph.
SIBSUMSR	Sample SAS program (that invokes SAS/GRAPH) to produce the space-utilization monthly-summary report.
SIBSYSXX	Sample SVAA component parameters.
SIBTSOP	Sample TSO logon proc to allocate the SVAA libraries.
SIBVOPXX	Sample console command processing parameters.
SIBXGR00	Sample Microsoft Excel macro to produce the recommended graphs from Reporter Data Extract output. This is a *binary* file.
SMFPRMIX	Sample SMF parameters to define SVAA SMF controls.

SUMMHIST	Sample job to summarize collection data.
SVAAJCL	Sample JCL to start the SVAA started task.
SVAASRVR	Sample JCL to run the SIBSFCCM program. This proc is automatically executed by the START SFC command.
SVAASS	Sample JCL to run the SIBMVSS program to execute the ALTER SSNAME command.
UPCASE	ISPF edit macro to convert all lowercase characters to uppercase characters in all members of your hlq.SIBSAMP library.

Appendix C. Storage Requirements

SVAA Subsystem

Before installing SVAA, you should verify that adequate CSA and ECSA storage exists. SVAA CSA and ECSA usage is shown in Table C-1.

Table C-1. *SVAA CSA and ECSA requirements*

	Requirement
CSA storage	32K
ECSA storage	264K + (4K * # SSIDs)

Note: "SSIDs" includes all SSIDs present for all 3990-3 (and above) subsystems attached to the OS/390 system.

Dynamic DDSR

Table C-2. *Dynamic DDSR storage requirement*

	Requirement
Private storage	16K

Note: Dynamic DDSR obtains 16K of private storage from subpool 230 below the 16M line. It is freed when the task ends.

Appendix D. Device Mapping

When the SVAA subsystem is initialized or reinitialized, SVAA performs I/O operations to DASD devices to identify the device characteristics. This process is called device mapping. If SVAA is initialized during the system IPL process (i.e., with IEFSSN entry INIT(Y)) and DASD devices are reserved from another host for an extended period of time, completion of the IPL will generally be delayed by up to 1 MIH interval for each reserved device encountered. Such delays can be avoided if the device mapping process ignores (1) non-SVA devices for which you do not want to collect cache performance information and (2) SVA devices that are always reserved from another host.

The Device Exclusion Table

A device exclusion table (DET) enables you to identify devices to be excluded from device mapping operations. In SIBSAMP, SVAA provides an optional usermod, SIBETUM, for creating such a table. SIBETUM contains sample usermod directives and sample SIBEXDEV statements for generating a DET. Device addresses listed in an exclusion table are treated by SVAA as devices “not defined to the host.”

If it is easier to identify the device addresses that are *not* to be excluded from device mapping operations, you can designate the DET to be an *inclusion* table. SIBSAMP member SIBDMINC contains sample SIBEXDEV statements for creating an inclusion table. For an inclusion table, device addresses that are *not* specified in the table are treated by SVAA as devices “not defined to the host.”

Note: Addresses for Power PPRC Bridge devices and PAV Alias devices can be specified in exclusion or inclusion tables, but are ignored since device mapping never maps Bridge devices or PAV Alias devices.

An exclusion table should *not* contain addresses for SVA devices attached to the host on which the exclusion table resides. Conversely, if you use an *inclusion* table, it should contain the device addresses for *all* SVA devices attached to the host. Not following this practice can increase the net capacity load (NCL), reduce SVAA functionality, and possibly reduce SVA performance as a result of the increase in NCL.

Non-SVA device addresses for which SVAA reporting is desired must be specified in the same manner as SVA devices.

To Implement a Device Exclusion Table:

1. Alter the sample usermod SIBETUM by coding the SIBEXDEV statements to identify the device addresses to be excluded from device mapping operations.
2. Perform the SMP/E installation of the SIBETUM usermod.
3. Copy the resulting SIBDMEXC module from your SMP/E target SIBLINK data set into your system linklist data set.
4. Refresh LLA (F LLA,REFRESH)
5. Use SIBMVSS to initialize/reinitialize the device mapping queue (DMQ) with the following SIBMVSS control statement:

```
ALTER SSNAME(NAME(SVAA) REINIT RIM(DMQ))
```

Note: If INIT(Y) is specified in IEFSSN (as described in “Step 8d: Customizing the IEFSSNxx Member” on page 3-4), an IPL can be used instead of steps 4 and 5.

Syntax for SIBEXDEV statements

SIBEXDEV

```
►►──label──SIBEXDEV──UNIT=──┬─(addr─┬────────┬─)─┬──┬──────────────────┬──────────────►
                            │       └─,──cnt─┘   │  └─,LIST=─┬─YES─┬──┘
                            └─addr───────────────┘           └─NO──┘

►──┬──────────────────────────┬──┬──────────────────────┬────────────────────────────►◄
   │          ┌─EXCLUDE─┐     │  │         ┌─ENTRY─┐    │
   └─,TABLE=──┴─INCLUDE─┴─────┘  └─,TYPE=──┴─LAST──┴────┘
```

Parameters

label

Optional

This parameter specifies the assembler label on the SIBEXDEV invocation statement.

Values: 1 to 8 alphanumeric character string, starting in column 1.

Default value: A null string.

UNIT

Required

This parameter specifies the unit address.

Values: You can specify a range of addresses by enclosing the starting unit address and number of consecutive addresses within parenthesis. The sum of *addr* and *cnt* cannot cross a channel boundary.

addr (hexadecimal) specifies a unit address (required).

cnt (decimal) specifies a number of consecutive unit addresses (optional).

Default value: *cnt* defaults to 1.

LIST

Optional

This parameter specifies whether the full macro expansion is to be listed.

Values: **YES** specifies the full macro expansion is to be listed.

NO specifies the full macro expansion is not to be listed.

If the value is omitted, the current assembler PRINT settings are to be used.

Default value: omitted (use current PRINT settings)

TABLE

Optional

This parameter specifies whether the table is an inclusion table or an exclusion table. The TABLE parameter can be used only in the first SIBEXDEV statement.

Values: **INCLUDE** specifies an inclusion table. Only device addresses included in the table are processed by SVAA.

EXCLUDE specifies an exclusion table. Device addresses in the table are *not* processed by SVAA.

omitted (no value)—treated the same as **EXCLUDE**.

Default value: **EXCLUDE**

TYPE

Optional

This parameter specifies the type of macro expansion.

Values: **ENTRY** generates DET entries.

LAST generates the final DET entry (or entries). Exactly one SIBEXDEV statement in the DET must specify **TYPE=LAST**.

Default value: **ENTRY**

Sample SIBEXDEV Statements

Figure D-1 shows samples of SIBEXDEV statements for a device exclusion table.

```
*  Identify device D43 as a device to be excluded.  Specifying
*  TYPE=ENTRY is optional; it's the default.  TABLE=EXCLUDE
*  identifies this as an exclusion table.  If only one SIBEXDEV
*  statement is required to identify all the devices to be
*  excluded, that single statement must specify TYPE=LAST.
*
     SIBEXDEV TYPE=ENTRY,UNIT=D43,TABLE=EXCLUDE
*
*  Identify devices D44, D45, D46, D64, D65, D67, and D68
*  as devices to be excluded using the range format of the
*  SIBEXDEV macro.  Also identify D50, D52, D54, and D58
*  as devices to be excluded using the single unit format
*  of the SIBEXDEV macro.
*
      SIBEXDEV UNIT=(D44,3)
      SIBEXDEV UNIT=(D50)
      SIBEXDEV UNIT=(D52)
      SIBEXDEV UNIT=D54
      SIBEXDEV UNIT=D58
      SIBEXDEV UNIT=(D64,2)
      SIBEXDEV UNIT=(D67,2)
*
*  Idendify device "D70" as a device to be excluded.  Specifying
*  TYPE=LAST indicates this as the last device entry in the table.
*  (The table must contain exactly one TYPE=LAST statement.)
*  Note that no assembler END statement is required.
*
     SIBEXDEV TYPE=LAST,UNIT=D70
```

Figure D-1. *Sample SIBEXDEV statements for a device exclusion table*

Appendix E. SVAA User Exits

If you activate SVAA user exits, they must reside in a library with the same requirements as the *hlq*.SIBLOAD load module library. (See "Step 11b: Verifying SIBLOAD Library Requirements" on page 3-18 for the requirements for the *hlq*.SIBLOAD library.)

Common Areas Available to User Exits

On entry to all user exits, register 1 points to a fullword that contains the address of a common user exit parameter list, mapped by the SIBUXP DSECT. Some of the most useful fields SIBUXP contains are:

- The subpool (SIBUXP_SUBPOOL) and key (SIBUXP_KEY) the exit should use to GETMAIN additional storage if the user exit storage block is not large enough.
- The address (SIBUXP_GSTRGP) and length (SIBUXP_GSTRGLEN) of a storage block (mapped by the SIBUXSQE DSECT) available to all SVAA user exits invoked by this task.
- The address (SIBUXP_ESTRGP) and length (SIBUXP_ESTRGLEN) of a storage block available to this user exit. All invocations of the same exit *in this task* use the same storage block.
- A register save area for use by the exit (SIBUXP_REGSAVE).

Table E-1 shows the record layout for the common header fields for each user exit (SIBUXP).

Table E-1 (Page 1 of 2). *Layout of common header (SIBUXP) parameter list record*

Type	Dec Offset	Hex Offset	Length	Value	Name	Description
A	0	0	1		SIBUXP_VERSION	Version code of this structure
				1	SIBUXP_VERS	Current version
A	1	1	1		SIBUXP_RSVD1	Reserved
A	2	2	2		SIBUXP_LENGTH	Length of the common SIBUXP header area
A	4	4	4		SIBUXP_ITYPE	Type of invocation
				1	SIBUXP_INVCINIT	Initial invocation
				2	SIBUXP_INVCNORM	Normal invocation
				3	SIBUXP_INVCTERM	Termination
A	8	8	1		SIBUXP_SUBPOOL	Exit should use this subpool if additional storage is required
A	9	9	1		SIBUXP_KEY	Key in which exit is invoked
X	10	A	6		SIBUXP_RSVD2	Reserved
X	16	10	2		SIBUXP_GSTRGLEN	Length of user exit global storage block
A	18	12	2		SIBUXP_ESTRGLEN	Length of storage block for this user exit
A	20	14	4		SIBUXP_GSTRGP	Pointer to user exit global storage element (SIBUXSQE)
A	24	18	4		SIBUXP_ESTRGP	Pointer to storage element for the user exit (SIBUXSQE)
X	28	1C	72		SIBUXP_REGSAVE	Save area for use by the user exit
A	100	64	4		SIBUXP_PRMLEN	Length of the PLIST supplied to the user exit

Table E-1 (Page 2 of 2). *Layout of common header (SIBUXP) parameter list record*

Type	Dec Offset	Hex Offset	Length	Value	Name	Description
D	104	68	0		SIBUXP_PRMP	Start of individual exit's PLIST area

Note: Types are defined as:

A – address
X – hexadecimal character
D – double word
C – character

Table E-2 shows the record layout for the common storage area fields for each user exit (SIBUXSQE).

Table E-2. *Layout of common storage (SIBUXSQE) element list record*

Type	Dec Offset	Hex Offset	Length	Value	Name	Description
C	0	0	8		SIBUXSQE_EPNAME	Exit name or 'GLOBAL' for the storage area presented to all user exits executing within this task
A	8	8	1		SIBUXSQE_SUBPOOL	Subpool of this storage element
A	9	9	1		SIBUXSQE_STRGKEY	Key of this storage element
A	10	A	2		SIBUXSQE_STRGLEN	Length of the storage block in this SIBUXSQE
				256	SIBUXSQE_EBLKLEN	Length of individual exit's UXSQE storage block
				256	SIBUXSQE_GBLKLEN	Length of global UXSQE storage block
X	12	C	4		SIBUXSQE_RSVD1	Reserved
D	16	10	0		SIBUXSQE_STRGBLK	Start of storage block for the exit

Note: Types are defined as:

A – address
X – hexadecimal character
D – double word
C – character

SVAA SIBIOATX User Exit

The SIBIOATX user exit is invoked by SVAA to verify a user's authority to send a
| particular ECAM message to an SVA subsystem or to issue a PPRC subcommand. The SIBSAMP data set includes a sample SIBIOATX exit, with its default actions. You can modify this exit's actions to meet your needs.

The SIBIOATX exit indicates the action to be taken in SIBIOAXP_EXITRET. The exit can unconditionally allow the request, unconditionally deny the request, or ask SVAA to invoke RACROUTE to perform security checking. If the latter is chosen, the exit can modify the access type (SIBIOAXP_ACCSTYPE), the class name (SIBIOAXP_CLASSNAM), and/or the resource name (SIBIOAXP_RSRCENAM) to be used.

SVAA Actions if SIBIOATX is not Implemented

If you choose not to use the SIBIOATX exit, SVAA invokes Security Access Facility (SAF) services to check the user's authority to use one of the following sets of parameters to send messages to an SVA subsystem:

- If the message is restricted:
  - Security class name = **FACILITY**
  - Security resource name = **STGADMIN.IDC.SETCACHE**
  - Access type = **UPDATE**
- If the message is not restricted:
  - Security class name = **FACILITY**
  - Security resource name = **STGADMIN.IDC.LISTDATA**
  - Access type = **READ**

Note: See Appendix A, “SVAA Communication Devices” for more information about restricted and unrestricted SVAA messages.

The message is sent to the SVA subsystem if the user is authorized to send messages. If the user is not authorized, the request is denied. If the security system cannot determine whether the user is authorized, the request is allowed if the security access type is read; otherwise it is denied.

| If you choose not to use the SIBIOATX exit to filter PPRC subcommands, or choose (via the exit) to defer the subcommand to RACROUTE, SVAA invokes the Security Access Facility (SAF).

| Please note that, by default, all PPRC subcommands are allowed by SVAA when the security system cannot determine whether the user is authorized (that is, the security class and resource names are not defined as detailed below).

- SVAA SAF services first checks the user's authority to issue any PPRCOPY subcommand using the following profile:
  - Security class name = **FACILITY**
  - Security resource name = **STGADMIN.ANT.PPRC.COMMANDS**
  - Access type = **READ**
- If the PPRC QUERY subcommand is issued but was not permitted with the above profile, SVAA SAF services attempts a second check for authorization to issue the QUERY subcommand using the following profile:
  - Security class name = **FACILITY**
  - Security resource name = **STGADMIN.ANT.PPRC.CQUERY**
  - Access type = **READ**

Note: If you change the security access rules, you must restart the TSO session or end the current SIBADMIN session and restart it for SVAA to recognize the change.

Return Codes

Return codes for the SIBIOATX user exit are listed in Table E-3.

Table E-3. *SIBIOATX return codes*

Code	Description
0	Invoke RACROUTE to validate request
1	Request unconditionally allowed
2	Request unconditionally denied

Entry Specifications

Upon entry to SIBIOATX, the register contents are as listed in Table E-4.

Table E-4. *SIBIOATX entry specifications*

Register	Contents
0	Undefined
1	Pointer to a fullword containing the address of the SIBUXP common parameter list header
2 - 12	Undefined
13	OS save area
14	Return address
15	Entry point address for SIBIOATX

Return Specifications

Return specifications for the SIBIOATX user exit are listed in Table E-5.

Table E-5. *SIBIOATX return specifications*

Register	Contents
0 - 12	Undefined
13	OS save area
14	Return address
15	Return code (hexadecimal)

Program Attributes

The execution attributes of the SIBIOATX user exit are:

- Reentrant
- Refreshable
- Problem State
- Problem Program PSW key
- AMODE=31
- Enabled Unlocked Task Mode
- Non-Cross Memory Mode

Parameter List

Table E-6 shows the layout for SIBIOAXP, the parameter list for the SIBIOATX user exit.

Table E-6 (Page 1 of 2). *Layout of SIBIOAXP parameter list record*

Type	Dec Offset	Hex Offset	Length	Value	Name	Description
A	0	0	4		SIBIOAXP_MSGCLASS	SVAA message class
				1	SIBIOAXP_CONFIG	Configuration
				2	SIBIOAXP_CFGRPT	Configuration reporting
				3	SIBIOAXP_PERFRPT	Performance reporting

Table E-6 (Page 2 of 2). *Layout of SIBIOAXP parameter list record*

Type	Dec Offset	Hex Offset	Length	Value	Name	Description
				4	SIBIOAXP_SPACERPT	Space utilization reporting
				5	SIBIOAXP_MAT	Media Acceptance Test
				7	SIBIOAXP_GCID	Guaranteed cache by ID
				9	SIBIOAXP_AUDIT	Audit trail
				10	SIBIOAXP_ADMIN	Administrative
				11	SIBIOAXP_RSSD	3990 Read Subsystem Data
				12	SIBIOAXP_SNAPVOL	Snap volume request
				13	SIBIOAXP_SNAPDSN	Snap data set request
				14	SIBIOAXP_PPRCCMD	PPRC command
A	4	4	4		SIBIOAXP_ACCSTYPE	Security access type
				2	SIBIOAXP_ACCSREAD	Read access
				4	SIBIOAXP_ACCSUPDT	Update access
				8	SIBIOAXP_ACCSCNTL	Control access
				128	SIBIOAXP_ACCSALTR	Alter access
A	8	8	4		SIBIOAXP_EXITRET	Exit return
				0	SIBIOAXP_ROUTESAF	Invoke RACROUTE to validate request
				1	SIBIOAXP_ALLOWUNC	Unconditionally allow request
				2	SIBIOAXP_DENYUNC	Unconditionally deny request
C	12	C	8		SIBIOAXP_SUBSYSNM	SVA subsystem name
C	20	14	8		SIBIOAXP_CLASSNAM	Resource class name Default=FACILITY
C	28	1C	100		SIBIOAXP_RSRCENAM	Resource name. The number of characters cannot exceed the maximum for the class. May be blank padded or null terminated. The default is: STGADMIN.IDC.LISTDATA or STGADMIN.IDC.SETCACHE or STGADMIN.ANT.PPRC.COMMANDS
	128	80	8		SIBIOAXP_USERID	Foreground or background user ID modifiable by the exit. The field is left-justified and padded to the right with blanks.
	136	88			SIBIOAXP_SIZE	Total structure size.

Note: Types are defined as:

A – address
X – hexadecimal character
D – double word
C – character

SVAA SIBSNDAX User Exit

The SIBSNDAX user exit can be invoked by SVAA to verify the user's authority to send a command to a subtask in the SVAA address space (for example, SIBSRP, SIBVOP, SIBDSR). SVAA includes a sample SIBSNDAX exit and its default actions in SIBSAMP. You can modify this exit's actions to meet your needs.

SIBSNDAX indicates the action to be taken in SIBSNDXP_EXITRET. The exit can unconditionally allow the command to be sent, unconditionally reject the command, or ask SVAA to invoke RACROUTE to perform security checking. If the latter is chosen, the exit can modify the access type (SIBSNDXP_ACCSTYPE), the class name (SIBSNDXP_CLASSNAM), and/or the resource name (SIBSNDXP_RSRCENAM) to be used.

SVAA Actions if SIBSNDAX is not Implemented

If you choose not to use the SIBSNDAX exit, SVAA invokes Security Access Facility (SAF) services to check the user's authority to use one of the following sets of parameters to send messages to an SVA subsystem:

- If the target task is SIBSIM or SIBSRP, and the command is not **STOP**:
  - Security class name = **FACILITY**
  - Security resource name = **STGADMIN.IDC.LISTDATA**
  - Access type = **READ**
- If the task is SIBVOP or SIBDSR, or if the command is **STOP**:
  - Security class name = **FACILITY**
  - Security resource name = **STGADMIN.IDC.SETCACHE**
  - Access type = **UPDATE**

The message is sent to the SVAA address space if the user is authorized to send messages. If the user is not authorized, the request is denied. If the security system cannot determine whether the user is authorized, the request is allowed if the access type is read; otherwise it is denied.

Return Codes

Return codes for the SIBSNDAX user exit are listed in Table E-7.

Table E-7. *SIBSNDAX return codes*

Code	Description
0	Invoke RACROUTE to validate request
1	Request unconditionally allowed
2	Request unconditionally denied

Entry Specifications

Upon entry to SIBSNDAX, the register contents are as listed in Table E-8.

Table E-8. *SIBSNDAX entry specifications*

Register	Contents
0	Undefined
1	Pointer to a fullword containing the address of the SIBUXP common parameter list header
2 - 12	Undefined
13	OS save area
14	Return address
15	Entry point address for SIBSNDAX

Return Specifications

Return specifications for the SIBSNDAX user exit are listed in Table E-9.

Table E-9. *SIBSNDAX return specifications*

Register	Contents
0 - 12	Undetermined
13	OS save area
14	Return address
15	Exit return value

Program Attributes

The execution attributes of SIBSNDAX user exit are:

- Reentrant
- Refreshable
- Problem State
- Problem Program PSW key
- AMODE=31
- Enabled Unlocked Task mode
- Non-Cross Memory Mode

Parameter List

Table E-10 shows the layout for SIBSNDXP, the parameter list for the SIBSNDAX user exit.

Table E-10 (Page 1 of 2). *Layout of SIBSNDXP parameter list record*

Type	Dec Offset	Hex Offset	Length	Value	Name	Description
A	0	0	4		SIBSNDXP_TASK	Task type
				1	SIBSNDXP_SRP	Reporter
				2	SIBSNDXP_VOP	Virtual Operator Panel
				3	SIBSNDXP_DDSR	Deleted Data Space Release
				4	SIBSNDXP_SIM	Service Information Messages
A	4	4	4		SIBSNDXP_ACCSTYPE	Security access type
				2	SIBSNDXP_ACCSREAD	Read access
				4	SIBSNDXP_ACCSUPDT	Update access
				8	SIBSNDXP_ACCSCNTL	Control access
				128	SIBSNDXP_ACCSALTR	Alter access
A	8	8	4		SIBSNDX_EXITRET	Exit return
				0	SIBSNDXP_ROUTESAF	Invoke RACROUTE to validate request
				1	SIBSNDXP_ALLOWUNC	Unconditionally allow request
				2	SIBSNDXP_DENYUNC	Unconditionally deny request
C	12	C	4		SIBSNDXP_SSN	MVS subsystem name to which the command is to be routed
C	16	10	8		SIBSNDXP_CLASSNAM	Name of the resource class Default=FACILITY

Table E-10 (Page 2 of 2). *Layout of SIBSNDXP parameter list record*

Type	Dec Offset	Hex Offset	Length	Value	Name	Description
C	24	18	100		SIBSNDXP_RSRCENAM	Name of the resource. The number of characters cannot exceed the maximum for the class and can be blank padded or null terminated. The default is: STGADMIN.IDC.LISTDATA or STGADMIN.IDC.SETCACHE
A	124	7C	2		SIBSNDXP_RSVD1	Reserved
H	126	7E	2		SIBSNDXP_CMDLEN	Length of command string
	128	80			SIBSNDXP_SIZE	Length of fixed portion of structure
C	128	80	variable		SIBSNDXP_COMMAND	Start of variable length command string

Note: Types are defined as:

A – address
X – hexadecimal character
D – double word
C – character

SVAA SIBSRPSW User Exit

The SIBSRPSW user exit can be invoked by the SVAA Reporter data collection task before closing or switching an output logging file. A sample user exit (called SIBSRPSW) is provided in the SIBSAMP library. You must modify this exit's actions to meet your needs if you want to use the main and alternate logging files instead of SMF for your collected data.

The exit is invoked when any of the following circumstances occur:

- The output file is full
- An I/O error has occurred on the output file
- An access error (an open, close, or allocation error) has occurred on the output file
- An SVAA “SET COLLECTION(COLLID(...) SWITCH)” command is issued

SIBSRPSW indicates the action to be taken in SIBFSWXP_EXITRET. The return code chosen must be appropriate for the exit reason (see description of SIBFSWXP_EXITRET). If it is not appropriate, the default action is taken. The action taken also depends on whether there is an alternate file.

Customizing the SIBSRPSW Exit

To customize the SIBSRPSW exit to handle full logging files, make sure that the exit:

1. Sets up symbolic parameters to send to the SIBOFFLD procedure.
2. Requests Reporter to send a START command to OS/390 to invoke the SIBOFFLD procedure with the symbolic parameters.

You must also make sure that the SIBOFFLD procedure is authorized (through RACF or otherwise) to update the target offload file you specify in the SIBOFFLD procedure JCL.

SIBOFFLD Procedure JCL

You should install the procedure in Figure E-1 in a standard proclib and customize the variables (for example, names and space allocation) for your environment.

SVAA provides a sample SIBOFFLD procedure in the SIBSAMP data set.

```
//SIBOFFLD PROC FILENAM=,       NAME OF FILE TO OFFLOAD
//         PRIME=20,            OFFLOADED FILE PRIMARY CYLS
//         SEC=2                OFFLOADED FILE SECONDARY CYLS
//*
//* COPY THE REPORTER DATA COLLECTION FILE
//*
//STEP1  EXEC  PGM=IEBGENER
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  DSN=&FILENAM,DISP=SHR
//SYSUT2   DD  DSN=hlq.XSARPT.DATA(+1),DISP=(,CATLG,DELETE),
//         UNIT=SYSDA,SPACE=(CYL,(&PRIME,&SEC),RLSE)
//SYSIN    DD  DUMMY
//*
//* MAKE THE REPORTER OUTPUT FILE EMPTY
//*
//STEP2  EXEC  PGM=IEBGENER
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  DUMMY,DCB=&FILENAM
//SYSUT2   DD  DSN=&FILENAM,DISP=SHR
//SYSIN    DD  DUMMY
//         PEND
```

Figure E-1. *SIBOFFLD procedure JCL*

Notes:

1. If the SYSUT2 statement names a GDG, you must create the GDG. If the sample DSCB does not contain DCB parameters, you must add them to the JCL.
2. To offload to tape rather than to disk, remove the disk allocation parameters from the SIBOFFLD procedure and the SIBSRPSW exit, and replace the disk information in the SYSUT2 statement with unit, label, and retention period information for copying to tape.

SVAA Actions if SIBSRPSW is not Implemented

If you choose not to use the SIBSRPSW exit, SVAA takes one of the following actions:

- If a SWITCH command was issued:
  - If there is an alternate logging file, the current file is closed and the alternate file is opened.
  - If there is no alternate logging file, the command is ignored (a message shows a predetermined default return code).
- If the file was full or there was an I/O error and if an alternate file has been specified, output is switched to the alternate file.

- If there was an allocation open or close error, or if no alternate file has been specified, the output file is closed and all collection runs using the file are terminated.

Return Codes

Valid return codes from the SIBSRPSW user exit are listed in Table E-11.

Table E-11. *SIBSRPSW return codes*

Code	Description
0	Close the current file and switch to alternate file; redrive the I/O
1	Close the current file and switch to alternate file; flush the I/O
3	Close and reopen the current file, reposition to the start of the current file, and redrive the I/O
4	Close and reopen the current file, reposition to the start of the current file, and flush the I/O
5	Close the current file and flush all I/O until the file becomes empty
6	Switch to alternate file and open the file
7	Ignore the user's command to switch to alternate file
8	Close the current file and stop all collection runs using the file

These codes are returned for the conditions listed in Table E-12.

Table E-12. *SIBSRPSW conditions*

Condition	Possible Return Codes
I/O error conditions	0, 1, 8
Full file conditions	0, 1, 3, 4, 5, 8
File access (open/close allocation) errors	6, 8
Switch commands	6, 7, 8

Entry Specifications

Upon entry to SIBSRPSW, the register contents are as listed in Table E-13.

Table E-13. *SIBSRPSW entry specifications*

Register	Contents
0	Undefined
1	Pointer to a fullword containing the address of the SIBUXP common header
2 - 12	Undefined
13	OS save area
14	Return address
15	Entry point address for SIBSRPSW

Return Specifications

Return specifications for the SIBSRPSW user exit are listed in Table E-14.

Table E-14. *SIBSRPSW return specifications*

Register	Contents
0 - 12	Undefined
13	OS save area
14	Return address
15	Exit return value

Program Attributes

The execution attributes of the SIBSRPSW user exit are:

- Reentrant
- Refreshable
- Problem State
- Problem Program PSW key
- AMODE=31
- Enabled Unlocked Task mode
- Non-Cross Memory Mode

Parameter List

Table E-15 shows the record layout for SIBFSWXP, the parameter list for the SIBSRPSW user exit.

Table E-15 (Page 1 of 2). *Layout of SIBFSWXP parameter list record*

Type	Dec Offset	Hex Offset	Length	Value	Name	Description
A	0	0	4		SIBFSWXP_FILEATTR	File attributes
				1	SIBFSWXP_MAINLOG	Main logging file
				2	SIBFSWXP_ALTEMPTY	Empty alternate file exists
A	4	4	4		SIBFSWXP_EXITREAS	Reason for switch exit call
				1	SIBFSWXP_FILEFULL	Output file is full
				2	SIBFSWXP_SWTCHCMD	Switch command received
				3	SIBFSWXP_IOERR	File I/O error
				4	SIBFSWXP_ACCERR	File Open/Close/Allocate error
A	8	8	4		SIBFSWXP_EXITRET	Exit return code
				0	SIBFSWXP_SWCHRDRV	Switch to alternate and redrive the I/O
				1	SIBFSWXP_SWCHFLSH	Switch to alternate but flush the I/O
				2	SIBFSWXP_NOSWRDRV	Redrive the I/O to the end of the current file (CMS only)
				3	SIBFSWXP_RPOSRDRV	Reposition to the start of the current file and redrive the I/O
				4	SIBFSWXP_RPOSFLSH	Reposition to the start of the current file and flush this I/O only
				5	SIBFSWXP_EMTYFLSH	Close the file and flush all I/O until the file becomes empty
				6	SIBFSWXP_OPENALT	Switch to the alternate file and open it

Table E-15 (Page 2 of 2). *Layout of SIBFSWXP parameter list record*

Type	Dec Offset	Hex Offset	Length	Value	Name	Description
				7	SIBFSWXP_IGNORCMD	Ignore the SWITCH command
				8	SIBFSWXP_CLOSFILE	Close the file and stop all collection runs using the file
A	12	C	4		SIBFXWXP_PROCACTN	Determines whether SVAA starts a JCL procedure
				0	SIBFSWXP_NOPROC	Don't start a procedure
				1	SIBFSWXP_STRTPROC	Start a file-offload procedure
C	16	10	44		SIBFSWXP_DSNAME	Data set name of current output file
C	60	3C	44		SIBFSWXP_ALTDSN	Alternate file name or blanks
F	104	68	4		SIBFSWXP_OPTSLEN	Length of procedure options
C	108	6C	8		SIBFSWXP_PROCNAME	Name of procedure to be started; ignored unless PROCACTN is set to SIBFSWXP_STARTPROC
C	116	74	115		SIBFSWXP_PROCOPTS	Additional parameters on start command (symbolic substitution; for example, NAME=)
X	231	E7	1		SIBFSWXP_RSVD1	Reserved
D	232	E8			SIBFSWXP_SIZE	Total structure length

Note: Types are defined as:

A – address
X – hexadecimal character
D – double word
C – character

Appendix F. SVAA Server Cache

The SVAA Server maintains a cache of information about the overall SVA subsystem, I/O interfaces, physical devices, and functional devices. The server keeps a separate cache for each SVA subsystem to which it has access.

Note: The SVAA server cache should not be confused with SVA subsystem cache. The latter is used in the transfer of user data to and from the SVA subsystem.

The SVAA server cache includes the following elements:

Cache Element	Examples of Information Included
Array status	Array drain status
Functional device performance	Device capacity, number of requests, amount of data transferred, time available
Functional device properties	Device characteristics which vary depending on the type of device. May include device DTL (domain, target, LUN), device type, read/write enabled status, ECAM privileged status
Functional device space	Device tracks available and used
I/O performance	Speed, number of IOs, amount of time busy
I/O interface properties	Name, type, status
Physical device performance	Busy time, number of bytes transferred for read/write operations
Physical device status	Location, status, array assignment, capacity
Subsystem performance	Free space, NCL, back-end capacity, SVA cache sizes, number of ECAM messages processed and bypassed
Subsystem properties	Installed features, general information such as number of devices, number of I/O interfaces supported

Note: PPRC properties are not included in SVAA server cache.

Why SVAA Uses Cache

The SVAA server cache helps to optimize overall system performance. By keeping some information in its cache, the SVAA server reduces the number of times it must access the SVA subsystem to obtain information about subsystem components. This improves the request response time, particularly for point-in-time reports.

How Cache is Updated

The SVAA server cache can be updated in the following ways:

- As part of a periodic cache refresh.

  By default, the SVAA server automatically refreshes all its cache elements every 900 seconds (fifteen minutes). You can use the SET CACHEREFRESHRATE subcommand to override this rate for some or all cache elements to better meet the needs of your installation. See “Changing Cache Refresh Rates” on page F-3 for details.

- Through an SVA subsystem update initiated by the SVAA Server.

  Anytime an SVAA server causes a change to an SVA subsystem component, that server's cache is immediately updated with the new information. For example, if a user on Server1 deletes a functional device, Server1's cache is immediately updated to remove the device. The cache of any other SVAA servers is not updated, however, until a periodic cache refresh occurs. This is also true if the device was deleted without using the SVAA Server, such as through SIBADMIN or SIBBATCH. See “Cache Update Issues” on page F-3 for additional information.

When a cache refresh rate is set to DEFAULT or a specific number of seconds, a thread is created to collect data at the requested interval. There are ten cache elements for each SVA subsystem. If each cache element has an associated interval, ten threads are created at SVAA Server initialization. An additional ten threads are created for each SVA subsystem being monitored.

At its specified interval, the thread collects data and stores it in the SVAA server's cache.

The SET SERVERIDLETIME subcommand can be used to control whether the SVAA server cache should continue to collect SVA subsystem data during periods when the SVAA server has no requests to process. Using SET SERVERIDLETIME can reduce the amount of system resources required to keep the data in cache current.

Notes:

1. Depending on the number of SVAA servers and the number of SVA subsystems monitored, the MAXTHREADS limit in the **BPXPRMxx** member of SYS1.PARMLIB may be reached resulting in an "Out of Memory" condition. The following message will appear in the SVAA Server log:

   ```
   SIB7005E Exception: java.lang.OutOfMemoryError
   ```

   In this case, the MAXTHREADS limit should be increased.

   You can determine the current setting of MAXTHREADS by issuing the OS/390 operator command:

   ```
   D OMVS,O
   ```

   If the cache refresh interval is OFF, no thread is created, and data is not collected at intervals or stored in the server's cache. The use of server cache is disabled, and SVAA will always request information directly from the SVA subsystem.

2. Turning off cache may significantly affect request response times; therefore, it should be used with care.

Cache Update Issues

An SVAA server's cache is not immediately updated when an SVA subsystem update is initiated by another server, the LOP (Local Operator Panel), Detached Operator Panel, SIBADMIN, or SIBBATCH. Therefore, the cache of all servers may not always reflect the exact status of the SVA subsystem at any given point in time.

In situations where it is critical that an SVAA server's cache be current, you can turn caching off for the elements in question. See “Turning Off Cache” for details.

Changing Cache Refresh Rates

By default, all cache elements for all SVA subsystems are refreshed every 900 seconds (fifteen minutes). The SET CACHEREFRESHRATE subcommand allows you to change specified refresh rates to any frequency ranging from one minute to 24 hours.

You can set different cache rates for different elements. You can also turn some or all caching off. See “Turning Off Cache” for details.

Notes:

1. Before any changes will take effect, the SVAA Server must be shut down and restarted.
2. Setting the refresh rate to a value lower than the default may impact CPU, I/O, and memory resources. Be sure to monitor your resource consumption after switching to a lower interval.

Turning Off Cache

The **OFF** keyword of the SET CACHEREFRESHRATE subcommand allows you to turn off SVAA server cache for specified cache elements. If a cache element is turned off, the SVAA Server will always request information directly from the SVA subsystem.

Note: Turning off cache may significantly affect request response times; therefore, it should be done with care.

You can turn caching on again by using the SET CACHEREFRESHRATE subcommand to re-assign a refresh rate.

Note: Before any changes will take effect, the SVAA Server must be shut down and restarted.

Displaying Cache Refresh Rates

The QUERY CACHEREFRESHRATE subcommand displays the current cache refresh rate settings. You can choose which elements you want to display.

Glossary

This glossary is included in each book in the Shared Virtual Array Administrator library. All of the terms are associated with SVAA, but not all are used in this specific document.

A

Alias. A pseudo-device used by the operating system to support an additional I/O path to a Base device. Each Alias device supports one additional I/O to a Base. See also: Base and Parallel Access Volume.

array. A group of storage devices that are used collectively to achieve data redundancy and/or improved performance. In the SVA, an array consists of either 7 or 15 drive modules. See also: dual-redundancy array.

array cylinder. The collection of all physical cylinders in a dual-redundancy array that have the same physical cylinder address (CC). The SVA allocates back-end space in units of array cylinders. There are two types of array cylinders: free and allocated.

array device. The disk devices that are logically grouped together when a FORM ARRAY command is issued at the local operator panel or from SVAA.

array track. The collection of all physical tracks in a dual-redundancy array that have the same physical track address (CC, HH).

B

back-end storage. The data storage portion of a storage subsystem. In the SVA, the disk arrays.

Base. A real device that supports additional I/O paths to itself in the form of Alias devices. Each Alias device supports one additional I/O to a Base. Multiple Alias devices can be associated with a single Base. See also: Alias and Parallel Access Volume.

base functional device ID (BFDID). The functional device identifier that maps to or from the lowest (base) interface address on a given channel.

C

cache. Solid state, random access memory that is located in a controller. The cache retains frequently used data for faster access by the channel. In the SVA, all data access is through cache.

cache fast write (CFW). A form of fast write in which data is written directly to cache storage without using nonvolatile storage and is available for later destaging.

Capacity on Demand (COD). A feature that allows SVA customers to exceed their purchased physical capacity (PCAP) limit by up to 860GB of additional temporary effective capacity to prevent writes from being blocked during temporary and sometimes critical usage periods.

channel end. The indication from the channel that it has completed an operation.

channel interface. The Disk Array Controller circuitry that attaches to the host channels.

cluster. See storage cluster.

collected free space %. The percentage of array cylinders that are free array cylinders (collected and completely free of user data).

compaction. The SVA process that eliminates inter-record gaps normally associated with CKD DASD. Compaction reduces the amount of wasted disk array space, thus reducing the net capacity load on the subsystem.

compression. The SVA process that reduces the size of data records by translating them to a different encoding scheme that requires fewer bytes of real storage.

controller. See Disk Array Controller.

count-key-data (CKD). A recording format that writes variable-length records. Each record consists of 1) a count field, which specifies the length of the (optional) key field and data field of the record, 2) the (optional) key field, and 3) a data field. The first record on each track contains a fourth field, home address.

current data. User data, stored in a disk array, that has valid pointers from internal SVA mapping tables.

D

DASD fast write (DFW). A form of fast write to cache in which data is written concurrently to cache and nonvolatile storage (NVS) and is subsequently scheduled for destaging to the disk arrays. Both copies are retained in the SVA Disk Array Controller until the data is completely written to the disk arrays.

Data Bridge. A pair of devices used by Power PPRC to transmit all tracks on all primary devices from the primary subsystem to the secondary subsystem. These devices are not used to store customer data.

dedicated connection. In an Enterprise Systems Connection Director (ESCD), a connection between two ports that is not affected by information contained in link frames. This connection restricts these ports from communicating with any other port. The two ports have a dedicated connection that appears as one continuous link.

Deleted Data Space Release (DDSR). An SVAA facility for OS/390 that informs the Disk Array Controller when functional volume data sets are deleted. The physical disk array space occupied by the deleted data can immediately become free space, thereby reducing the net capacity load on the subsystem.

destage. The nonsynchronous write of new or updated data from the cache storage or nonvolatile storage to the Disk Array Units.

device. See (1) drive module and (2) functional device.

device end. An indication from an I/O device that it has ended an operation.

device reconstruction. The SVA automatic background function of recreating and rewriting all of the data that was stored on a failed device to a spare device using the functional track recovery process.

direct access storage device (DASD). A storage device in which the medium is always available to the read/write head without having to be mounted by an external agent.

disk array. The SVA's logical grouping of drive modules. See also: dual-redundancy disk array.

disk array capacity. The formatted physical capacity of a disk array excluding redundancy data.

Disk Array Controller. The SVA control unit that provides the interface intelligence between the host(s) and the back-end storage.

Disk Array Unit (DAU). A single physical frame containing drive modules that comprise the disk array storage in an SVA subsystem.

domain. See SCSI domain.

drain. The SVA process that gradually moves data stored on a device or a disk array to other devices. Drain operations allow for the nondisruptive deinstallation of a device or a Disk Array Unit.

drive module. A disk storage device consisting of the access arms and heads, disk surfaces, and the supporting electronics required to locate, write, and read data. Each drive module is physically packaged as a single field-replaceable unit (FRU) within the SVA.

drive reconstruction. See device reconstruction.

dual-redundancy disk array. A disk array that allows for real-time automatic recovery of data from up to two failed devices within the array.

In the V2X and V960 SVAs, a dual-redundancy disk array consists of 15 (13+2) drive modules. The array has a capacity equivalent to 13 drives of user data and 2 drives of redundancy data. (In the SVA, redundancy data is distributed among all 15 drives).

In the 9500 and earlier SVAs, arrays of 7 (5+2) drive modules can also be formed.

Dynamic Configuration. An SVA feature that allows the channel interfaces and up to 4096 functional volumes to be defined and/or altered. The functional configuration of an SVA subsystem can be determined by user requirements rather than available drive modules.

E

ECAM device. A functional device over which SVAA-based communication between the SVA Disk Array Controller and the host CPU(s) takes place.

esoteric names. The names a user assigns to DASD volumes that have the same device type.

ESCON channel. A channel that uses ESCON cables to transmit data between the host and the Disk Array Controller.

Extended Control and Monitoring (ECAM). The communications protocol that permits communication between SVAA and the SVA.

extent. A range of disk addresses expressed as a cylinder head range (CCHH) for a CKD device, or a logical block address (LBA) for a SCSI device.

F

fast write. A write operation that does not require immediate synchronous transfer of data to a DASD device, thus reducing the time an application must wait for channel end and device end for an I/O operation.

fault symptom code (FSC). An error code, generated by a control unit or subsystem, that points to the area or FRU most likely causing a problem.

fault tolerance. The capability of a subsystem to continue operating without interruption and/or intervention despite a failure within the subsystem (e.g., hardware, power, cooling). Fault tolerance is generally measured in relation to inherent reliability, availability, serviceability, and recoverability for the product.

FDID map. See functional device identifier mapping.

fence. The automatic or manual separation of a logical path or physical component from the remaining operating portion of the subsystem. The fencing process provides for continuous operation of the subsystem and allows for deferred nondisruptive servicing of field-replaceable units (FRUs) via hot-plugging.

A logical barrier on a node or path that prevents the use of that node or path.

FICON channel. A channel that uses fiber connections to transmit data between the host and the Disk Array Controller.

field-replaceable unit (FRU). The smallest self-contained component that can be individually replaced during a service or repair action.

fixed block architecture (FBA). (Contrast with CKD) A recording format in which every track of the device is formatted with a fixed number of fixed-length records (generally called sectors), each of which contains an identifier (ID) field and a data field.

flexvolume. A 3380 or 3390 CKD volume defined with less than the maximum number of cylinders. The range of cylinders allowed depends on the device type.

free array cylinder. An array cylinder that contains no current or non-current user data.

free space collection (FSC). The automatic SVA background task that relocates data from fragmented array cylinders in order to collect free space into empty array cylinders. Free space collection maximizes the efficiency of array cylinder writes.

free space collection load. The average percentage of array cylinder space that must be relocated in order to create empty array cylinders in the SVA.

front end. The portion of the SVA Disk Array Controller data path that passes data between the channels and the cache.

functional. The term used to describe the SVA interface as viewed by the host, application, and users. This interface appears as a 3990-3 subsystem interface.

functional/allocated. The user-allocated portion of a functional volume's space; that is, data sets as defined in the VTOC.

functional capacity. The data storage capacity that the host, application, and users view. Used in reference to the space available for storing data in (1) a single functional device, or (2) all defined functional devices in an SVA subsystem.

functional device. The volume image that the host operating system receives when the “Read Device Characteristics” CCW is issued.

functional device identifier (FDID). The identifier for a functional device as it is known to the SVA. FDIDs range from 0 to FFF (hexadecimal) or from 0 to 4095 (decimal).

functional free space. The unallocated/unused portion of a functional volume's space, as defined in the VTOC.

functional track. The equivalent of a 3380- or 3390-DASD track. A functional track record is stored on contiguous sectors in an allocated array cylinder.

functional track directory (FTD). The SVA internal mapping table that contains one entry for each functional track associated with the functional volumes currently defined by the user.

functional track recovery (FTR). The automatic SVA process of recovering data from a physical track that is unreadable due to a media defect or a failed device. The SVA accomplishes functional track recovery by reading and processing the user data and redundancy data at corresponding physical track locations on the remaining devices in the array.

functional volume. See functional device.

G

generation data group (GDG). A collection of data sets with the same base name, such as PAYROLL, that are kept in chronological order. Each data set is called a generation data set.

global spares. See spare devices.

L

large volume. A 3390-9 CKD volume defined with 32760 cylinders.

link address. An address assigned during initialization that identifies a channel or control unit so that the channel or control unit can send and receive frames, and perform I/O operations. See logical paths.

LLAPI. An ECAM device driver available to vendors which provides the ability to query an SVA subsystem and its devices as well as the ability to manipulate functional tracks.

logical array. A grouping of devices into an array. The grouping of devices does not depend on their physical location.

logical partition. The subset of a processor unit that is allocated to support the operation of a systems control program.

logical paths. The relationship between a channel and a control unit that designates the physical path to be used for device-level communication between the channel and the control unit. This relationship is defined within the channel and control unit by a link address assigned to the control unit and a link address assigned to the channel.

M

MAT partition. The SVA partition consisting of drive modules that are not yet available for storing user data. Drive modules are automatically members of the MAT partition when they are first physically inserted in the SVA or when they have been drained of data.

Media Acceptance Test partition. See MAT partition.

N

net capacity load (NCL). This number is two KB times the number of physical sectors actually used to store user data, not including redundancy data. NCL is a percentage of the total number of sectors that are storing user data and is based on physical capacity used.

nonquiesced snap. A snap taken when the system is in full read-write access mode.

nonvolatile storage (NVS). The redundant solid state memory in the Disk Array Controller that remains active when ac power is removed. NVS protects any data that has not been written to the disk arrays.

P

Parallel Access Volume. A combination of a real device (Base) and one or more pseudo-devices (Aliases) that together support multiple concurrent I/Os to enhance performance.

parallel channel. A channel that uses bus-and-tag cables to transmit data between the host and the Disk Array Controller.

partition. The logical separation of devices, arrays, or groups of arrays to allow different modes of operation. The SVA supports a MAT partition, a Test partition, a Production partition, a Spares partition, and an Unavailable partition.

Note: The Test partition is not available in the V2X, V960, or 9500 SVA.

PAV. See Parallel Access Volume.

physical capacity. The physical space contained in (1) a single drive module, (2) a partition, or (3) an SVA subsystem.

physical device. See drive module.

privileged ECAM device. Privileged ECAM devices are the only devices that SVAA can use to send messages to the subsystem to request a change in the SVA's state. Such messages include those that alter the subsystem configuration or start a drain.

At least one privileged ECAM device must be defined in each SVA; all functional volumes in an SVA subsystem can be defined as privileged ECAM devices.

Production partition. The SVA partition consisting of drive modules assigned to production arrays for storing user data.

PROFSIBA macro. The profile executed when the SVAA SIBADMIN program is started.

PROFSIBS macro. The profile executed when the SVAA Subsystem Reporting Program is started.

Q

quiesce. To end a process by allowing operations to complete normally.

quiesced snap. A snap taken while the system is quiesced; all buffered transactions are flushed to disk storage.

R

read hit. The situation in which data requested by the read operation is located in cache.

read miss. The situation in which data requested by the read operation is not located in cache.

reconstruction. See device reconstruction

redundancy group. A logical grouping of devices that are protected from data loss due to a device failure by the use of redundancy (parity) data that is stored across the devices. Arrays in the SVA are redundancy groups that protect data against two simultaneous device failures. See also: dual-redundancy disk array.

Reporter. The SVAA subsystem reporting program—the SVAA component that collects subsystem performance data and produces reports based on that data, as well as on space utilization.

S

SCSI channel. See SCSI I/O interface.

SCSI domain. An SVA addressing scheme, prefixed to SCSI target and LUN addresses, that extends the number of addressable volumes from SCSI-attached host systems.

serial channel. A channel that uses fiber-optic (ESCON) cables to transmit data between the host and the Disk Array Controller. See also: ESCON channel.

Service Information Message (SIM). A message generated by the host processor upon receipt of sense information from the SVA that contains notification of a need for repair or customer action, or status information.

Shared Virtual Array (SVA). StorageTek's online, random access disk array storage subsystem composed of a Disk Array Controller and 16 to 64 disk drive modules.

Shared Virtual Array Administrator (SVAA). StorageTek's host software product that enables implementation of the extended storage management facilities of the SVA, and offers additional functions including SnapShot, DDSR, and reporting capabilities.

SIBADMIN module. The module used to invoke SVAA in command mode.

SIBBATCH module. The SVAA module for batch configuration and reporting.

SIBDSR module. The SVAA module for deleted data space release (DDSR).

SIBIOATX exit. The Subsystem I/O Authorization user exit that verifies a user's authority to send a control message to a subsystem.

SIBSNDAX exit. The Command Authorization user exit that verifies a user's authority to send a command to a subtask in the SVAA address space.

SIBSRP module. The SVAA module for the subsystem reporting program.

SIBSRPSW exit. The Subsystem File Switch user exit that is invoked by the SVAA SRP data collection task before closing or switching a logging file.

SIBVOP module. The SVAA operator console command program.

SIM alert. An operator console message that alerts the operator that an action requiring attention has occurred.

slot. The physical location of an SVA subsystem drive module.

snap. (noun) A duplication of a source volume or data set with SnapShot (see SnapShot). A snap is also the result of a successful SnapShot operation (not the use of a data mover). Synonymous with SnapShot. Contrast with *data mover copy*.

snap. (verb) To duplicate a functional volume or data set with SnapShot.

SnapShot. StorageTek's high-speed data-duplication facility, available only with the SVA and packaged with SVAA. SnapShot achieves great time-savings in duplicating volumes or data sets because it only creates a second set of pointers to the data. No additional physical disk space is used in the process.

source. The minidisk or volume from which data is snapped.

spare devices. SVA drive modules that are physically installed but not logically associated with an array. Spare devices are used by the SVA to form new arrays or to automatically reconstruct and logically replace failed devices.

spares. See spare devices.

Spares partition. The SVA partition consisting of all of the spare devices in the subsystem. See spare devices.

SSID. See subsystem identifier (SSID)

Status Bridge. A pair of devices used by Power PPRC to transmit acknowledgements that the data was received at the other end. These devices are not used to store customer data.

storage cluster. A power and service region that processes channel commands and controls the data storage devices. The SVA contains two storage clusters, each of which contains interfaces for up to 16 channels.

Storage Management Subsystem (SMS). An IBM approach to storage management in which a host system determines data placement and an automatic data manager handles data backup, movement, space, and security.

subsystem free space. Storage space in the disk arrays that does not contain user data.

subsystem identifier (SSID). The identifier for a 3990 controller emulated within the SVA. From one to sixteen SSIDs (logical 3990s) can be defined in each subsystem. Within an installation, each logical 3990 is defined by a unique four-digit (hexadecimal) SSID.

subsystem reporting program (SRP). The SVAA component that collects subsystem performance data and produces reports based on that data, as well as on space utilization. See also: Reporter.

SVAA profile facility. When invoked, this facility allows the user to specify commands for an SVAA session.

T

target. The minidisk or volume to which data is snapped.

Test partition. The SVA partition consisting of drive modules assigned to a test array and containing test data. The Test partition allows user-controlled, host-driven, testing of arrays, as though they were production arrays.

Note: The Test partition is not available in the V2X, V960, or 9500 SVA.

tray. The physical packaging of eight drive modules within the disk array area of the SVA.

U

Unavailable partition. The SVA partition consisting of drive modules that are not available for use in an array. Drive modules that are not installed or have failed are in this partition.

unit. See Disk Array Unit.

V

| **virtual control unit (VCU)**. Identifies a virtual control
| unit emulated within the SVA. There are currently 16
| VCUs within a V2X or V2Xf represented by
| hexadecimal values 0-F. These VCUs are identified to
| the host system as subsystem identifiers (SSID), which
| are defined at installation time.

virtual cylinder. An operating system unit of measure available to allow a system administrator to view and manage the total amount of functional cylinders available in an SVA subsystem.

virtual device identifier (VDID). Another term for FDID. See functional device identifier (FDID).

virtual operator panel (VOP). An SVAA facility that allows operator interaction with, and control of, the SVA via a host operator console in lieu of the local operator panel (LOP), the DOP, or remote operator panel.

volatile memory. See cache volatile memory.

volume. See functional volume.

volume preferencing (VP). A facility of SnapShot that filters the choice of an SMS target volume for newly allocated data sets.

volume serial number. A six-character alphanumeric name that identifies a disk volume to the host operating system.

W

write hit. The situation in which data to be updated by a write operation is located in cache.

write miss. The situation in which data to be updated by a write operation is not located in cache.

Abbreviations and Acronyms

API	application programming interface
BFDID	base functional device ID
CCHH	cylinder-head address (CC is the two-byte cylinder number, HH is the two-byte head number
CCW	channel command word
CFW	cache fast write
CKD	count-key-data
CLI	command line interface
COD	Capacity on Demand
CSA	common service area
CSI	consolidated software inventory
DADSM	direct access device space management
DASD	direct access storage device
DAU	Disk Array Unit
DDSR	Deleted Data Space Release
DFP	Data Facility Product
DFSMS	Data Facility Storage Management Subsystem
DFW	DASD fast write
DLIB	distribution library
DOP	Detached Operator Panel
DSF	Data Support Facilities
DTL	domain-target-LUN
ECAM	Extended Control and Monitoring
ECSA	extended common service area
ESA	Enterprise Systems Architecture
ESCON	Enterprise Systems CONnection
ESDI	enhanced small device interface
FDID	functional device identifier
FICON	FIbre CONnection
FMID	function modification identifier
FRU	field-replaceable unit
FSC	fault symptom code, or free space collection
FTD	functional track directory
FTR	functional track recovery
GB	gigabyte
GDG	generation data group
GRS	Global Resource Serialization
GTF	generalized trace facility
HCD	hardware configuration definition
HSI	Host Subsystem Interface
ICKDSF	ICK Data Support Facilities
IDCAMS	IDC access method services
IDID	interface device identifier
IML	initial microprogram/microcode load
IOCP	I/O configuration program
IPL	initial program load
ISPF	Interactive System Productivity Facility
I/O	input/output
LBA	logical block address
LOP	Local Operator Panel
LUN	logical unit number
MAT	Media Acceptance Test
MB	megabyte
MIH	missing interrupt handler
MVS	Multiple Virtual Storage
NCL	net capacity load
NVS	nonvolatile storage
PAV	Parallel Access Volume
PCAP	physical capacity
PPRC	peer-to-peer remote copy
PTF	program temporary fix
RACF	Resource Access Control Facility
RAID	redundant array of inexpensive disks
REXX	Restructured Extended Executor
RFA	record format assist
RIM	resource initialization module
RMF	resource measurement facility
SAF	Security Access Facility
SCP	system control program
SCSI	small computer system interface
SFC	Server Framework Component
SIM	service information message
SMF	system management facility
SMP/E	System Modification Program Extended

Abbreviations and Acronyms

SMS	Storage Management Subsystem
SRP	Subsystem Reporting Program
SSID	subsystem identifier
SVA	Shared Virtual Array
SVAA	Shared Virtual Array Administrator
TSO	Time Sharing Option
UCB	unit control block
VCU	virtual control unit
VDID	virtual device identifier
volser	volume serial number
VOP	virtual operator panel
VP	volume preferencing
VSAM	Virtual Storage Access Method
VTOC	volume table of contents
VVDS	VSAM volume data set
VVR	VSAM volume record

Index

Reader's Comment Form

Product Name: Shared Virtual Array Administrator for OS/390 **Software Level:** Version 3.1
Manual Name: Installation, Customization, and Maintenance **Document Number:** 311290809

Please check or fill in the items, adding explanations or comments in the spaces provided.

Which of the following terms best describes your job?

_ Field Engineer	_ Manager	_ Programmer	_ Systems Analyst
_ Engineer	_ Mathematician	_ Sales Representative	_ Systems Engineer
_ Instructor	_ Operator	_ Student/Trainee	_ Other (explain below)

How did you use this publication?

_ Introductory text _ Reference manual _ Student/Trainee _ Instructor text
_ Other (explain) ____

Did you find the material easy to read and understand?	_ Yes	_ No	(explain below)
Did you find the material organized for convenient use?	_ Yes	_ No	(explain below)

Specific criticisms (explain below):

Clarifications on pages ____
Additions on pages ____
Deletions of pages ____
Errors on pages ____

Explanations and other comments:

Note: *Staples can cause problems with automated-mail sorting equipment. Please use pressure-sensitive tape to seal this form. If you would like a reply, please supply your name and address on the reverse side of this form. Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A.*

TO COMPLY WITH POSTAL REGULATIONS, FOLD EXACTLY ON DOTTED LINES AND TAPE (DO NOT STAPLE)

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 2 LOUISVILLE, CO U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

STORAGE TECHNOLOGY CORPORATION
MANAGER, DISK STORAGE LEARNING PRODUCTS
ONE STORAGETEK DRIVE
LOUISVILLE, COLORADO 80028-2121
U.S.A.

FOLD HERE AND TAPE DO NOT STAPLE FOLD HERE AND TAPE

If you would like a reply, please print:

Your Name: ______________________

Company Name: ______________ Department: ______________

Street Address: ______________________

____

Cut or Fold Along Line

Printed in U.S.A.

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN **Web** sun.com

Argentina: 5411-4317-5636 • **Australia:** 1-800-550-786 • **Austria:** 43-1-601-26-0 • **Balkans:** 301-6188-111 • **Belgium:** 32 2-704 89 83 • **Brazil:** 55-11-51872100 • **Brunei:** 65-216-8333 • **Canada:** 1-800-422-8020 **(General);** 416-964-2001 **(Learning Management System Sales,Toronto)** • **Chile:** 562-372-4500 • **Colombia:** 571-629-2323 **Czech Republic:** 420 2 33009311 • **Denmark:** 45 4556 5040 • **Egypt:** 00 202 570 9442 • **Finland:** 358-9-525-561 • **France:** 33-1-41-33-17-17 • **Germany:** 49-89-460-08-2788 • **Greece:** 30-01-6188101 • **Hong Kong:** 852-2877-7077 • **Hungary:** 361-202-4415 • **India:** 91-80-229-8989 • **Indonesia:** 65-216-8333 • **Ireland:** 353-1-668-4377 **Israel:** 972-9-9710500 • **Italy:** 39-02-9259511 • **Japan:** 81-3-5779-1820 • **Korea:** 82-2-3453-6602 • **Malaysia:** 603-2116-1887 • **Middle East:** 00 9714 3366333 • **Mexico:** 525-261-0344 • **Netherlands:** 31-33-4515200 • **New Zealand:** 0800-786-338 • **North West Africa:** 00 9714 3366333 • **Norway: From Norway:** 47-22023950, **To Norway:** 47-23369650 • **Pakistan:** 00-9714-3366333 • **People's Republic of China:** 8610-6803-5588 • **Philippines:** 632-885-7867 • **Poland:** 48-22-8747848 • **Portugal:** 351-21-413-4000 • **Russia:** 7-095-935-8411 • **Saudi Arabia:** 00 9714 3366333 • **Singapore:** 65-216-8300 • **South Africa:** 27-11-256-6300 • **Spain:** 34-902-210-412 • **Sri Lanka:** 65-2168333 • **Sweden:** 46-8-631 22 00 • **Switzerland:** 41-1-908-90-50 **(German)** 41-22-999-0444 **(French)** • **Taiwan:** 886-2-25185735 • **Thailand:** 662-344-6855 • **Turkey:** 90 212 335 22 00 • **United Kingdom:** 44-1276-416-520 • **United States:** 1-800-422-8020 • **Venezuela:** 582-905-3800 • **Vietnam:** 65-216-8333 • **Worldwide Headquarters:** 1-650-960-1300