OS/390

MVS Using the Subsystem Interface

SC28-1789-07

OS/390

MVS Using the Subsystem Interface

SC28-1789-07

Note

Before using this information and the product it supports, be sure to read the general information under Appendix C, “Notices” on page 295.

Eighth Edition, September 2000

This is a major revision of SC28-1789-06.

This edition applies to Version 2 Release 10 of OS/390 (5647-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/s390/os390/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

About This Book

This book introduces you to subsystems, what they are and why you might want to write your own. It describes how to set up your subsystem and how to use it. MVS provides some services to help you build and use subsystems; these services are described in this book.

In addition, this book describes services provided by IBM subsystems that a program can use. The program need not be a subsystem to use these services.

Who Should Use This Book

This book is for ***system programmers*** or ***application developers*** who are writing a subsystem or requesting system services available through the subsystem interface (SSI).

This book assumes that the reader has extensive experience with MVS, is familiar with its basic concepts, can code JCL statements to execute programs or cataloged procedures, can code in assembler language, and can read assembler, loader, and linkage editor output.

How to Use This Book

Depending upon the tasks you want to perform, the following is a guide to the chapters you can refer to.

For general information about the SSI, see “Introduction to Subsystems and the Subsystem Interface (SSI)” on page 1.

If you are familiar with the SSI, and you are writing a program that uses services provided by IBM subsystems, see:

- “Making a Request of a Subsystem” on page 7
- “SSI Function Codes Your Program Can Request” on page 13.

If you are familiar with the SSI, and you are writing your own subsystem, see:

- “Setting Up Your Subsystem” on page 157
- “Services for Building and Using Your Subsystem” on page 167
- “SSI Function Codes Your Subsystem Can Support” on page 185
- “Troubleshooting Errors in Your Subsystem” on page 255.

Where to Find More Information

Where necessary, this book references information in other books, using shortened versions of the book title. For complete titles and order numbers of the books for all products that are part of OS/390, see *OS/390 Information Roadmap*.

Summary of Changes

Summary of Changes
for SC28-1789-07
OS/390 Version 2 Release 10

This publication contains information previously presented in SC28-1789-06, which supports OS/390 Version 2 Release 8. The following summarizes the changes to that information.

New Information

The DOM Processing call (SSI Function Code 14) is added.

Changed Information

This and all prior editions of this book also reflect the addition, modification, and deletion of information to support miscellaneous terminology, maintenance, and editorial changes. Vertical lines in the left margin of this edition indicate technical changes or additions to the text and illustrations *in this edition.*

Summary of Changes
for SC28-1789-06
OS/390 Version 2 Release 8

This publication contains information previously presented in SC28-1789-05, which supports OS/390 Version 2 Release 7. The following summarizes the changes to that information.

New Information

- JES Client/Server Print Interface

Summary of Changes
for SC28-1789-05
OS/390 Version 2 Release 7

Changed Information

- ESTAE Routine
- SSI 78
- SSI 79 (Added new disposition indicator and new output field)

Summary of Changes
for SC28-1789-04
OS/390 Version 2 Release 5

This book contains information previously presented in *MVS/ESA Using the Subsystem Interface,* SC28-1789-03, which supports OS/390 Release 4.

The following summarizes the changes:

Changed Information

- SSI Function Code 54

  The SSI 54 function is updated for Client Print.

- SSI Function Code 79

  The SSI 79 function is updated for Client Print.

- SSI Function Code 80

  The SSI 80 function is updated for Client Print.

Summary of Changes
for SC28-1789-03
OS/390 Version 2 Release 4

This book contains information previously presented in *MVS/ESA Using the Subsystem Interface,* SC28-1789-02, which supports OS/390 Release 3.

The following summarizes the changes:

Changed Information

- SSI Function Code 80

  The SSI 80 function is enhanced to supply additional fields to support workload manager-managed (WLM) service classes.

Summary of Changes
for SC28-1789-02
OS/390 Release 3

This book contains information previously presented in *MVS/ESA Using the Subsystem Interface,* SC28-1789-01, which supports OS/390 Release 2.

The following summarizes the changes:

New Information

- **New SYSOUT Application Program Interface, SSI function code 79.**

  SSI function code 79 allows a user-supplied program to access JES SYSOUT data sets independently from the normal print or network functions that JES provides. This allows the program to process the SYSOUT and inform JES to dispose the data set. Disposition of the data set includes: deleting it, changing its characteristics, sending it across the network, or allowing a JES-managed device to process it.

  SSI function code 79 allows greater processing capability than does SSI function code 1 (Process SYSOUT).

- **New extended status function call, SSI function code 80.**

  SSI function code 80 is a general-purpose interface that allows a user-supplied program to obtain detailed status information about jobs in the JES queue. This SSI is supported by JES2 subsystems only.

Changed Information

- SSI Function Code 54

The SSI 54 function is enhanced to supply a string of these classes through the use of new IBM-defined keywords.

This SSI function code has been updated with new keywords.

Summary of Changes
for SC28-1789-01
OS/390 Release 2

This book contains information previously presented in *MVS/ESA Using the Subsystem Interface,* SC28-1789-00, which supports OS/390 Release 1.

Summary of Changes
for SC28-1789-00
OS/390 Release 1

This book contains information previously presented in *MVS/ESA Using the Subsystem Interface,* SC28-1502, which supports MVS/ESA System Product Version 5.

Introduction to Subsystems and the Subsystem Interface (SSI)

This chapter describes basic concepts that you need to understand if you want to write your own subsystem or want to use services provided by IBM subsystems.

What is a Subsystem?

A subsystem is a service provider that performs one function or many functions, but does nothing until it is requested. Although the term “subsystem” is used in other ways, in this book a subsystem must be the master subsystem or be defined to MVS in one of the following ways:

- Processing the IEFSSNxx parmlib member during IPL

  You can use either the keyword format or positional format of the IEFSSNxx parmlib member. IBM recommends that you use the keyword format, which allows you to define and dynamically manage your subsystems.
- Issuing the IEFSSI macro
- Issuing the SETSSI system command

(Note that the master subsystem (MSTR) is a part of MVS and is not defined in any of these ways.) Some examples of IBM-supplied subsystems that use the SSI:

- JES2
- JES3
- IMS
- NetView
- OPC

There are two types of subsystems:

- The **primary** subsystem. The job entry subsystem that MVS uses to do work. It can be either JES2 or JES3.
- **Secondary** subsystems. Secondary subsystems provide functions as needed by IBM products, vendor products, or the installation.

MVS communicates with subsystems through the SSI.

What is the SSI?

The SSI is the interface used by routines (IBM, vendor, or installation-written) to request services of, or to pass information to, subsystems. An installation can design its own subsystem and use the SSI to monitor subsystem requests. An installation can also use the SSI to request services from IBM-supplied subsystems. The SSI acts only as a mechanism for transferring control and data between a requestor and the subsystem; it does not perform any subsystem functions itself.

Unique Attributes of the SSI

The SSI is a way for one routine to call another routine. There are a number of other ways that a routine can call another routine, such as:

- Branch and link register (BALR) 14,15
- LINK or LINKX macro
- Program call (PC)
- SVC

The SSI is different from these linkage interfaces, however, in that:

- The called routine does not have to be there. That is, when a routine calls the subsystem, the SSI checks to see if the subsystem either is not interested in the request or does not exist. The caller then receives an appropriate return code.
- A caller's request can be routed to multiple subsystem routines.

Types of Subsystem Requests

The SSI handles two types of requests: **directed** requests and **broadcast** requests.

Directed requests, which can be defined by the installation, are made to **one** named subsystem. For a directed request, the caller informs the named subsystem of an event, or asks the named subsystem for information. For example, you can access JES SYSOUT data sets with a directed request.

Figure 1 shows the processing for a directed request.

See “SSI Function Codes Your Program Can Request” on page 13 for more information on the services available to your program using directed requests.

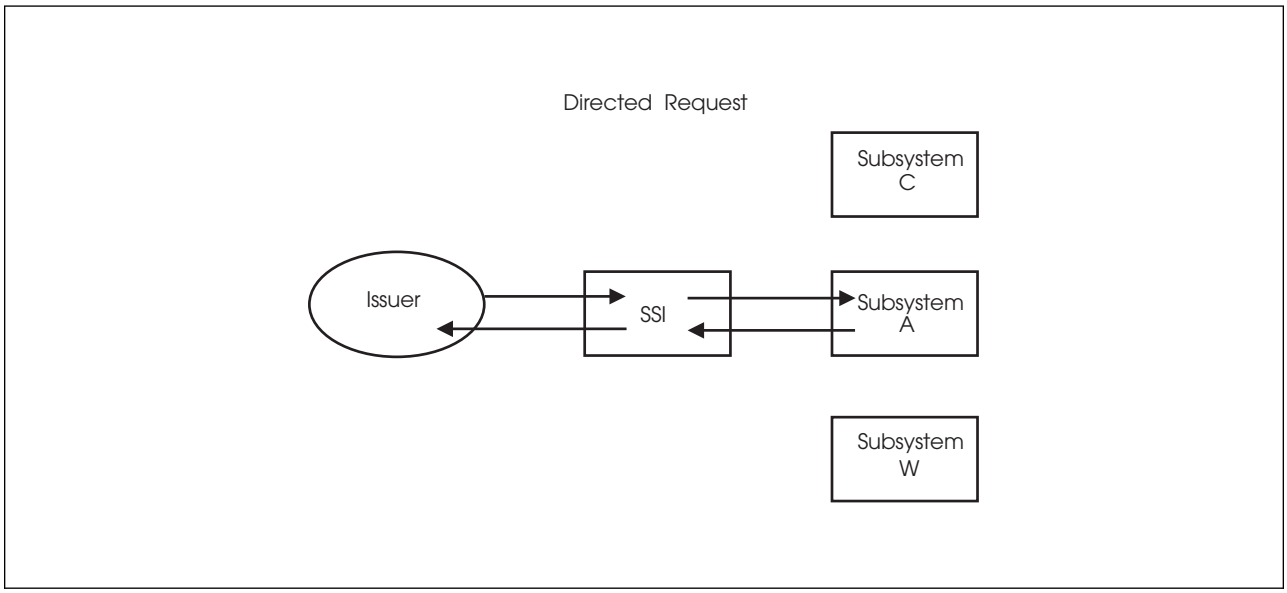

Figure 1. Processing for a Directed Request

Broadcast requests, which are defined by MVS, provide the ability for subsystems to be informed when certain events occur in the system. Broadcast requests, differ from directed requests, in that the system allows **multiple** subsystems to be informed when an event occurs. The SSI gives control to each subsystem that is

active and that has expressed an interest in being informed of the event. For example, your subsystem can be informed when a WTOR message is issued in order to automate a response to the WTOR.

Figure 2 shows the processing for a broadcast request.

See “SSI Function Codes Your Subsystem Can Support” on page 185 for more information on the broadcast function codes your subsystem can support.

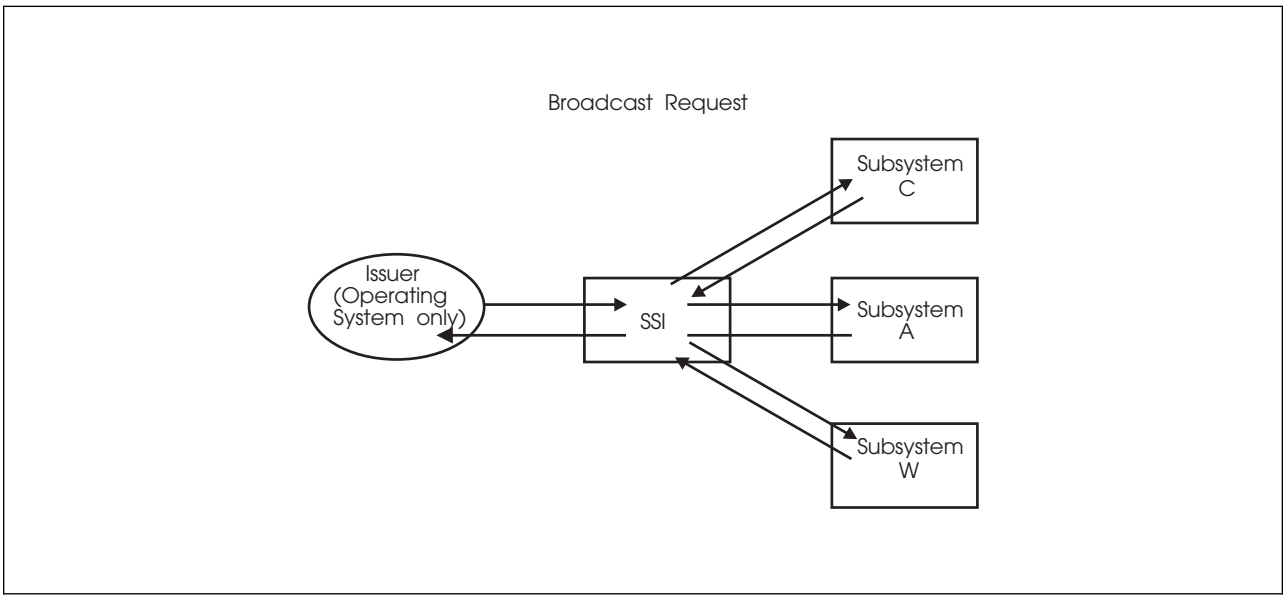

Figure 2. Processing for a Broadcast Request

Controlling SSI Processing

The IEFJFRQ installation exit provides a way to examine and modify subsystem function requests. See *OS/390 MVS Installation Exits* for more information on the capabilities and use of the IEFJFRQ exit.

Why Write Your Own Subsystem?

You can extend the function of the operating system by writing and invoking your own subsystem.

Using a subsystem for an installation-defined function not provided by MVS requires an in-depth knowledge of procedures, problems, and goals at your installation; as well as a knowledge of MVS. You must take many things into consideration when deciding whether a subsystem is needed. Some factors to consider include:

- You might have many programs that need the same functions. If you use a subsystem to supply these functions, the request is made in terms of the function needed.
- You might want to take installation-specific action in response to certain events. If these events cause a broadcast SSI call, you can set up a subsystem to receive control at that time. However, if you choose to make a subsystem eligible for a broadcast request, your subsystem gets control on every request

for that function. Thus, you must weigh the benefits of having the subsystem handle that function against the possible impact on performance.

- The requesting program can be isolated from problems involving the subsystem.
- Using subsystems to provide services allows more flexibility and compatibility. Not every change in the subsystem will require you to recompile; the interface between the requesting program and the subsystem remains the same.
- You might want to use a subsystem to set up installation requirements only at initialization time. During system initialization, control passes to the subsystem initialization routines named in SYS1.PARMLIB member IEFSSNxx. The initialization routine establishes changes defined by the installation. In this case, the initialization routine performs the function at initialization and does not set up separate function routines; the subsystem-provided programs that process the function identified by the function codes.

  You must decide whether you want to use this function of subsystems for this purpose. Consider that some of the control blocks built reside below 16 megabytes in common storage and, if your subsystem should fail, you may not be able to complete initialization of your system.

Do not use a subsystem to do the following:

- To anchor persistent control blocks. Use the Name/Token callable services instead. Subsystems that exist only to provide an anchor degrade the performance of SSI request processing. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on the Name/Token callable services.
- To receive control for end-of-task and end-of-memory conditions. Use the RESMGR macro instead. Subsystems that exist only as resource managers degrade the performance of SSI request processing. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on the RESMGR macro.

If you decide that you need a subsystem, see “Setting Up Your Subsystem” on page 157 for the information necessary to accomplish that task.

What is a Dynamic Subsystem?

Dynamic subsystems are those subsystems that can be defined in one of the following ways:

- Processing the keyword format IEFSSNxx parmlib member during IPL
- Issuing the IEFSSI macro
- Issuing the SETSSI system command.

Subsystems have the choice of being dynamic. Subsystems that are not dynamic can be defined only at IPL using the positional form of the IEFSSNxx parmlib member, in which case, they cannot use dynamic SSI services.

In addition to its role in communicating with subsystems, the SSI provides a set of authorized system services that are available only to dynamic subsystems that installations, applications and subsystems can invoke to:

- Define (add) a new subsystem dynamically (without requiring an IPL)

- Activate a subsystem that is already defined
- Deactivate a subsystem that is already defined
- Store and retrieve subsystem-dependent information
- Define subsystem options, which include deciding:
  - If a subsystem can respond to the SSI commands
  - Which subsystem a subsystem should start under.
- Query subsystem information
- Define and modify the response of a subsystem to function requests.

Defining or adding a subsystem is primarily a way of making the subsystem's unique name known to the system. A subsystem is active when it is ready to process requests that the SSI directs to it. To deactivate a subsystem, the subsystem informs the SSI that it is no longer accepting requests. For example, a subsystem may request that it be deactivated to update the list of function requests that it supports, or to respond to a problem.

The dynamic SSI services reject any requests to manipulate subsystems that were not defined dynamically.

The services that allow installations, applications and subsystems to define and modify the response of a subsystem to function requests replace and enhance the existing IEFJSVEC service. You can still use the existing IEFJSVEC service, which is described in Appendix B, “Using IEFJSVEC with Your Subsystem” on page 279, however IBM recommends that you use the services described in “Services for Building and Using Your Subsystem” on page 167 instead of IEFJSVEC. These services provide an easier way to define or change the functional response of a subsystem.

Making a Request of a Subsystem

This chapter describes how to use the SSI to make a request of a subsystem. The subsystem may either be an installation-defined subsystem, a vendor-supplied subsystem, or a subsystem provided by IBM. See “SSI Function Codes Your Program Can Request” on page 13 for the list of the functions that can be requested of IBM subsystems.

To request a function of a subsystem, do the following:

1. Set up the environment needed to make the request.
2. Make the request with the IEFSSREQ macro.
3. Check the information returned from both the SSI and the subsystem and take the appropriate action.

Set Up the Environment

With exceptions, your requesting program must be in the same state (problem or supervisor) as the subsystem. For IBM-supplied functions, see the specific function code descriptions in “SSI Function Codes Your Program Can Request” on page 13 for information on the environmental requirements that must be met. The SSI supports address mode (AMODE) switching. Your program must include mapping macros for the CVT and the JESCT control blocks.

Diagnosis, Modification or Tuning Information

Note that the IEFSSREQ macro uses the JESSSREQ field in the JESCT control block to locate the SSI routing routine.

End of Diagnosis, Modification or Tuning Information

You must tell the SSI the function you are requesting and the subsystem with which you want to communicate. You make a request by obtaining storage for the following control blocks:

- SSOB
- SSOB function dependent area (if required)
- SSIB.

These control blocks, and your program's save area, must reside in an area addressable by the called subsystem's function routine.

Subsystem Options Block (SSOB)

The subsystem options block (SSOB) identifies the function that you are requesting. The SSOB consists of a 28-byte header that you must fill in for every call to a subsystem through the SSI. The SSOB is the parameter list for the IEFSSREQ macro.

Function codes are the way a caller identifies the service or processing requested of a subsystem. You specify a function code by placing the appropriate code in the SSOBFUNC field. Another important field is SSOBRTRY. In the case of an abend,

this flag determines whether the directed function recovery routine will percolate or retry. IBM recommends that you set this flag. Setting this flag will cause the SSI to attempt to resume processing if it fails. If the flag is not set, the SSI will percolate by default.

Use the IEFSSOBH mapping macro to build the SSOB header.

SSOB Function Dependent Area

In addition to the SSOB, the specific function you invoke might require additional information, which can be passed in a function dependent area identified in the SSOB. You specify which SSOB function dependent area that you want to use by setting the SSOBINDV field in the SSOB to the address of the SSOB function dependent area.

The mapping macro used to map the SSOB function dependent area varies based on the specific function you invoke.

Subsystem Identification Block (SSIB)

The subsystem identification block (SSIB) identifies the particular subsystem to which a request is being directed. Your program can provide an SSIB or can use an SSIB provided by the system.

A **life-of-job SSIB** is an SSIB that is automatically provided by the system. The subsystem name specified in the life-of-job SSIB is the name of the subsystem that initiated the currently running job, started task, or TSO/E user. This is usually the primary JES, but could be:

- An alternate JES2
- The master subsystem

If your program does not create an SSIB, it must set the address of the SSIB in the SSOB (SSOBSSIB) to zero. This setting tells the system to use the life-of-job SSIB.

Before you make an SSI request you need to evaluate whether the subsystem name provided by the system in the life-of-job SSIB is the correct subsystem for the function you are requesting. The system provides the subsystem name in the life-of-job SSIB, based on whether the unit of work is a batch job, started task, or LOGON as follows:

- Batch jobs

  For batch jobs, the job is initiated under the JES that selects the job, that is, either the primary or alternate JES. The subsystem name in the life-of-job SSIB is either the primary or alternate JES subsystem name.

- Started tasks

  If a START command with the SUB= parameter is specified, the started task is initiated under the subsystem name specified on the SUB= parameter. This is also the subsystem name in the life-of-job SSIB.

  If you specify SUB=MSTR, the master subsystem starts the job even if it is not a subsystem. To do this, however you must meet the requirements of the master subsystem. See *OS/390 MVS JCL Reference* for considerations when running a started task under the master subsystem.

If a START command (without the SUB= parameter) is specified, and is for a started task with the same name as a subsystem that is capable of being a job entry subsystem (JES), the started task is initiated under the master subsystem. The subsystem name in the life-of-job SSIB is MSTR.

If a START command (without the SUB= parameter) is specified and is for a started task with the same name as a subsystem that is not capable of being a job entry subsystem (JES), the started task is initiated under the primary JES subsystem. The subsystem name in the life-of-job SSIB is the primary JES subsystem name.

If a START command (without the SUB= parameter) is specified and is for a started task with a name that is not the name of a subsystem, the started task is initiated under the primary JES subsystem. The subsystem name in the life-of-job SSIB is the primary JES subsystem name.

- TSO/E users

  For TSO/E users, the LOGON is initiated under the primary JES. The subsystem name in the life-of-job SSIB is the primary subsystem name.

If the subsystem name provided in the life-of-job SSIB is not the correct subsystem name based on the function you want to invoke, your program must provide an SSIB. See “SSI Function Codes Your Program Can Request” on page 13 for the subsystem name that must be specified when making requests for functions provided by IBM subsystems.

To create an SSIB, your program can use the following procedure:

1. Map the format of the SSIB with the IEFJSSIB mapping macro.
2. Clear the fields in the SSIB to binary zeros.
3. Set the SSIBID and SSIBLEN fields to the appropriate values.
4. Set the SSIBSSNM field to the name of the subsystem. (If the subsystem name is less than 4 characters, it should be specified left-justified, and padded to the right with blanks.)
5. Set the SSIBJBID field if required.
6. Set the SSIBSUSE field if required.

   Note: The SSIBSUSE field is available for use by a vendor-supplied or installation-supplied subsystem for an SSIB that a program provides on a vendor-defined or installation-defined SSI request. A vendor-defined or installation-defined subsystem must not use the SSIBSUSE field in the life-of-job SSIB. Your program may need to set the SSIBSUSE field if the IBM-defined, vendor-defined or installation-defined SSI request requires it.
7. Store the address of the SSIB in the SSOB (SSOBSSIB field).

Make the Request with the IEFSSREQ Macro

When you have set up the environment and built the necessary control blocks, you are ready to issue the IEFSSREQ macro to make the request. There are no parameters on the IEFSSREQ macro; the SSOB, SSOB function dependent area (if provided), and SSIB provide the information the SSI and the subsystem need to perform their function.

Input Register Information

Before issuing the IEFSSREQ macro, the caller must ensure that the following registers contain the specified information:

Register	Contents
1	Address of a one-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.
13	Address of a standard 18-word save area.

Syntax of IEFSSREQ

The syntax of the IEFSSREQ macro is:

`[symbol]`	`IEFSSREQ`

where *symbol* is any valid assembler language symbol. Note that one or more blanks are required before and after IEFSSREQ.

Figure 3 shows the environment when you make a subsystem request.

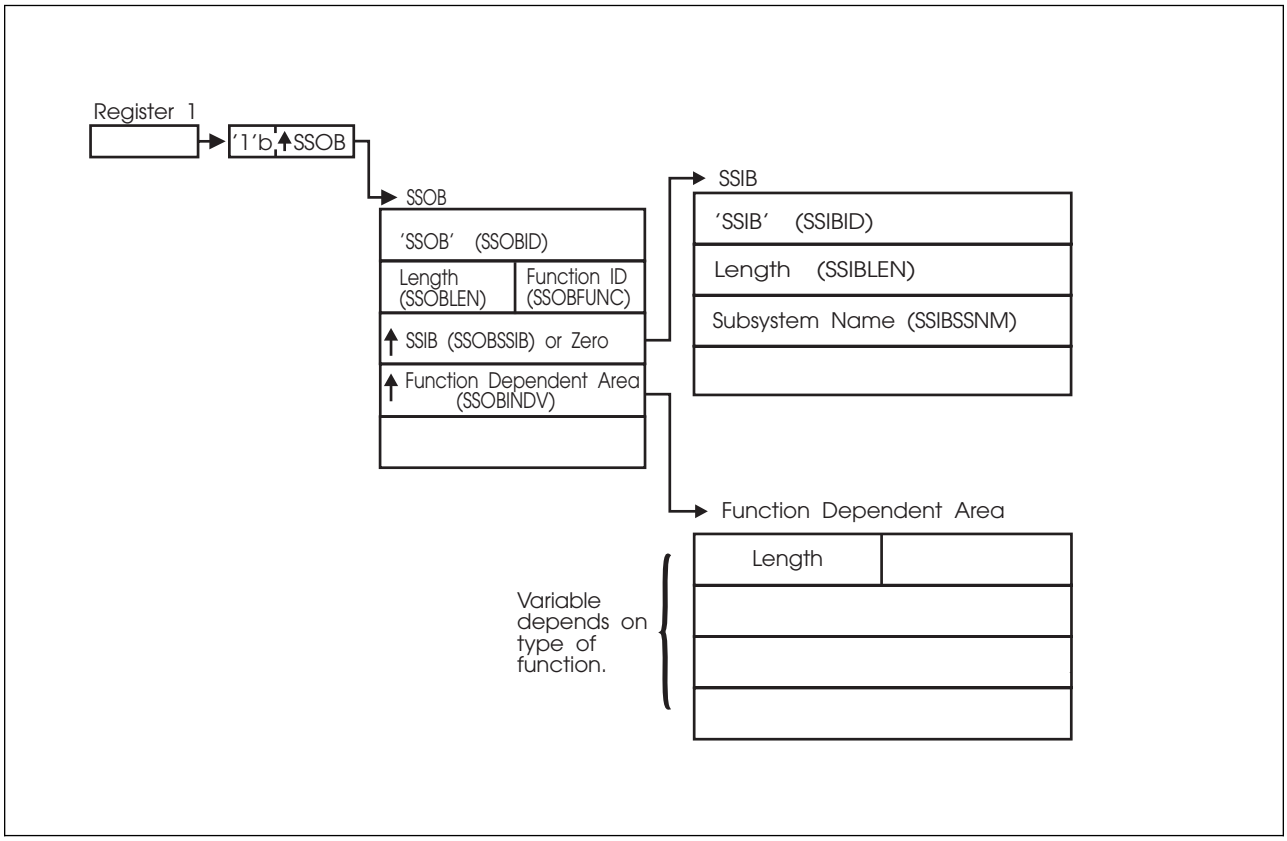

Figure 3. Making a Subsystem Request

Check the Return Code

For a directed subsystem request, the SSI returns one of the following decimal return codes in register 15:

Return Code (Decimal)	Meaning
SSRTOK (0)	Successful completion — the request went to the subsystem.
SSRTNSUP (4)	The subsystem does not support this function.
SSRTNTUP (8)	The subsystem exists, but is not active.
SSRTNOSS (12)	The subsystem is not defined to MVS.
SSRTDIST (16)	The pointer to the SSOB control block or the SSIB control block is not valid, or the function code specified in the SSOBFUNC field is greater than the maximum number of functions supported by the subsystem specified in the SSIBSSNM field.
SSRTLERR (20)	The SSOB or SSIB have invalid lengths or formats
SSRTNSSI (24)	The SSI has not been initialized.

If the return code in register 15 is zero, the SSI was able to pass the request to the subsystem, and the SSOB function dependent area might contain information returned by the subsystem. The contents of the return code in the SSOB (SSOBRETN), and other fields, depend on which function you requested.

Summary of Steps

When issuing the IEFSSREQ macro you can follow these steps:

1. Set up the environment:
   - Obtain storage for control blocks
   - Set up register 1 and 13 (Note that the save area must be accessible to the function routine.)
   - Initialize the SSOB
   - Initialize the SSOB function dependent area (if required)
   - Initialize the SSIB (if necessary)
   - Enter supervisor state (if necessary)
2. Make the request:
   - Invoke IEFSSREQ
   - Return to problem state (if necessary)
3. Check the return codes:
   - Check the SSI return code in register 15 and the subsystem return code in SSOBRETN, and take appropriate action.
   - Free the storage.

Example 5 in Appendix A, Examples — Subsystem Interface Routines shows a coding example of a routine making a request of a subsystem.

SSI Function Codes Your Program Can Request

This chapter contains detailed information on directed function codes your program can request. IBM subsystems provide these function codes.* The following is a list of SSI function codes, along with their purpose, the subsystems that support the function and the type of subsystem request.

Function Code	Requested Function	Subsystem*	Type of Request
1	Process SYSOUT data sets	JES2/JES3	Directed
15	Verify subsystem function	Master	Directed
20	Request job ID	JES2/JES3	Directed
21	Return job ID	JES2/JES3	Directed
54	Request subsystem version information	JES2/JES3/Master	Directed
75	Notify user message service	JES2/JES3	Directed
79	SYSOUT Application Program Interface (SAPI)	JES2/JES3	Directed
80	Extended Status Function Call	JES2/JES3	Directed

*Not all supported levels of the IBM subsystems support all the function codes available with the current release of OS/390. JES2 users should refer to the *OS/390 JES2 Migration*, and JES3 users should refer to the *OS/390 JES3 Migration* for specific information about which levels of their subsystem support these function codes.

Your program need not be a subsystem to use these function codes. In addition to the SSI function codes provided by the operating system, installations can also define and use their own function codes, using the range 236 to 255. You can design your own directed requests for these function codes.

SSI Function Code Descriptions

Your program can use several SSI function codes when coding for an MVS/ESA-JES2/JES3 environment. This section contains detailed descriptions of the SSI function codes listed at the beginning of this chapter.

See example 5 in Appendix A, Examples — Subsystem Interface Routines for a coding example of a routine making a request of a subsystem.

Process SYSOUT Data Sets Call — SSI Function Code 1

The Process SYSOUT Data Sets call (SSI function code 1) allows a user-supplied program to access JES SYSOUT data sets independently from the normal functions (such as print, network) JES provides, so that the characteristics of the SYSOUT data sets can be either retrieved or updated. The program using this interface is called an external writer. It operates in an address space external to JES, generally for requesting and printing JES-managed SYSOUT data sets that reside on spool.

Retrieval Attributes: For both JES2 and JES3, the program can select SYSOUT data sets for retrieval purposes according to a variety of different selection attributes, such as the form name or SYSOUT class. Both JES2 and JES3 can either keep or delete the retrieved data set.

Update Attributes: For JES3 only, the program can select SYSOUT data sets for update purposes according to a variety of different selection attributes, such as the destination or SYSOUT class. The program can even delete data sets from the JES spool.

Type of Request

Directed SSI call.

Use Information

The caller of the SSI function code 1 is the external writer. See “External Writer Considerations” on page 32 for detailed information on the definition of a standard external writer. See also *OS/390 JES2 Initialization and Tuning Guide* and *OS/390 JES3 Initialization and Tuning Guide* for more information on the external writer.

The external writer uses SSI function code 1 to retrieve (JES2 and JES3) and update (JES3 only) JES-managed SYSOUT data sets, allowing the writer to perform processing that JES does not provide.

For example, while JES provides the ability to print locally on a variety of printers, JES does not provide direct support for all forms of devices, such as microfiche printers. SSI function code 1 allows other programs to select SYSOUT from JES, and thus process it with their own devices.

Additionally, the function exists for these programs to perform disposition processing on the SYSOUT data set according to program control. For example, after reading the SYSOUT data set to a microfiche printer, the program may tell JES to do one of the following:

- Delete the data set
- Hold the data set for additional processing
- Reroute the data set to a different local or remote destination.

Before using the process SYSOUT data sets call, investigate using the functional system interface (FSI) as an alternative. The FSI also provides facilities for selection of SYSOUT work destined to an outside address space. See *OS/390 MVS Using the Functional Subsystem Interface* for more information on the FSI.

Issued to
JES2 or JES3.

Related SSI Codes
None.

Related Concepts
You should know how to use:

- Dynamic allocation (DYNALLOC) services to allocate/deallocate the JES-supplied data set.
- Sequential access method (SAM) to read the allocated SYSOUT data set and properly handle the process SYSOUT interface.
- Other standard MVS services, such as WAIT and POST logic.

Environment
Your external writer must include the following mapping macros:

- CVT
- IEFJESCT

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your program:

- IEFSSOBH
- IEFJSSIB
- IEFSSSO (with SOEXT=YES specified)

  Note: Specifying SOEXT=YES generates the 'long' form of the IEFSSSO with the PSO extension.

Your external writer must meet the following requirements:

Minimum Authorization	Supervisor state
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control Parameters	The SSOB and SSSO control blocks can reside in storage above 16 megabytes.
Recovery	The external writer should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on an ESTAE-type recovery environment.

Figure 4 on page 16 shows the environment at the time of the call for SSI function code 1.

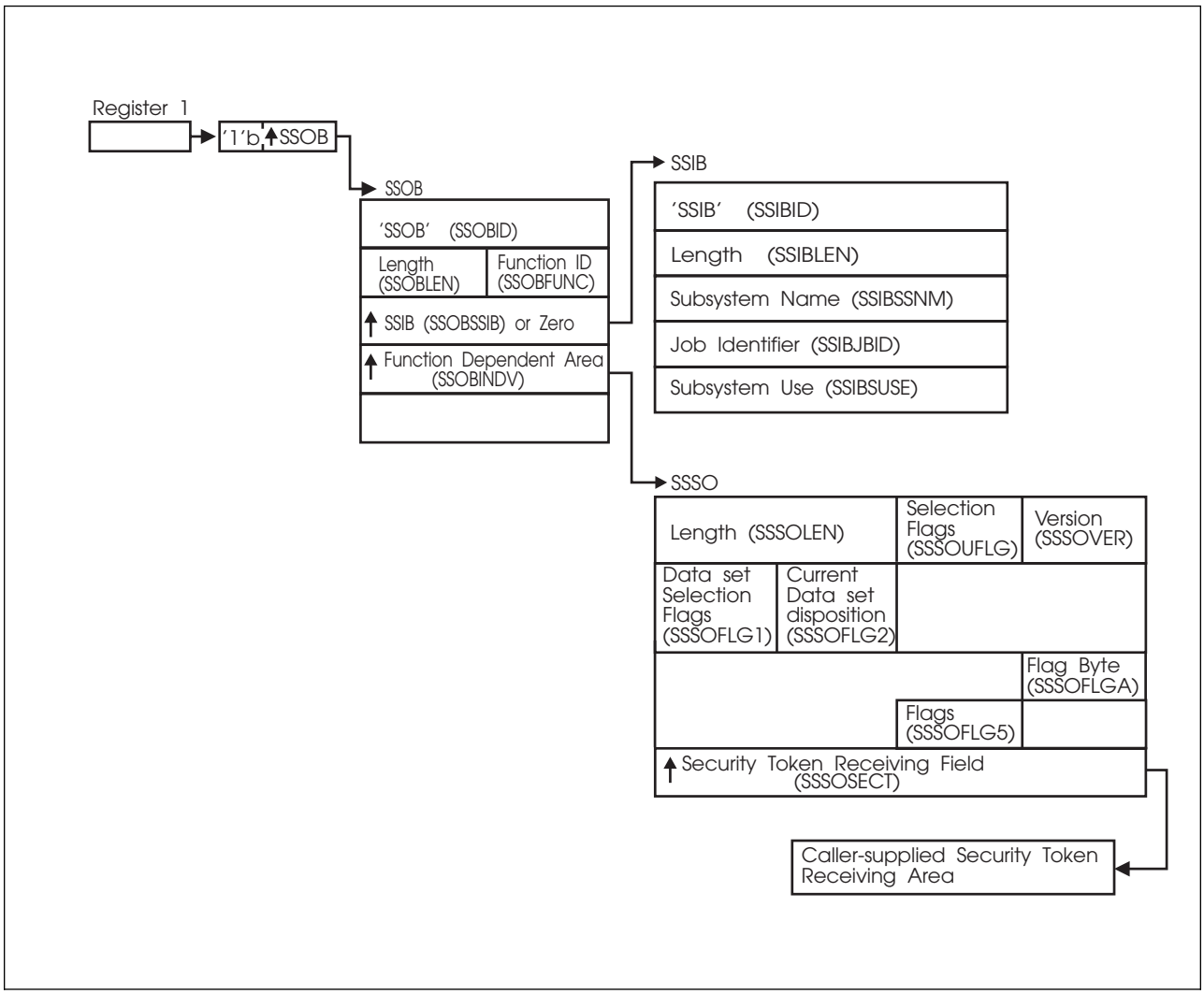

Figure 4. Environment at Time of Call for SSI Function Code 1

Input Register Information

Before issuing the IEFSSREQ macro, your external writer must ensure that the following general purpose registers contain:

Register	Contents
1	Address of a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.
13	Address of a standard 18-word save area.

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSSO

SSOB Contents: Your external writer sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function code 1 (SSOBSOUT)
SSOBSSIB	Address of an SSIB control block or zero (if this field is zero, the life-of-job SSIB is used.) See “Subsystem Identification Block (SSIB)” on page 8 for more information on the life-of-job SSIB.
SSOBINDV	Address of the function dependent area (SSSO control block)

Your external writer must set all other fields in the SSOB control block to binary zeros before issuing the IEFSSREQ macro.

SSIB Contents: If you don't use the life-of-job SSIB, your external writer must provide an SSIB and set the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of the subsystem to which this Process SYSOUT Data Sets call is directed. It is usually the primary JES, or in the case of JES2, a possible secondary JES. If your routine has not been initiated from such a JES, your external writer must issue a Request Job ID call (SSI function code 20) prior to this Process SYSOUT Data Sets call. You must use the same subsystem name in this SSIBSSNM field as you used for the Request Job ID call.
SSIBJBID	Job identifier — the job ID that was returned upon completion of the Request Job ID call (SSI function code 20).
SSIBSUSE	(JES3 only) Subsystem use — the SSIBSUSE value that was returned upon completion of the Request Job ID call (SSI function code 20).

Your external writer must set all other fields in the SSIB control block to binary zeros before issuing the IEFSSREQ macro.

SSSO Contents: Your external writer sets the following fields in the SSSO control block on input:

Field Name	Description
SSSOLEN	Length of the SSSO control block — set with SSSOSIZE value.
SSSOUFLG	User Selection Flags — defines the operation this call performs. The following options are available: **Flag Value is XX'00':** Setting this flag to zero indicates an initial request. Upon issuing the IEFSSREQ service for the SSI function code 1, your external writer should ensure this field is zero.

When the SSSOCTRL bit (in flag byte SSSOFLG2) is zero, JES provides the name of the next data set to be allocated.

Flag Value is non-zero:

For JES3 only, setting this flag to non-zero indicates that the caller performs immediate disposition processing on all data sets matching the other selection criteria (including the data set specified in the SSSODSN field). If your external writer has dynamically allocated a single data set, however, the updates described through the bit settings listed below should be performed through the appropriate dynamic text unit keys only. See "Processing Flow for Single Data Set Requests" on page 29 for more information on performing disposition processing on single data sets.

Your external writer can use one or more of the following bit settings when performing disposition processing on multiple data sets only:

- **SSSOSETC** — (JES3 only) Change the SYSOUT class to the value specified in the SSSOCLAS field.

  This is only valid for JES3 update requests for the selected data sets on the JES3 HOLD queue.

  Bit SSSORLSE might be used concurrently to move the data set from the HOLD queue to the WRITER queue. Information associated with this SYSOUT class is also updated with the JES3 defaults if that SYSOUT class was defined to JES3.

- **SSSODELC** — (JES3 only) Delete the selected data sets.

  This is only valid for JES3 update requests that have data sets on the WRITER or HOLD queue.

- **SSSOROUT** — (JES3 only) Change the destination of the selected data sets to the value specified in the SSSODEST field.

  This is only valid for JES3 update requests, and for the selected data sets on the JES3 HOLD queue.

  The SSSORLSE bit might be used concurrently to move the data set from the HOLD queue to the WRITER queue.

- **SSSOHOLD** — Hold all selected data sets.

  Neither JES2 nor JES3 honors this bit.

- **SSSORLSE** — (JES3 only) Release all selected data sets that are eligible for printing or further processing by JES3.

  This is only valid for JES3 update requests, and for the selected data sets on the JES3 HOLD queue.

  Bits SSSOSETC and SSSOROUT might also be issued concurrently.

SSSOVER Version Number — the current version number. Set with the value of SSSOCVER.

SSSOFLG1 Data set selection flags — determines the data sets the caller wants.

Your external writer can set one or more of the following selection bits:

- **SSSOHLD** — Use HELD data sets in the selection criteria.

  For JES2, do not set this bit. Your external writer selects work for JES2 only if that work is on the OUTPUT queue with an OUTDISP of WRITE or KEEP.

  For JES3, setting this bit allows the external writer to process work from either:

  – JES3 WRITER queue only (if SSSOHLD is off)

  – JES3 WRITER and HOLD queues (HOLD=EXTWTR only) if SSSOHLD is on.

  To ensure your external writer selects work from only the HOLD queue, select a specific SYSOUT class assigned to HOLD=EXTWTR, or the WRITER name (which creates data sets queued only to the HOLD queue). This way work destined for JES3 writers on the OUTPUT queue will not be accidentally allocated or processed.

- **SSSOSCLS** — Use SYSOUT class in a selection criterion.

  Your external writer can set up to eight specific SYSOUT (1-character EBCDIC values A-Z, 0-9) classes in the SSSOCLSL field. These classes are:

  – Selected in priority order

  – Left-justified, and padded to the right with blank (X'40') characters in the SSSOCLSL field.

- **SSSODST** — Use the remote destination in a selection criterion.

  Your external writer specifies the destination in the SSSODEST field.

- **SSSOSDST** — An alternative way for the external writer to specify SSSODST, and has the same equated value as SSSODST.

- **SSSOSJBN** — Use the job name as a selection criterion.

  Your external writer specifies the job name in the SSSOJOBN field.

- **SSSOSJBI** — Use the job ID as a selection criterion.

  Your external writer specifies the job ID in the SSSOJOBI field.

- **SSSOSPGM** — Use the external writer name (the second positional parameter in the SYSOUT= keyword on the DD JCL statement), or userid as a selection criterion. Either value (depending on the bit setting for either SSSOWTRN or SSSOUSER) is stored in the SSSOPGMN field.

- **SSSOSFRM** — Use the form name as a selection criterion.

Set selection bit SSSOSFRM. When using 8-character forms, also set selection bit SSSOSFR8.

1. 4-character form name

   Use the 4-character form name field (SSSOFORM), and set the SSSOSFRM bit. Do not set the 8-character selection bit (SSSOSFR8).

2. 8-character form name

   Use the 8-character form name field (SSSOFOR8), and set both the SSSOSFRM and SSSOSFR8 bits. If using an 8-character form, place the name of the form in the SSSOFOR8 field, and not in the SSSOFORM field.

- **SSSOSFR8** — Use the 8-character form name field (SSSOFOR8) as a selection criterion. Make sure that you do not use the 4-character form name field (SSSOFORM). JES ignores the SSSOFORM field.

  If your external writer sets the this bit, the SSSOSFRM bit must also be set to indicate selection by either 4-character or 8-character forms.

SSSOFLG2 Flag byte

Your external writer can set one or more of the following selection bits:

- **SSSOCTRL** — Processing Completion Flag

  If your external writer sets this bit off, it performs a retrieval request. The next data set name (if selectable by JES) to be processed is returned in the SSSODSN field.

  If your external writer sets this bit on, it has made the last call to JES. Your external writer should only set the SSSOCTRL bit on when ending its address space, so that performance will not be negatively affected by the disassociating of resources (collected by your external writer) in the JES address space. This can include such resources as storage, and queues of control blocks.

  For JES3, your external writer can issue this final IEFSSREQ call only when the SSSOCTRL bit is set on, and when the external writer is ending.

- **SSSOPSEE** — Process SYSOUT extension

  Your external writer sets this bit on if SOEXT=YES was specified on the IEFSSSO macro invocation to indicate that additional fields exist in the IEFSSSO.

  For example, the DDNAME version (proc step name, step name, dd name) of the returned data set is in a field in the process SYSOUT extension.

SSSOJOBN Job name for a retrieval request.

Your external writer:

- Sets the value of the specific job name in the SSSOJOBN

field left-justified, and padded to the right with blank (X'40') characters.

- Sets the SSSOSJBN bit for this selection to occur.

SSSOJOBI Job ID for a retrieval request.

Your external writer:

- Sets the value of the specific job ID in the SSSOJOBI field left-justified, and padded to the right with blank (X'40') characters. Examples of valid job IDs are:
  - 'JOB12345'
  - 'STC12345' or 'TSU01234' (in JES2).
- Sets the SSSOSJBI bit for this selection to occur.

SSSOCLAS (JES3 only) Single character SYSOUT class that the output data sets must be changed to during an update request.

Your external writer sets the SSSOSETC bit for modification to occur.

SSSOFLGA Flag byte containing the SSSOUSER and SSSOWTRN bits.

Your external writer can set either the SSSOUSER bit (userid), or the SSSOWTRN bit (writer name), but not both.

If your external writer sets the SSSOUSER bit, the value contained in the SSSOPGMN field is a userid. Setting the SSSOUSER bit for userid selection allows your external writer to access the data set if both:

- A data set resource profile in the security product (RACF) does not exist to protect it.
- The JESSPOOL security class is active. For information on the JESSPOOL security class, see *OS/390 SecureWay Security Server RACF Security Administrator's Guide*.

If your external writer sets the SSSOWTRN bit, the value contained in the SSSOPGMN field is a writer name. Setting the SSSOWTRN bit for writer name selection allows your external writer to access the data set if both:

- A data set resource profile in the security product (RACF) exists.
- The JESSPOOL security class is active.

Your external writer must set the SSSOSPGM flag bit even if the SSSOUSER or SSSOWTRN bit is set, so that the SSSOPGMN field can be used as a selection criterion.

Note, for JES2 external writers that have the SSSOSPGM bit set on but have not set the SSSOUSER bit or the SSSOWTRN bit, and have set the SSSOPGMN field to all blank (X'40') characters, JES2 returns only the data sets whose userid and writer name are both filled with blank (X'40') characters.

SSSODEST Destination selected for either a retrieval request or an update request.

For a retrieval request, your external writer:

- Sets the value of the specific destination in the SSSODEST field left-justified, and padded to the right with blank (X'40') characters.
- Sets the SSSODST bit (or SSSOSDST) for this selection to occur.

For an update request (JES3 only), your external writer:

- Sets the value of the specific destination in the SSSODEST field left-justified, and padded to the right with blank (X'40') characters.
- Sets the SSSOROUT bit for this selection to occur.

SSSOPGMN Name selected for a retrieval request.

If your external writer set the SSSOWTRN bit in the SSSOFLGA flag byte, this field contains the writer name. Do not use 'NJERDR', 'INTRDR' or 'STDWTR' as the writer name.

If your external writer set the SSSOUSER bit in the SSSOFLGA flag byte, this field contains the userid.

Your external writer:

- Sets the value of the specific writer name or userid in the SSSOPGMN field left-justified, and padded to the right with blank (X'40') characters.
- Sets the SSSOSPGM field for this selection to occur.

Note, for JES2 external writers that have the SSSOSPGM bit set on but have not set the SSSOWTRN bit or the SSSOUSER bit, and have set the SSSOPGMN field to all blank (X'40') characters, JES2 returns only the data sets whose writer name and userid are both filled with blank (X'40') characters.

SSSODSN Data set name

For the initial retrieval request, your external writer sets this field to binary zeros. JES returns the name of a SYSOUT data set in this SSSODSN field.

In a subsequent dynamic allocation, your external writer uses the name of this data set for processing purposes. See “Processing Flow for Single Data Set Requests” on page 29 for more information on this field.

During dynamic unallocation for a single returned data set, operations, such as changing the destination and releasing the data set to print, are performed by using the appropriate dynamic unallocation text unit keys.

For subsequent retrieval requests, your external writer must not change the SSSODSN field.

For an update request (JES3 only, when the SSSOUFLG bit is non-zero and the SSSODSN field is zero), the attributes will be changed for all data sets matching the other selection criteria specified.

SSSOFORM Form selected (4-character specification) for a retrieval request.

Your external writer:

- Sets the value of the form name in the SSSOFORM field left-justified and padded to the right with blank (X'40') characters.
- Sets the SSSOSFRM bit for this selection to occur.

  If the SSSOSFR8 bit is also set, specify the 8-character form name in the SSSOFOR8 field, and this SSSOFORM field is not used.

  If the SSSOSFR8 bit is set, specify the form name in the SSSOFOR8 field, even if the form name is less than 4 characters.

SSSOCLSL SYSOUT class selected for a retrieval request.

Your external writer must also set the SSSOSCLS bit.

This list can contain one to eight SYSOUT classes as a selection criteria. JES processes the list from left to right, so that, if JES finds no data sets using the first character in the list and your external writer specified more than one class, JES searches the next SYSOUT class (if present).

For JES3 only, each new SYSOUT class character causes JES to restart the queue search process. Therefore, for performance considerations, place the most used SYSOUT classes in the front of the list.

The fields that follow from the SSSOFLG5 field through the SSSOFOR8 field are available as input fields only when you specify SOEXT=YES on the IEFSSSO invocation. IBM recommends that you specify SOEXT=YES on the IEFSSSO invocation, as additional information is returned to the external writer.

Field Name **Description**

SSSOFLG5 Flag byte

Your external writer can set one or more of the following selection bits:

- **SSSOTKNR** — Security token length and security token version information set.

  This bit determines whether the caller has supplied the security token length and security token version information in the field pointed to by SSSOSECT. JES provides the security token of the returned data set (mapped to the requested version and length) upon return from the retrieval request. See *OS/390 SecureWay Security Server External Security Interface (RACROUTE) Macro Reference* for more information.

SSSOSECT Address of a caller-supplied area that contains a security token (returned only if your external writer specifies the SSSOUSER bit).

If the SSSOTKNR bit has also been set, your external writer must also provide the length and version of the token that is returned

at the address specified in the SSSOSECT field. JES returns the security token in the format specified. See the SSSOTKNR bit and the SSSOTKNG bit on output for additional information.

If the SSSOTKNR bit has also been set off:

- The returned token is at the current level of the security authorization facility (SAF) security tokens
- The external writer is responsible for providing enough storage for the transfer to be made.

SSSOFOR8 8-character form name selected

Your external writer must have set both the SSSOSFRM and SSSOSFR8 bits for this selection to occur.

This field contains an 8-character form name that is left-justified and padded to the right with blank (X'40') characters.

If the SSSOSFR8 bit is also set, the 4-character form name in SSSOFORM is ignored. JES uses the name in the SSSOFOR8 field as the forms selection criteria.

Your external writer must set all other fields in the SSSO control block to binary zeros before issuing the IEFSSREQ macro.

Output Register Information

When control returns to your external writer, the general purpose registers contain:

Register	Contents
0	Used as a work register by the system
1	Address of the SSOB control block
2 — 13	Same as on entry to call
14	Return address
15	Return code

Return Code Information

The SSI places one of the following decimal return codes in register 15. Examine the return code to determine if the request was processed.

Return Code (Decimal)	Meaning
SSRTOK (0)	The Process SYSOUT Data Sets call completed. Check the SSOBRETN field for specific function information.
SSRTNSUP (4)	The subsystem specified in the SSIBSSNM field does not support this function.
SSRTNTUP (8)	The subsystem specified in the SSIBSSNM field exists, but it is not active.
SSRTNOSS (12)	The subsystem specified in the SSIBSSNM field is not defined to MVS.

SSRTDIST (16) The pointer to the SSOB control block or the SSIB control block is not valid, or the function code specified in the SSOBFUNC field is greater than the maximum number of functions supported by the subsystem specified in the SSIBSSNM field.

SSRTLERR (20) Either the SSIB control block or the SSOB control block has incorrect lengths or formats.

SSRTNSSI(24) The SSI has not been initialized.

Output Parameters

Output parameters for the function routine are:

- SSOBRETN
- SSSO

SSOBRETN Contents: When control returns to your external writer and register 15 contains a zero, the SSOBRETN field contains one of the following decimal values:

Value (Decimal) **Meaning**

SSSORTOK (0) Successful completion.

SSSOEODS (4) There are no more data sets to select with the requested selection criteria.

Your external writer has the following options:

- Wait until new work becomes available.

  See “The Writer Communication Area” on page 34 for information about the ECB that will be posted when work is available. In JES2, this POST only occurs for those external writers that are running as started tasks, and not batch jobs.

- Modify the criteria for new work.

  Your external writer may modify some of the entry criteria (for example, change the form number) to indicate a new selection, and initiate the IEFSSREQ process. Do not issue an IEFSSREQ with the SSSOCTRL bit set when the work is for a different set of characteristics.

- Perform job-level (update) disposition (JES3 only).

  For example, your function may have been leaving data sets on the spool until all the data sets from the job have been completely and successfully processed. Now, the external writer can perform a job-level disposition of delete with a subsequent IEFSSREQ call specifying the job ID.

- End current activity.

  Issue a final IEFSSREQ with the SSSOCTRL bit set. This completely disassociates the external writer from the JES. Perform this final call only when your external writer is ready to end the operation.

SSSONJOB (8) Job not found.

You specified the job name as a selection criterion, but the job name specified in the SSSOJOBN field did not match any job in the system.

SSSOINVA (12) Invalid search argument.

The job ID specified in the SSSOJOBI field failed syntactical parsing, or both the SSSOWTRN bit and the SSSOUSER bit had also been set in the SSSOFLGA flag byte.

SSSODUPJ (20) Duplicate job names

During a retrieval request, more than one job was found matching the name in the SSSOJOBN field. A job ID should be specified as a selection criteria to uniquely identify the job.

SSSOINVJ (24) Invalid job name/job ID combination

During a retrieval request, a job name and job ID were specified as selection criteria, but the job name is not associated with the job ID that the external writer supplied.

SSSOIDST (28) Invalid destination specified in field SSSODEST.

The return code information depends on which JES is being used:

JES2: The supplied destination did not exist in the JES destination routing tables.

JES3: The supplied destination is not syntactically correct (See *OS/390 MVS JCL Reference* for the correct syntax) or a valid NJE destination was supplied (an external writer cannot select work destined for NJE nodes).

SSSOAUTH (32) Authorization failed

(JES3 only) The user exit IATUX30 denied the external writer access to this request.

SSSOTKNM (36) Token map failed

The requested RACROUTE TOKENMAP function failed. JES does not set the SSSOTKNG bit, and no token is provided in the field pointed to by the SSSOSECT field.

SSSO Contents: The SSSO control block contains the following information about the data set returned from your external writer's retrieval request:

Field Name	Description
SSSOFLG2	Flag byte

- **SSSODDST** — DD name set in the extension.

  JES sets this flag upon return from a retrieval request, so that your external writer knows that the SSSOPRCD, SSSOSTPD, and SSSODDND fields have been returned with the three part DDNAME of proc-step name, step name, DDNAME.

SSSOCOPY Number of copies.

JES provides the data set to your external writer as many times as the copy count from the creating JCL specifies it.

The value of this field depends on which JES is being used:

JES2: This field is always set to '1' on a retrieval request.

JES3: This field is always set to '1' on a retrieval request.

SSSOJOBN Job name associated with the returned data set (retrieval request).

SSSOJOBI Job ID associated with the returned data set (retrieval request).

SSSOCLAS Class associated with the returned data set (retrieval request).

If your external writer set the SSSOSCLS bit and the SSSOCLSL field, this class matches a class in the list contained in the SSSOCLSL field (if a multiple class list was specified), or the single class in the SSSOCLSL field (if only one class was specified).

SSSOMLRL Maximum logical record length associated with the returned data set.

For JES3, if the length in the SYSOUT data set was not valid, a zero is returned. If the data set is a system data set, such as JESJCL, then a value of '133' is returned.

SSSODEST Destination associated with the returned data set (retrieval request).

SSSOPGMN Writer name or userid associated with the returned data set (retrieval request, if available).

The specific information returned depends on the setting of the SSSOWTRN or SSSOUSER bits (retrieval request).

JES2: If neither the SSSOWTRN or SSSOUSER bits are specified, then this field contains the writer name associated with this data set.

Note: The SSSOPGMN field is filled in regardless of whether the SSSOSPGM bit is set. It contains a userid only when the SSSOUSER bit is set.

SSSODSN Returned data set name (retrieval request).

Upon return from a retrieval request, your external writer must use this name in the dynamic allocation of the data set. See “Processing Flow for Single Data Set Requests” on page 29 for additional details.

The returned data set name is in the fully-qualified, form of: userid.jobname.jobid.dsnumber.dsname.

SSSOFORM First four characters of the form name associated with the returned data set name (retrieval request). The SSSOFOR8 field contains the 8-character form name.

SSSOWTRC Pointer to a communication area for your external writer for a retrieval request.

This area contains additional information about the:

- Data set
- Owning job
- Wait-for-work ECB.

Your external writer might need to use this information in its processing. See “The Writer Communication Area” on page 34 for more information.

The fields that follow from the SSSOFLG5 field through the SSSOOGNM field are available as output fields only when you specify SOEXT=YES on the IEFSSO invocation. The external writer sets the SSSOPSEE bit. IBM recommends that you specify SOEXT=YES on the IEFSSO invocation, as additional information is returned to the external writer.

Field Name	Description
SSSOFLG5	Flag byte • **SSSOTKNG** — Token mapped. JES sets the SSSOTKNG bit if the token was returned to the version requested by your external writer through the setting of the SSSOTKNR on the retrieval request. SSSOSECT points to the returned token with its new version and length. • **SSSOGNVA** — (JES2 only) Output group name provided in the SSSOOGNM field for a retrieval request.
SSSOLNCT	Line count of the returned data set provided for a retrieval request. The value is correct if the task that created the SYSOUT data set went through end-of-task processing. The line count includes only records with a non-zero text length that have data associated with them. The count does not include records that start with machine immediate control characters. For example, if a 600-line data set is produced with machine carriage control characters and includes one Skip-to-Channel-1-Immediate command every 60 lines, then there would be 610 records in the data set, but field SSSOLNCT would have a value of 600.
SSSOPRCD	Proc step name of the returned data set provided if: • JES set the SSSODDST bit upon return to your external writer. • A retrieval request is being made.
SSSOSTPD	Data set step name of the returned data set provided if: • JES set the SSSODDST bit upon return to your external writer. • A retrieval request is being made.

SSSODDND Data set ddname of the returned data set provided if:

- JES set the SSSODDST bit upon return to your external writer.
- A retrieval request is being made.

SSSOSECT Pointer whose contents remains unchanged from the retrieval request, but whose address now points to the returned data set token provided by SAF, if your external writer has set the SSSOUSER bit.

If your external writer did not set the SSSOTKNR bit, JES copied the token to the address specified in the SSSOSECT field. This copy was performed assuming that the receiving length is as long as the length of a version 1 security token. If your external writer did not allocate enough storage at the address pointed to by the SSSOSECT field, a protection exception might occur.

If both:

- Your external writer set the SSSOTKNR bit to indicate to SAF to return a token with a different version and length on the retrieval request, and JES successfully performed this function.
- JES set the SSSOTKNG bit

The SSSOSECT field points to the token in the correct format.

SSSOFOR8 Form name of the returned data set name for a retrieval request.

SSSOACCT (JES2 only) Address of an accounting string for the returned data set for a retrieval request, or zero.

Your external writer must be in AMODE 31 to access this data. The data is in the following format:

- A 1-byte field containing the number of pairs that follow.
- Zero or more accounting pairs, each of the form:
  - A 1-byte field containing the length of the accounting string.
  - The actual accounting string itself with the length that is specified in the first byte.

    A length of zero indicates an omitted field.

For example, if the original accounting information had been specified as (12,,ABCD), the field pointed to by the SSSOACCT would be: '03 02 F1 F2 00 04 C1 C2 C3 C4' in storage.

SSSOOGNM (JES2 only) JES2 output group name of the returned data set.

The SSSOGNVA flag is set if the field is valid.

Processing Flow for Single Data Set Requests

Your external writer can process single data set requests by:

- (JES2 and JES3) Processing one data set at a time.
- (JES3 only) Processing all data sets together (update request).

Processing One Data Set at a Time (JES2 and JES3): Your external writer can use the following steps for proper selection, allocation, retrieval, and unallocation of an individual SYSOUT data set:

1. Build the appropriate SSOB and SSSO control blocks for the request according to the individual selection criteria desired.
2. Issue the IEFSSREQ macro asking JES for the name of a data set.

   This step includes setting the SSSOUFLG flag byte to X'00', and the SSSOCTRL bit to 0.

   Upon return the name of the data set is in the SSSODSN field.
3. Allocate the data set through dynamic allocation (DYNALLOC macro).

   Your external writer can use the following text units:

   - DALDSNAM

     Used with the returned name from the SSSODSN field.
   - DALSSREQ

     Indicates a request that JES needs to handle. The parameter value in this text unit is the name of the subsystem that processed the IEFSSREQ macro.
   - DALRTDDN

     Indicates the DDNAME associated with the allocation be returned to the caller of DYNALLOC. Your external writer then places this DDNAME in the DCB macro that needs to open the SYSOUT data set as input for your reads. Prime the parameter in this text unit with blank (X'40') characters before issuing the DYNALLOC macro. This text unit is returned from DYNALLOC with the correct DDNAME.

     Your external writer will also use this DDNAME in the dynamic unallocation of the data set when performing unallocation processing.
4. Open the program-supplied DCB.

   Move the returned DDNAME from the DALRTDDN field as the DCB's DDNAME before issuing the OPEN.

   The following is an example of a DCB that may be used to obtain the records:

   ```
   INDCB    DCB   DSORG=PS,MACRF=GL,BUFNO=1,                              X

                  SYNAD=some-routine,EODAD=some-routine
   ```

   Note: Multiple QSAM buffers here do not improve performance. IBM recommends BUFNO=1.

   Your program can issue BSAM and QSAM macros in 31-bit mode. See *OS/390 DFSMS Macro Instructions for Data Sets*.
5. Optionally open any other devices that the program requires.
6. Access the records in the SYSOUT data set.
7. Upon EODAD, close the input DCB and issue the FREEPOOL macro unless you coded RMODES31=BUFF on the DCBE macro.
8. Unallocate the data set through dynamic allocation (DYNALLOC).

Optionally, you can perform disposition processing to change the attributes of the returned data set.

The specific text units to be used are:

- DUNDDNAM

  This text unit indicates an unallocation by DDNAME. The DDNAME the external writer must use is the same one used for the data set allocation.

- DUNOVDSP

  This text unit indicates a disposition override. You must specify one of the following:

  – Keep the data set on the spool. For JES2, when you specify keep as the disposition, JES2 assumes that the external writer has failed and treats the next PSO request as if you had set the SSSOCTRL bit.

  – Delete the data set from the spool.

  If you are performing immediate disposition and wish to delete the data set, use the X'04' value as the disposition flag. Otherwise, you can use the X'08' value to keep the data set on the spool.

Optionally you may use any of the following text units to modify the queue, change the destination, or change the SYSOUT class of the data set during unallocation.

In JES3, the only queue modification you can make is moving the data set from the HOLD queue to the WRITER queue.

- DUNOVSNH

  For JES2, the data set selected is already on the output queue with a disposition of WRITE or KEEP, and this text unit is not specifiable.

  For JES3, this text unit removes the data set from the HOLD queue, and puts it on the WRITER queue.

- DUNOVCLS

  For JES3, this text unit changes the SYSOUT class of the data set on the HOLD queue.

- DUNOVSUS

  For JES3, this text unit overrides the destination of the SYSOUT data set, and can be used to route SYSOUT to another destination.

9. Either issue the IEFSSREQ macro again for another data set, or issue the IEFSSREQ macro for a final call (the SSSOCTRL bit is set), to disassociate the program from JES.

Processing All Data Sets Together (JES3 only - update request): Your external writer performs the actions specified in the SSSO control block in all data sets matching the selection criteria in the SSSO control block, when the IEFSSREQ macro is issued with a non-zero SSSOUFLG flag byte. Individual data set names are not returned in this case.

The SSSODSN field can be zero if more than one data set matching the other selection criteria is modified. Any previously allocated single data sets must be unallocated, however, before this update request is made.

External Writer Considerations

A standard external writer is designed to request work and perform disposition processing of work to each JES in the following ways:

- It is initiated from the user's address space

  Therefore, it is a completely separate MVS job. This separation allows for processing overlap and address space integrity. In JES3, because the SSI is involved for scheduling communication, the external writers may exist on local processors as well as the global processor.

- It is functionally independent of JES

  There is neither a print processor running in the JES2 address space, nor a writer DSP running in the JES3 global address space.

- It is not automatically started by JES

  MVS does not supply an automatic facility to create this address space. If the external writer is running as a started task, you can use an operator START command to create this address space or you can submit a batch job. Your application (external writer) makes this decision. Your external writer should also have a mechanism to end itself.

- It may drive a non-JES supported device

  This is the primary purpose of the external writer. If the SYSOUT data set deals with plotting, for instance, a special code in the data may indicate to use the red pen instead of the blue pen. Your external writer can recognize this code as a control sequence, and perform the appropriate actions according to the output device.

- It allows the installation to control the selection of work

  Standard external writers select work through a SYSOUT class dedicated to external writers or a writer name. JES2 and JES3 handle external writer processing differently.

  JES2: The work to be processed is located on the output queue, and has an OUTDISP of WRITE or KEEP. However, conversational data sets, which include data sets located on the output queue with an OUTDISP of HOLD or LEAVE, are not processed in JES2 by the standard external writer. These data sets are destined to be processed by TSO/E users through the OUTPUT command.

  JES3: The work to be processed is located on either the WRITER queue, or the HOLD queue. However, IBM recommends that you process only data sets on the HOLD queue (either by specific SYSOUT class specification as defined on the initialization statement, or by writer name).

  Note: Work destined for TSO/E users (through HOLD=TSO on the specific SYSOUT class initialization statement) is not processed because those data sets are destined to be processed by TSO/E users through the OUTPUT command.

- It does not handle simultaneous multi-tasking within an address space

  The external writer facility in JES does not support concurrent subtasking of work. Unpredictable results will occur if attempted. Once an external writer begins the IEFSSREQ process the first time, calls through the IEFSSREQ are not allowed from any other subtask in the same address space until the first subtask has finished issuing its final call through (SSSOCTRL) IEFSSREQ.

- It interacts with JES by requesting work

  The external writer makes a request of JES for work by using the selection criteria, and then uses dynamic allocation to allocate a returned SYSOUT data set for processing.

- It handles retrieval requests

  Both JES2 and JES3 support retrieval requests. That is, the external writer issues the IEFSSREQ macro asking JES to supply the name of a selectable SYSOUT data set. The external writer processes that data set through dynamic allocation. See “Processing Flow for Single Data Set Requests” on page 29 for more information on the processing flow.

  Updates to selected attributes for a particular data set (such as destination and class change) can be made through the unallocation facility as described within this documentation.

- It handles update requests

  An update request is allowable only for JES3.

  JES3 allows update requests through the IEFSSREQ macro for one or more data sets whose selection criteria matches the criteria supplied by the external writer directly through the IEFSSREQ macro.

  However, individual data sets obtained through the IEFSSREQ retrieval/allocation process should have their attributes changed during the dynamic unallocation as described in the retrieval information above.

  Update requests may be performed on more than one data set at a time when the external writer:

  – Issues the IEFSSREQ macro

  – Does not specify a specific data set name within the SSSO control block.

  This is a powerful facility. However, you should be careful when using it, as the scope of such a modification may be large when more than one data set is involved.

- It uses MVS services to communicate to JES

  SSI function code 1 schedules work by allowing the external writer to indicate which types of data sets it wishes to process and then asking JES to return the name of a SYSOUT data set to the external writer. Dynamic allocation of this spooled data set is performed through dynamic allocation (DYNALLOC). The records of the spooled data set may be obtained through sequential access methods (SAM GETs). A dynamic unallocation is used to deallocate the SYSOUT data set (upon EODAD), which optionally changes some of its attributes.

- Spool access is provided by sequential access methods

  SAM is used to obtain the records of the SYSOUT data set from the spool. This implies familiar coding techniques, such as OPENs, GETs, and CLOSEs.

- It handles all data record processing

  Once a record is supplied to the external writer on a GET, the external writer has control of the record. For example, it can print the record or archive the record, depending on the purpose of the external writer.

- It may wait for JES to post it for new work if idle

When JES sends a no-work-available notice to the external writer, it may sit idle until it receives a POST from JES, telling it that work is available. It may then ask JES again for the newly available work.

This process uses WAIT and POST logic with an ECB returned to the external writer.

JES2 does not POST the external writer if invoked from a batch job; it must be a started task for such posting to occur.

The Writer Communication Area

On return from the IEFSSREQ macro, the SSSOWTRC field contains a pointer to the writer communication area, a series of fields in storage.

The first field in this area is a wait-for-work ECB that JES posts when work becomes available and an SSSOEODS return was previously issued. If you had received an SSSOEODS return, you could wait on this fullword and then retry your request (another IEFSSREQ macro).

All of the fields following the first fullword contain data about the data set returned during retrieval requests, and are contiguous in storage.

Writer Communication Area Contents: The fields in the writer communication area contain:

- Wait-for-work ECB (described earlier).

  Length of 4 bytes.

- Start time of the job creating the SYSOUT data set returned. The format is from the TIME macro with BIN specified.

  Length of 4 bytes.

- Start date of the job creating the SYSOUT data set returned, in packed decimal form where F is the sign: 0cyydddF.

  Length of 4 bytes.

- Owner of the output created by the job. This is the userid associated with the job that created the SYSOUT.

  Length of 8 bytes.

Example

The following is a coded example of a program that generates a Process SYSOUT Data Set call. It requests a SYSOUT data set from JES through a writer name and reads each record of the data set. When the routine reaches the end of the data, the SYSOUT data set is deallocated and the SYSOUT class and destination are updated. The routine ends and cycles back to the beginning to ask JES for the next data set.

This routine is non-reentrant, and must reside below 16 megabytes in an APF-authorized library.

```
SSIREQ01 TITLE '- DOCUMENTATION'
SSIREQ01 AMODE 31
SSIREQ01 RMODE 24
         SPLEVEL SET=4
*********************************************************************
* FUNCTION:  THIS PROGRAM PERFORMS THE FOLLOWING FUNCTIONS:         *
*                                                                   *
*        1. REQUESTS A SYSOUT DATA SET FROM JES THROUGH A WRITER    *
*           NAME (SHOWS AN EXAMPLE OF USING ONE OF THE AVAILABLE    *
*           SELECTION CRITERIA TO INFLUENCE WHICH SYSOUT DATA SET   *
*           IS SELECTED).  THIS PROGRAM IS INTENDED TO RUN ON JES3  *
*           ONLY, AS IT SHOWS SELECTION CRITERIA AVAILABLE ONLY TO  *
*           JES3.  (SPECIFICALLY, BIT SSSOHLD IS USED.)             *
*        2. IF ONE IS NOT AVAILABLE, THE OPERATOR CAN WAIT UNTIL    *
*           ONE IS AVAILABLE, OR EXIT THE PROGRAM.                  *
*        3. IF ONE IS AVAILABLE, IT IS DYNAMICALLY ALLOCATED.       *
*        4. EACH RECORD IS READ AND DISPLAYED TO THE OPERATOR.      *
*        5. UPON END-OF-DATA, THE SYSOUT DATA SET IS DEALLOCATED.   *
*           THE SYSOUT CLASS IS CHANGED TO 'A', AND THE             *
*           DESTINATION IS CHANGED TO 'PRT803'.                     *
*           (SHOWS AN EXAMPLE OF USING THE AVAILABLE DYNAMIC        *
*           ALLOCATION TEXT UNIT TO CHANGE THE ATTRIBUTES OF THE    *
*           RECEIVE SYSOUT DATA SET DURING UNALLOCATION.)           *
*        6. THE PROGRAM THEN CYCLES BACK AND ASKS JES FOR THE NEXT  *
*           DATA SET (GOES TO STEP 1).                              *
*                                                                   *
* NAME OF MODULE:  SSIREQ01                                         *
*                                                                   *
* REGISTER USE:                                                    *
*                                                                   *
*             0                  PARM REGISTER                    *
*             1                  PARM REGISTER                    *
*             2                  SSOB                             *
*             3                  SSSO                             *
*             4                  DCB                              *
*             5                  RB                               *
*             6                  MAX RECORD LENGTH                *
*             7                  DUMP CODE                        *
*             8                  ABEND VALUE REGISTER             *
*             9                  IEFSSREQ RETURN CODES            *
*             10                 BASE REGISTER                    *
*             11                 TEXT RECORD STRUCTURE PTR        *
*             12                 UNUSED                           *
*             13                 SAVE AREA CHAIN REGISTER         *
*             14                 PARM REGISTER / RETURN ADDR      *
*             15                 PARM REGISTER / COND CODE        *
*                                                                   *
* ATTRIBUTES:  SUPERVISOR STATE, AMODE(31), RMODE(24)               *
*                                                                   *
```

```
*                                                                     *
* NOTE:  THIS IS A SAMPLE.                                            *
***********************************************************************
         TITLE '- EQUATES'
***********************************************************************
*        GENERAL EQUATES                                              *
***********************************************************************
EQUHOBON EQU   X'80000000'         HIGH ORDER BIT ON
FF       EQU   X'FF'               ALL BITS ON IN A BYTE
***********************************************************************
*        AFTER COMPARE INSTRUCTIONS                                   *
***********************************************************************
GT       EQU   2                   A HIGH
LT       EQU   4                   A LOW
NE       EQU   7                   A NOT EQUAL B
EQ       EQU   8                   A EQUAL B
GE       EQU   11                  A NOT LOW
LE       EQU   13                  A NOT HIGH
*
***********************************************************************
*        AFTER ARITHMETIC INSTRUCTIONS                                *
***********************************************************************
OV       EQU   1                   OVERFLOW
PLUS     EQU   2                   PLUS
MINUS    EQU   4                   MINUS
NZERO    EQU   7                   NOT ZERO
ZERO     EQU   8                   ZERO
ZEROS    EQU   8                   ZERO
NMINUS   EQU   11                  NOT MINUS
NOV      EQU   12                  NOT OVERFLOW
NPLUS    EQU   13                  NOT PLUS
*
***********************************************************************
*        AFTER TEST UNDER MASK INSTRUCTIONS                           *
***********************************************************************
ALLON    EQU   1                   ALL ON
MIXED    EQU   4                   MIXED
NALLOFF  EQU   5                   ALLON+MIXED
ALLOFF   EQU   8                   ALL OFF
NALLON   EQU   12                  ALLOFF+MIXED
***********************************************************************
*        ABEND CODE INDICATIONS                                       *
***********************************************************************
BADR15   EQU   1                   IEFSSREQ R15 NON-ZERO
BADRETN  EQU   2                   SSOBRETN NON-ZERO AND NOT 8
BADS99A  EQU   3                   DYNALLOC ALLOC FAILED
BADOPEN  EQU   4                   OPEN DCB FAILED
BADS99U  EQU   5                   DYNALLOC UNALLC FAILED
BADRLEN  EQU   6                   PSO DATASET TOO LARGE (RECLEN)
```

```
*********************************************************************
*        GENERAL PURPOSE REGISTERS                                  *
*********************************************************************
R0       EQU   0                   PARM REGISTER
R1       EQU   1                   PARM REGISTER
R2       EQU   2                   SSOB
R3       EQU   3                   SSSO
R4       EQU   4                   DCB
R5       EQU   5                   RB
R6       EQU   6                   MAX RECORD LENGTH
R7       EQU   7                   DUMP CODE
R8       EQU   8                   ABEND VALUE REGISTER
R9       EQU   9                   RETURN CODES OR REASONS
R10      EQU   10                  BASE REGISTER
R11      EQU   11                  TEXT RECORD STRUCTURE PTR
R12      EQU   12                  UNUSED
R13      EQU   13                  SAVE AREA CHAIN REGISTER
R14      EQU   14                  PARM REGISTER / RETURN ADDR
R15      EQU   15                  PARM REGISTER / COND CODE
         TITLE '- CVT - COMMUNICATIONS VECTOR TABLE'
         CVT DSECT=YES,LIST=NO
         TITLE 'DCBD'
         DCBD  DSORG=PS
         TITLE '- IEFJESCT - JES CONTROL TABLE'
         IEFJESCT TYPE=DSECT
         TITLE '- SSOB'
         IEFSSOBH
SSOBGN   EQU   *                   START OF FUNCTIONAL EXTENSION
         TITLE '- SSSO'
         IEFSSSO SOEXT=YES
         TITLE '- IEFZB4D0 - SVC99 DSECTS'
         IEFZB4D0
         TITLE '- IEFZB4D2 - TU KEYS'
         IEFZB4D2
*********************************************************************
*  HOUSEKEEPING                                                     *
*********************************************************************
SSIREQ01 CSECT
         SAVE  (14,12)             FORM ID
         BALR  R10,0               ESTABLISH BASE REG
         USING *,R10               INFORM ASSEMBLER
         LA    R2,SA               CHAIN SAVEAREAS
         ST    R13,4(R2)           OLD IN NEW
         ST    R2,8(R13)           NEW IN OLD
         LR    R13,R2              RECHAIN THE SAVE AREAS
         TITLE '- PROCESS SYSOUT'
         WTO   'SSI CODE 01 Version 1'  LET OP KNOW WHAT LEVEL
         STORAGE OBTAIN,           GET STORAGE FOR SSOB/SSSO
               LENGTH=SSOBLEN1,
               COND=NO
         LR    R2,R1               SAVE BEGINNING OF STORAGE
         USING SSOBEGIN,R2         INFORM ASSEMBLER
         LA    R3,SSOBGN           PT TO BEGINNING OF SSSO
         USING SSSOBGN,R3          INFORM ASSEMBLER
         TITLE '- SSOB PROCESSING'
```

```
*********************************************************************
* NOW WORK ON THE SSOB.  THE LIFE-OF-JOB IS USED HERE, SO THE       *
* SSOBSSIB IS ZERO.                                                 *
*********************************************************************
         XC    SSOB(SSOBHSIZ),SSOB CLEAR THE SSOB
         MVC   SSOBID,=CL4'SSOB'   SSOB INITIALS INTO SSOB
         MVC   SSOBFUNC,=AL2(SSOBSOUT) MOVE FUNCTION ID INTO SSOB
         MVC   SSOBLEN,=AL2(SSOBHSIZ) MOVE SIZE INTO SSOB
         ST    R3,SSOBINDV         SAVE THE SSSO ADDRESS
         TITLE '- SSSO PROCESSING'
*********************************************************************
* NOW WORK ON THE SSSO.  SELECT A SELECTION CRITERIA BASES ON       *
* AN EXTERNAL WRITER NAME OF 'ANDREW'.                              *
*********************************************************************
         XC    SSSOBGN(SSSOSIZE),SSSOBGN  CLEAR THE SSSO
         MVC   SSSOLEN,=AL2(SSSOSIZE)     SET THE SIZE OF THE SSSO
         MVI   SSSOVER,SSSOCVER    SET THE VERSION NUMBER
         OI    SSSOFLG1,SSSOSPGM+SSSOHLD   SELECT BY WRITER NAME AND
*                                 THE HOLD QUEUE
         OI    SSSOFLGA,SSSOWTRN   IND. THAT SELECTION IS BY
*                                 WRITER NAME, NOT USERID
         MVC   SSSOPGMN,=CL8'ANDREW'  IND. CORRECT WRITER NAME
*                                    THAT IS USED AS SELECTION
         OI    SSSOFLG2,SSSOPSEE   IND. LONG FORM OF IEFSSSO
*********************************************************************
* NOW GO TAP JES ON THE SHOULDER FOR A DATASET!                     *
*********************************************************************
NEXTDS   DS    0H                  GET NEXT DSNAME FROM JES
         MODESET MODE=SUP          GET INTO SUPERVISOR STATE
         LR    R1,R2               R1=ADDRESS OF SSOB
         O     R1,=A(EQUHOBON)     TURN ON THE HIGH-ORDER BIT
         ST    R1,MYSSOBPT         SAVE POINTER FOR SSREQ
         LA    R1,MYSSOBPT         POINT TO SSOB POINTER
         IEFSSREQ ,                GO TO JES FOR A DATASET
         MODESET MODE=PROB         BACK TO PROBLEM STATE
         LA    R8,BADR15           ASSUME BAD REG 15 RETURN
         LTR   R9,R15              DID THE IEFSSREQ WORK OK?
         BC    NZERO,ABEND         NOT GOOD...TAKE AN ABEND
         LA    R8,BADRETN          ASSUME BAD SSOBRETN
         ICM   R9,B'1111',SSOBRETN CHECK OUT SSOBRETN
         BC    NZERO,TESTIT        NON-ZERO, INVESTIGATE FURTHER
*********************************************************************
* WE HAVE A DATA SET.  NOW DYNAMICALLY ALLOCATE IT, READ AND DISPLAY*
* THE RECORDS USING SEQUENTIAL ACCESS METHOD AS EXAMPLE OF HOW TO   *
* RETRIEVE THE DATA.                                                *
```

```
*********************************************************************
         TITLE '- ALLOCATE RETURNED DATASET'
*********************************************************************
* ALLOCATE THE RETURNED SYSOUT DATASET                              *
*********************************************************************
         LA    R8,BADRLEN          ASSUME SIZE TOO LARGE FOR WTO
         SR    R6,R6               CLEAR REG 6
         ICM   R6,B'0011',SSSOMLRL GET MAX RECORD LENGTH
         CH    R6,=H'150'          IS MAX RCD LENGTH>150??
         BC    GT,ABEND            YES - TIME FOR US TO GO HOME
         STH   R6,RECLEN           SAVE MAX RECORD LENGTH
         LA    R5,MY99RB           PT TO RB
         USING S99RB,R5            ADDRESSABILITY TO THE RB
         XC    S99RB(RBLEN),S99RB  ZERO THE RB
         MVI   S99RBLN,RBLEN       RB LENGTH
         MVI   S99VERB,S99VRBAL    RB VERB CODE=ALLOC
         LA    R1,MY99TPTA         ADDR SVC 99 ALLOC TU PTRS
         ST    R1,S99TXTPP         STORED IN RB
         LA    R1,MY99RBPT         PT TO RB POINTER
         MVC   TXTDSNAM,SSSODSN    MOVE DATASET NAME TO BE ALLOCATED
         DYNALLOC                  ISSUE DYNAMIC ALLOCATION
         LA    R8,BADS99A          ASSUME IT DIDN'T WORK
         LR    R9,R1               COPY FOR DUMP
         LTR   R15,R15             SVC 99 WORK OKAY??
         BC    NZERO,ABEND         NO, TAKE A DUMP
*********************************************************************
* SYSOUT DATASET ALLOCATED OKAY.   MOVE RETURNED DDNAME INTO        *
* THE DCB PRIOR TO OPENING IT.                                      *
*********************************************************************
         LA    R4,INDCB            PT TO THE INPUT DCB
         USING IHADCB,R4           ADDRESSABILITY
         MVC   DCBDDNAM(8),TXTDDA99 MOVE IN RETURNED DDNAME
         MVC   TXTDDU99,TXTDDA99   SAVE FOR UNALLOCATION
         MVC   DCBLRECL,SSSOMLRL   MOVE MAX LENGTH RECORD IN
*                                                                   *
         OPEN  INDCB               OPEN THE DCB
         LA    R8,BADOPEN          ASSUME THE OPEN FAILED
         LR    R9,R4               COPY FOR DUMP
         TM    DCBOFLGS,DCBOFOPN   DID IT WORK?
         BC    ALLOFF,ABEND        NOPE, TAKE A DUMP
         TITLE '- GET THE RECORDS - DISPLAY TO PROGRAM'
GETNEXT  DS    0H                  LOOP FOR READING/DISPLAYING
```

```
*********************************************************************
* SWITCH TO 24 BIT MODE FOR GET MACRO                               *
*********************************************************************
         LA    R15,SSITO24         SWITCH TO 24 BIT MODE ...
         BSM   0,R15               ... FOR RESTRICTED MACRO
SSITO24  DS    0H
         GET   INDCB               R1==> RECORD AFTER THE GET
         L     R15,SSITO31A        RETURN TO 31 BIT MODE ...
         BSM   0,R15               ... AND CONTINUE
SSITO31A DC    A(SSITO31+EQUHOBON) FOR MODE SWITCHING
*********************************************************************
* RETURN TO 31 BIT MODE AND CONTINUE                                *
*********************************************************************
SSITO31  DS    0H
         EX    R6,MOVEIT           MOVE UP TO 150 BYTES OF REC
         LA    R11,RECLEN          POINT TO RECORD FOR OUTPUT
         WTO   TEXT=(11),ROUTCDE=11 DISPLAY TO JOBLOG
         MVI   RECTEXT,C' '        CLEAR RECORD OUT...
         MVC   RECTEXT+1(L'RECTEXT-1),RECTEXT  ..FOR NEXT ONE
         B     GETNEXT             GO GET NEXT RECORD
         TITLE '- EODAD ROUTINE'
MYEODAD  DS    0H                  END-OF-DATASET
         CLOSE INDCB               CLOSE THE INPUT DCB
         DROP  R4                  IHADCB
*********************************************************************
* UNALLOCATE THE SYSOUT DATASET, CHANGING CLASS + DESTINATION       *
*********************************************************************
         XC    S99RB(RBLEN),S99RB  ZERO THE RB
         MVI   S99RBLN,RBLEN       RB LENGTH
         MVI   S99VERB,S99VRBUN    RB VERB CODE=UNALLOC
         LA    R1,MY99TPTU         ADDR SVC 99 ALLOC TU PTRS
         ST    R1,S99TXTPP         STORED IN RB
         LA    R1,MY99RBPT         PT TO RB POINTER
         DYNALLOC                  ISSUE DYNAMIC UNALLOCATION
         LA    R8,BADS99U          ASSUME IT DIDN'T WORK
         LR    R9,R1               COPY FOR DUMP
         LTR   R15,R15             SVC 99 WORK OKAY??
         BC    NZERO,ABEND         NO, TAKE A DUMP
         B     NEXTDS              GO GET NEXT DATA SET
         TITLE '- BAD RETURN FROM IEFSSREQ'
TESTIT   DS    0H
*********************************************************************
*  R8 HAS THE 'BADRETN' ASSUMPTION VALUE FOR POSSIBLE ABEND.        *
*  R9 HAS A NON-ZERO VALUE FROM SSOBRETN FROM THE IEFSSREQ.         *
*********************************************************************
         CH    R9,NOMORE           END OF DATA SET RETURN?
         BC    NE,ABEND            NOPE - QUIT!
*********************************************************************
*  WE RECEIVED THE END-OF-DATA CONDITION.  ASK WHETHER WE           *
*  SHOULD WAIT ON RETURNED ECB, OR COMPLETE NOW,                    *
*********************************************************************
```

```
         XC    MYECB,MYECB         CLEAR THE ECB
         WTOR  'ENTER 'W' OR WAIT, ANYTHING ELSE TO EXIT',
               MYREPLY,
               1,
               MYECB
         WAIT  ECB=MYECB
         OI    MYREPLY,C' '        FORCE REPLY TO UPPER CASE
         CLI   MYREPLY,C'W'        SHOULD WE WAIT?
         BC    NE,EXIT             NO, EXIT
*********************************************************************
* WAIT INDICATED.  SET UP WAIT ON THE RETURNED ECB.                 *
*********************************************************************
         MODESET KEY=ZERO          GET INTO KEY 0
         L     R1,SSSOWTRC         POINT TO RETURNED DATA AREA
         WAIT  ECB=(1)             R1==>RETURNED WAIT-FOR ECB
         MODESET KEY=NZERO         BACK TO ORIGINAL
         B     NEXTDS              WE'RE POSTED - GO GET IT!
         TITLE '- CLOSE OUT ROUTINES'
EXIT     DS    0H                  FINAL CALL, RETURN TO MVS
         MVI   SSSOFLG2,SSSOCTRL   IND. FINAL CALL TO JES
         MODESET MODE=SUP          GET INTO SUPERVISOR STATE
         LA    R1,MYSSOBPT         POINT TO SSOB POINTER
         IEFSSREQ ,                GO TO JES FOR GIVE BACK
         MODESET MODE=PROB         BACK TO PROBLEM STATE....
         STORAGE RELEASE,          FREE SSOB/SSSO
               LENGTH=SSOBLEN1,
               ADDR=(R2)           HERE'S WHERE IT LIVES
         L     R13,4(,R13)         OLD SA PTR
         RETURN (14,12),RC=0       BACK TO MVS
         TITLE '- ABEND ROUTINES'
*********************************************************************
* THIS IS THE ABEND ROUTINE.  R8 CONTAINS THE PROGRAM REASON CODE,  *
* R9 CONTAINS SPECIFIC ERROR/REASON CODE AS RETURNED BY THE         *
* SERVICE ROUTINE.                                                  *
*********************************************************************
ABEND    DS    0H                  ISSUE THE ABEND MACRO
         ABEND (8),DUMP,STEP       TAKE A DUMP IF WANTED
         TITLE '- DATA AREAS'
SA       DS    9D                  SAVE AREAS
MYECB    DS    F                   DOUBLEWORD FOR WTOR
*
MYREPLY  DS    CL1                 REPLY AREA FOR WTORS
RESRV    DS    XL3                 ROUND TO FULL WORD
         TITLE '- DYNALLOC DATA'
*********************************************************************
* THE FOLLOWING CONTROL BLOCKS ARE FOR DYNAMIC ALLOCATION AND       *
* UNALLOCATION.                                                     *
*********************************************************************
```

```
* S99 REQUEST BLOCK POINTER                                          *
*********************************************************************
MY99RBPT DC    A(EQUHOBON+MY99RB)  S99 RB PTR
*********************************************************************
* S99 REQUEST BLOCK                                                  *
*********************************************************************
MY99RB   DS    CL(RBLEN)           MY SVC 99 RB
RBLEN    EQU   (S99RBEND-S99RB)    LENGTH OF RB FOR MY99RB
*********************************************************************
* TEXT UNIT POINTERS FOR ALLOCATION                                  *
*********************************************************************
MY99TPTA DC    A(TXTDALDS)         TU FOR DATASET NAME
         DC    A(TXTSSREQ)         NAME OF SUBSYSTEM TU PTR
         DC    A(EQUHOBON+TXTRTDDN) RETURN DD NAME TU
*********************************************************************
* TEXT UNIT POINTERS FOR UNALLOCATION                                *
*********************************************************************
MY99TPTU DC    A(TXTDUNDD)         TU FOR UNALLOC BY DDNAME
         DC    A(TXTDUNNH)         NOHOLD TU
         DC    A(TXTDUNCL)         CHANGE THE CLASS TU
         DC    A(EQUHOBON+TXTDUNDS) CHANGE THE DEST TU
*********************************************************************
* TEXT UNITS FOR ALLOCATION                                          *
*********************************************************************
TXTDALDS DC    AL2(DALDSNAM)       DATASET NAME KEY
         DC    X'0001'             NUMBER
         DC    AL2(44)             DSNAME LENGTH
TXTDSNAM DS    CL44' '             DSNAME FROM IEFSSREQ
TXTCLOSE DC    AL2(DALCLOSE)       UNALLOCATE AT CLOSE KEY
         DC    X'0000'             # FIELD (0000 REQUIRED)
TXTSSREQ DC    AL2(DALSSREQ)       REQUEST OF SUBSYSTEM
         DC    X'0001'             # FIELD (0001 REQUIRED)
         DC    X'0004'             LEN OF SS NAME FOLLOWING
         DC    CL4'JES3'           NAME OF SUBSYSTEM
TXTRTDDN DC    AL2(DALRTDDN)       RETURN DDNAME FIELD
         DC    X'0001'             # FIELD (0001 REQUIRED)
         DC    X'0008'             LEN OF PARM
TXTDDA99 DC    CL8' '              RETURNED DDNAME PARM FIELD
```

```
*********************************************************************
* TEXT UNITS FOR UNALLOCATION                                       *
*********************************************************************
TXTDUNDD DC    AL2(DUNDDNAM)       TU FOR DDNAME UNALLOC
         DC    X'0001'             NUMBER
         DC    AL2(8)              DDNAME LENGTH
TXTDDU99 DS    CL8' '              DDNAME FROM DYNALLOC
TXTDUNNH DC    AL2(DUNOVSNH)       TU FOR NOHOLD
         DC    X'0000'             # FIELD (0000 REQUIRED)
TXTDUNCL DC    AL2(DUNOVCLS)       TU FOR CHANGE OF CLASS
         DC    X'0001'             # FIELD (0001 REQUIRED)
         DC    X'0001'             LEN OF SYSOUT CLASS
         DC    CL1'A'              CHANGED SYSOUT CLASS
TXTDUNDS DC    AL2(DUNOVSUS)       TU FOR CHANGE OF REMOTE
         DC    X'0001'             # FIELD (0001 REQUIRED)
         DC    X'0008'             LEN OF CHANGED REMOTE
         DC    CL8'PRT803'         CHANGED REMOTE NAME
MYSSOBPT DS    F                   POINTER TO SSOB FOR IEFSSREQ
NOMORE   DC    AL2(SSSOEODS)       NO MORE DATASETS FROM JES
MOVEIT   MVC   RECTEXT(*-*),0(R1)  OBJ OF AN EXECUTE
RECLEN   DS    H                   LENGTH OF OUTPUT RECORD
RECTEXT  DS    CL150               UP TO 150 BYTES OF SYSOUT
INDCB    DCB   DSORG=PS,MACRF=GL,BUFNO=2,EODAD=MYEODAD,               X
               DDNAME=WILLCHNG
         TITLE '- LITERALS'
         LTORG ,
         END
```

Verify Subsystem Function Call — SSI Function Code 15

The Verify Subsystem Function call (SSI function code 15) allows a user-supplied program to:

- Verify the existence of a specific subsystem
- Obtain the address of the SSCVT that corresponds to a specific subsystem
- Obtain the subsystem affinity index value used when making subsystem affinity requests.

Notes:

1. The subsystem index value is valid only for use on the MVS processor on which it was obtained and only during the current IPL.
2. A valid subsystem affinity index value is returned only for subsystems defined through the methods described in “Defining Your Subsystem” on page 160.

For more information, see “Maintaining Information About the Callers of Your Subsystem” on page 180.

Type of Request

Directed SSI call.

Issued to

Master subsystem.

Related SSI Codes

None.

Related Concepts

You need to understand the subsystem affinity service. See “Maintaining Information About the Callers of Your Subsystem” on page 180 for more information.

Environment

The caller (issuer of the IEFSSREQ macro) must include the following mapping macros:

- CVT
- IEFJESCT

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your program:

- IEFSSOBH
- IEFJSSIB
- IEFSSVS

The caller must meet the following requirements:

Minimum Authorization	Problem state, any PSW key
Dispatchable unit mode	Task or SRB
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts

Locks	No locks held
Control Parameters	The SSOB, SSIB, and SSVS control blocks can reside above or below 16 megabytes.
Recovery	The caller should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Assembler Services Guide* for more information on an ESTAE-type recovery environment.

Figure 5 shows the environment at the time of the call for SSI function code 15.

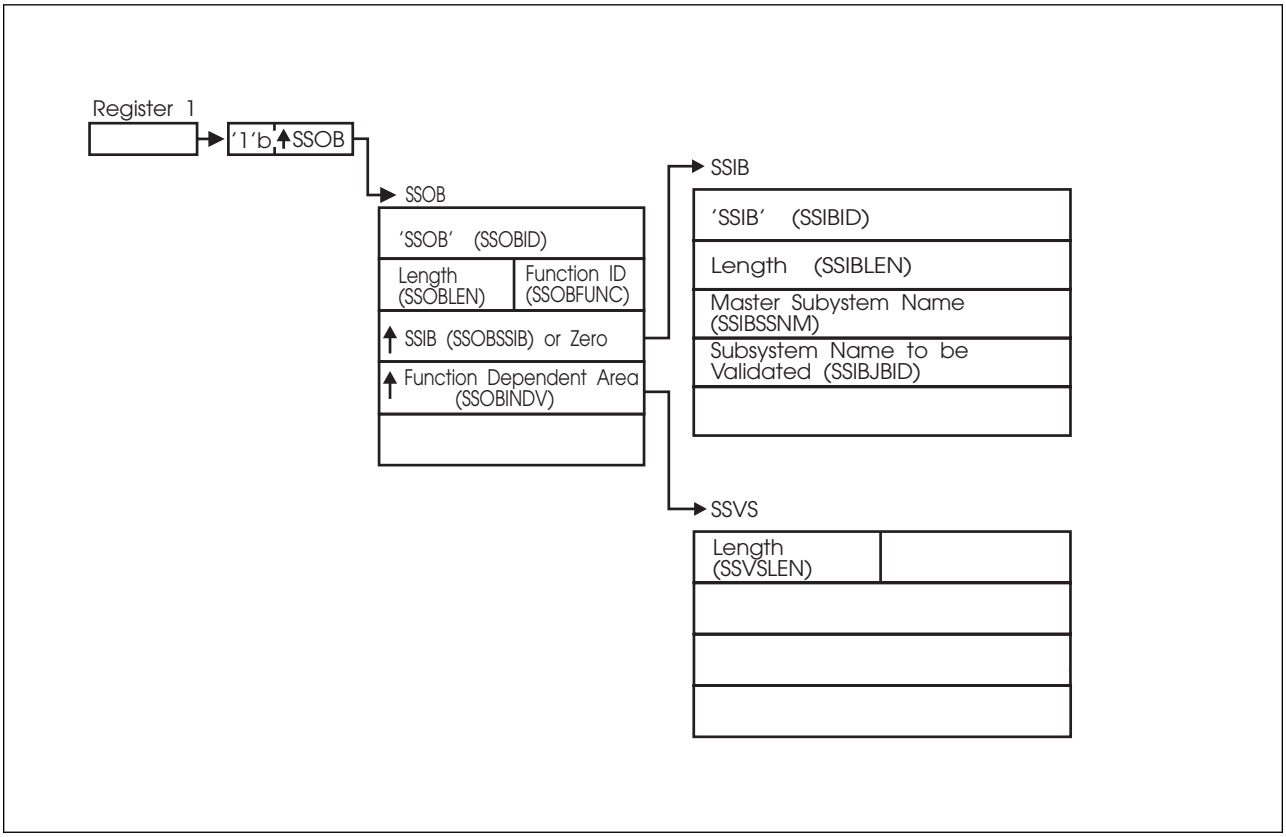

Figure 5. Environment at Time of Call for SSI Function Code 15

Input Register Information

Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

Register	Contents
1	Address of a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.
13	Address of a standard 18-word save area

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSVS

SSOB Contents: The caller sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function code 15 (SSOBVERS)
SSOBSSIB	Address of an SSIB control block or zero (if this field is zero, the life-of-job SSIB is used). See “Subsystem Identification Block (SSIB)” on page 8 for more information on the life-of-job SSIB.
SSOBINDV	Address of the function dependent area (SSVS control block)

Set all other fields in the SSOB control block to binary zeros before issuing the IEFSSREQ macro.

SSIB Contents: If you don't use the life-of-job SSIB, the caller must provide an SSIB and set the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of the subsystem that this verify subsystem function call is directed to (MSTR).
SSIBJBID	Name of the subsystem to be verified **Note:** This is an 8-character field. Because subsystem names can only be 1-4 characters, the subsystem name specified should be left-justified and padded to the right with blank (X'40') characters.

Set all other fields in the SSIB control block to binary zeros before issuing the IEFSSREQ macro.

SSVS Contents: The caller sets the following fields in the SSVS control block on input:

Field Name	Description
SSVSLEN	Length of the SSVS (SSVSSIZE) control block

Set all other fields in the SSVS control block to binary zeros before issuing the IEFSSREQ macro.

Output Register Information

When control returns to the caller, the general purpose registers contain:

Register	Contents
0	Used as a work register by the system
1	Address of the SSOB control block
2 — 13	Same as on entry to call
14	Return address

15 Return code

Return Code Information

The SSI places one of the following decimal return codes in register 15. Examine the return code to determine if the request was processed.

Return Code (Decimal)	Meaning
SSRTOK (0)	The Verify Subsystem function call completed. Check the SSOBRETN field for specific function information.
SSRTNSUP (4)	The subsystem specified in the SSIBSSNM field does not support the Verify Subsystem function call.
SSRTNTUP (8)	The subsystem specified in the SSIBSSNM field exists, but is not active.
SSRTNOSS (12)	The subsystem specified in the SSIBSSNM field is not defined to MVS.
SSRTDIST (16)	The pointer to the SSOB control block or the SSIB control block is not valid, or the function code specified in the SSOBFUNC field is greater than the maximum number of functions supported by the subsystem specified in the SSIBSSNM field.
SSRTLERR (20)	Either the SSIB control block or the SSOB control block has incorrect lengths or formats.
SSRTNSSI(24)	The SSI has not been initialized.

Output Parameters

Output parameters for the function routine are:

- SSOBRETN
- SSVS

SSOBRETN Contents: When control returns to the caller and register 15 contains a zero, the verify subsystem function places one of the following decimal values in the SSOBRETN field indicating whether the subsystem name in the SSIBJBID field is valid:

Value (Decimal)	Meaning
SSVSNAM (0)	Valid subsystem name
SSVSJBNM (4)	The name in the SSIBJBID field is not the name of a defined subsystem.

SSVS Contents: The SSVS control block contains the following information if a valid subsystem name was specified:

Field Name	Description
SSVSSCTP	Pointer to the subsystem's SSCVT.
SSVSNUM	The subsystem affinity index value that you can use in a SSAFF macro request. See “Maintaining Information About the Callers of Your Subsystem” on page 180 for more information on the SSAFF macro.

Product-sensitive programming interface

Request Job ID Call — SSI Function Code 20

The Request Job ID call (SSI function code 20) allows an authorized address space to establish a job structure. Once the caller receives a job ID, the address space can use JES services.

Type of Request

Directed SSI call.

Use Information

The following are a few examples of how a program running in an address space started under the master subsystem can, once it has obtained a job ID, use the primary subsystem (JES) services:

- Allocate an internal reader to submit jobs that run under JES. See *OS/390 MVS Programming: Assembler Services Guide* for more information on the internal reader.
- Allocate a SYSOUT data set (SSI function code 1) so that the program can retrieve a data set after using SSI function code 1.

While the address space might have been started under the master subsystem before JES initialization, the Request Job ID SSI call is honored only after JES is initialized.

Because the address space was not started under JES control, JES does not have an internal job structure for the address space. Use of SSI function code 20 establishes the necessary structure so that subsequent requests for JES services for that address space may be performed properly.

Issued to

A JES, typically the primary subsystem. In a JES2 environment, the call may be made to both the primary JES2 as well as any secondary JES2. It is even possible to request job IDs from both a primary JES2 and a secondary JES2 at the same time, though each job ID requires a separate IEFSSREQ call.

Related SSI Codes

Issue the Return Job ID call (SSI function code 21) after the Request Job ID call so that additional Request Job ID calls can be made.

Related Concepts

You need to understand:

- JES2 can issue ENF (event notification facility) signal 40 during initialization or orderly termination to communicate the fact that JES2 has initialized, or is ending.
- JES3 issues ENF signal 40 during initialization or when the JES3 address space is ending (regardless of orderly shutdown or abnormal termination).
- Issue the Return Job ID call (SSI function code 21) to "disconnect" from JES and return the job ID that was obtained with SSI function code 20.

- When JES2 processes the Request Job ID call from a task started under the master subsystem, some of the attributes of this task will be defined by the STCCLASS initialization statement. Specifically, the value defined on the MSGCLASS parameter determines if the joblog output produced from the SSI function code 20 job is suppressed. In this example, you must define the MSGCLASS parameter of the STCCLASS initialization statement so that the class has a disposition of purge. Note that changing the MSGCLASS value may produce an undesirable effect on other started tasks in your system.

Environment

The caller (issuer of the IEFSSREQ macro) must include the following mapping macros:

- CVT
- IEFJESCT

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your program:

- IEFSSOBH
- IEFJSSIB
- IEFSSRR

The caller must meet the following requirements:

Minimum Authorization	Supervisor state
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control Parameters	The SSOB, SSIB, and SSRR control blocks can reside in storage above 16 megabytes.
Recovery	The caller should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on an ESTAE-type recovery environment.

Figure 6 on page 50 shows the environment at the time of the call for SSI function code 20.

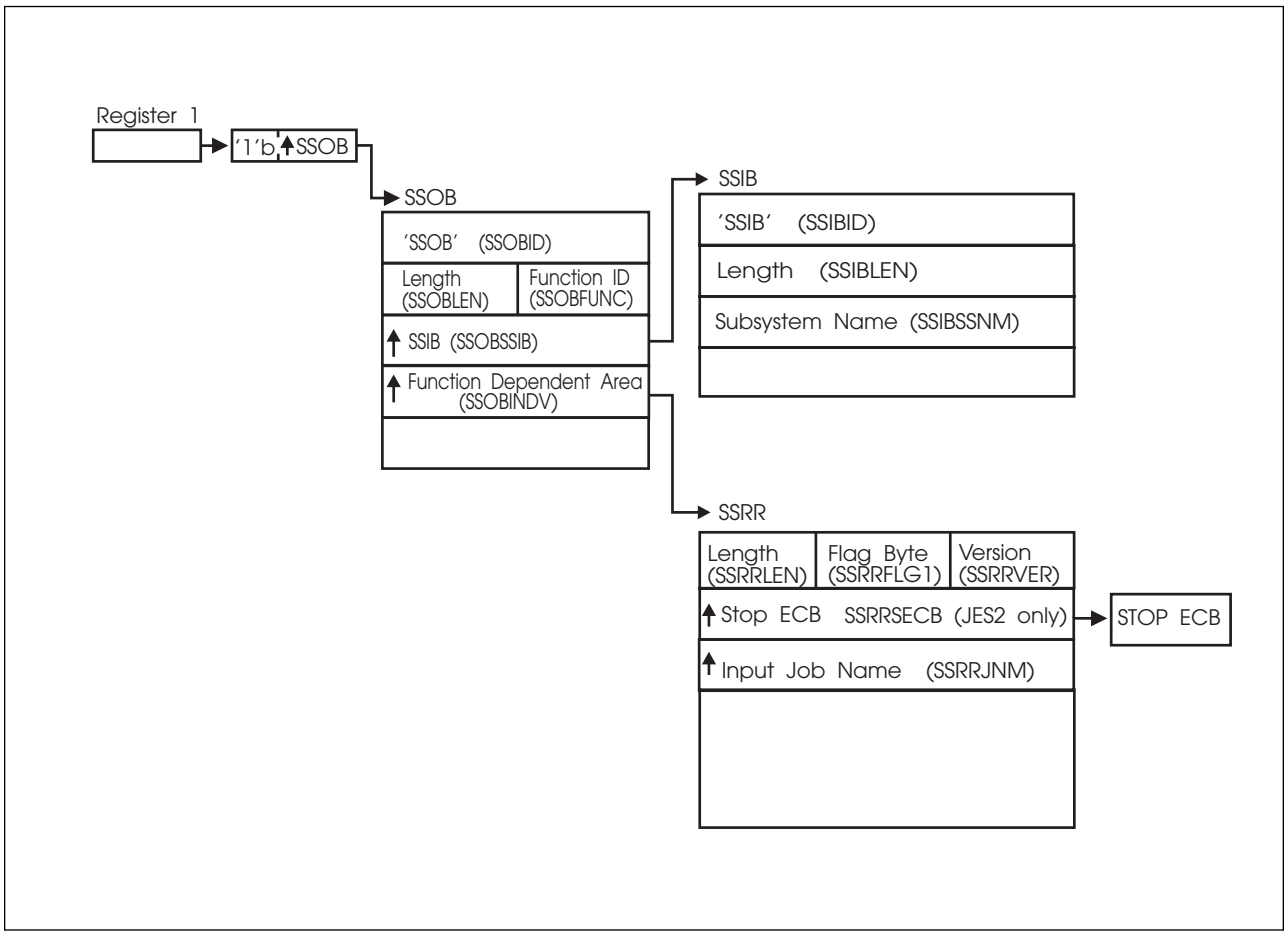

Figure 6. Environment at Time of Call for SSI Function Code 20

Input Register Information

Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

Register	Contents
1	Address of a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.
13	Address of a standard 18-word save area.

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSRR

SSOB Contents: The caller of the function code sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB (SSOBHSIZ) control block

SSOBFUNC SSI function code 20 (SSOBRQST)

SSOBSSIB Address of an SSIB control block

SSOBINDV Address of the function dependent area (SSRR control block)

Set all other fields in the SSOB control block to binary zeros before issuing the IEFSSREQ macro.

SSIB Contents: The caller of the function code sets the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of the subsystem to which this Request Job ID call is directed. It is usually the primary JES, or in the case of JES2, a possible secondary JES.

Set all other fields in the SSIB control block to binary zeros before issuing the IEFSSREQ macro.

SSRR Contents: The caller of the function code sets the following fields in the SSRR control block on input:

Field Name	Description
SSRRLEN	Length of the SSRR (SSRRSIZE) control block
SSRRFLG1	Flag byte The caller of this function code can set one or more of the following bits: • **SSRRUASC** If SSRRUASC is set, JES assigns the JES-provided job name to the job found in the ASCB control block as follows: 1. Started task from the ASCBJBNS field, if the job is running as a started task, MOUNT, or LOGON. 2. Batch job from the ASCBJBNI field, if the job is running as a batch job or APPC transaction program. • **SSRRJNMP** If SSRRJNMP is set, JES uses the user-provided jobname in the SSRRJNM field. **Note:** The caller can set either the SSRRUASC bit or the SSRRJNMP bit, but not both. • **SSRRJOBL** If SSRRJOBL is set, JES explicitly creates a joblog. • **SSRRNJBL** If SSRRNJBL is set, JES does not explicitly create a joblog.

Note: JES explicitly creates a joblog by default when neither the SSRRJOBL bit nor the SSRRNJBL bit is set. Note that the caller cannot set both the SSRRJOBL bit and the SSRRNJBL bit.

SSRRVER Version of mapping for the caller. Set this field to SSRRCVER (an IBM-defined integer constant within the SSRR control block).

SSRRSECB For JES2 only, contains the pointer to a caller-supplied ECB. When JES2 posts this ECB, JES2 is ending. In response, issue the Return Job ID call (SSI function code 21).

Note: Do not rely on this ECB always being posted during the ending of JES2. JES2 can also end abnormally.

SSRRJNM An optional job name to be used for this job. The name is left-justified and padded to the right with blank (X'40') characters. JES uses this name as the job name if the caller set the SSRRJNMP bit in the SSRRFLG1 flag byte, as described earlier.

Set all other fields in the SSRR control block to binary zeros before issuing the IEFSSREQ macro.

Output Register Information

When control returns to the caller, the general purpose registers contain:

Register	Contents
0	Used as a work register by the system
1	Address of the SSOB control block
2 — 13	Same as on entry to call
14	Return address
15	Return code

Return Code Information

The SSI places one of the following decimal return codes in register 15. Examine the return code to determine if the request was processed.

Return Code (Decimal)	Meaning
SSRTOK (0)	The Request Job ID call completed. Check the SSOBRETN field for specific function information.
SSRTNSUP (4)	The subsystem specified in the SSIBSSNM field does not support this function.
SSRTNTUP (8)	The subsystem specified in the SSIBSSNM field exists, but is not active.
SSRTNOSS (12)	The subsystem specified in the SSIBSSNM field is not defined to MVS.

SSRTDIST (16) The pointer to the SSOB control block or the SSIB control block is not valid, or the function code specified in the SSOBFUNC field is greater than the maximum number of functions supported by the subsystem specified in the SSIBSSNM field.

SSRTLERR (20) Either the SSIB control block or the SSOB control block has invalid lengths or formats.

SSRTNSSI(24) The SSI has not been initialized.

Output Parameters

Output parameters for the function routine are:

- SSIB
- SSOBRETN

SSIB Contents: The SSIB control block contains:

- The JES name (supplied by the user on input)
- The 8-character returned job ID
- The subsystem use value (contained in the SSIBSUSE field-JES3 only)

The subsystem name (SSIBSSNM), returned job ID (SSIBJBID) and subsystem use value (SSIBSUSE-JES3 only) must be used on subsequent IEFSSREQ calls to the appropriate JES for subsequent services.

SSOBRETN Contents: When control returns to the caller and register 15 contains a zero, the SSOBRETN field contains one of the following decimal values:

Value (Decimal)	Meaning
SSRROK (0)	Successful completion. JES assigned a job ID to the caller. The job ID is available in the SSIBJBID field. See “Restrictions” on page 54 for information on the processing that takes place after successful completion has been obtained.
SSRRFAIL (4)	The Request Job ID call did not successfully complete. This can happen if JES is in the process of ending, and therefore cannot return job IDs. This caller cannot make use of subsequent JES services.
SSRRFREQ (8)	The Request Job ID call is already known to this JES, and may not have a second job ID established.
SSRRNOEC (16)	For JES2 only, an ECB was not supplied through the SSRRSECB pointer field on the Request Job ID call.
SSRRPRME (20)	There is an error in the SSRR data area. For example, both the SSRRJOBL bit and the SSRRNJBL bits may be set.
SSRRPERR (36)	The JES processing this call has returned a program error. This can happen if the JES does not have enough virtual storage available to create either the job structure or other control blocks for the requesting address space.

Restrictions

For both JES2 and JES3, the following restrictions apply to the caller issuing the Request Job ID call:

- Cannot receive multiple job IDs for different tasks running in the same address space, because the job ID is associated with an address space.
- Can only make one Request Job ID call, unless a Return Job ID call is done, to the same JES, in which case another Request Job ID call can be made.

  Note: The returned job ID will probably not be the job ID that was previously received.
- Must use the subsystem name that was used in the Request Job ID in the SSIB control block (for IEFSSREQ) or in the DALSSREQ text unit (for DYNALLOC) for any subsequent service request.

  This name uniquely identifies the appropriate receiving JES, either primary (JES2 or JES3), or secondary (JES2 only).

For JES2 only, the following restriction applies to the caller issuing the Request Job ID call:

- Must use different SSIB control blocks to direct more than one Request Job ID call to multiple (and different) JES2 subsystems simultaneously. This restriction applies only when more than one JES2 is running (that is, when there are additional secondary JES2 subsystems).

Considerations When Using the Automatic Restart Manager

If a program registers with the automatic restart manager before requesting a job ID, the automatic restart manager will not associate the program with JES. If a system failure occurs, the automatic restart manager can restart the program on any system in the sysplex, possibly one in a different JES2 multi-access spool configuration (MAS) or JES3 complex from where the program was running before the system failure. The program cannot depend on access to jobs or output it created in the original MAS or complex.

If a program registers with the automatic restart manager after requesting a job ID, the automatic restart manager will associate the program with JES. If a system failure occurs, the automatic restart manager can restart the program on any member in the same MAS or complex. If the program requests job IDs from more than one JES, the automatic restart manager uses the JES from the first request.

End of Product-sensitive programming interface

Product-sensitive programming interface

Return Job ID Call — SSI Function Code 21

The Return Job ID call (SSI function code 21) allows an authorized address space to return to JES the job structure that was obtained by invoking the Request Job ID call (SSI function code 20).

Once the caller returns the job ID, that address space may no longer use JES services (on behalf of this particular job ID) unless a Request Job ID SSI call is made again.

Type of Request

Directed SSI call.

Use Information

A program uses this request to give back to JES the job ID that it received from a previous Request Job ID call (SSI function code 20). The caller issues the Return Job ID call (SSI function code 21) when the address space determines that it no longer needs JES services.

Issued to

A JES, typically the primary subsystem. In a JES2 environment, the call may be made to both the primary JES2 as well as any secondary JES2 subsystems, when services from either subsystems have been obtained through a previous Request Job ID call (SSI function code 20).

Related SSI Codes

The Request Job ID call (SSI function code 20) must be used to obtain the job ID supplied by JES before the caller can request the Return Job ID call.

Related Concepts

You need to understand the Request Job ID call (SSI function code 20).

Environment

The caller (issuer of the IEFSSREQ macro) must include the following mapping macros:

- CVT
- IEFJESCT

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your program:

- IEFSSOBH
- IEFJSSIB
- IEFSSRR

The caller must meet the following requirements:

Minimum Authorization	Supervisor state
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN

ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control Parameters	The SSOB, SSIB, and SSRR control blocks can reside in storage above 16 megabytes.
Recovery	The caller should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on an ESTAE-type recovery environment.

Figure 7 shows the environment at the time of the call for SSI function code 21.

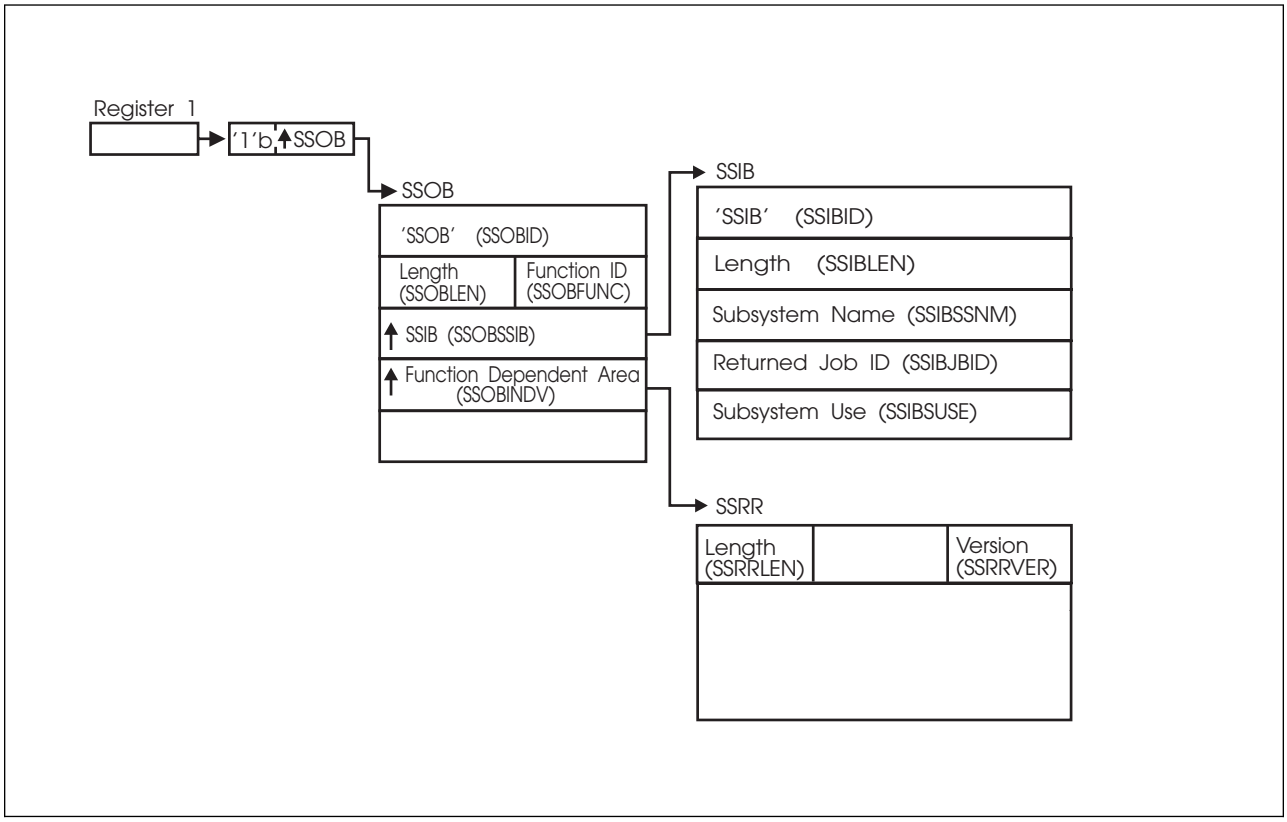

Figure 7. Environment at Time of Call for SSI Function Code 21

Input Register Information

Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

Register	Contents
1	Address of a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.
13	Address of a standard 18-word save area.

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSRR

SSOB Contents: The caller sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function code 21 (SSOBRTRN)
SSOBSSIB	Address of an SSIB control block
SSOBINDV	Address of the function dependent area (SSRR control block)

Set all other fields in the SSOB control block to binary zeros before issuing the IEFSSREQ macro.

SSIB Contents: The caller sets the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of the subsystem to which this Return Job ID call is directed. This name identifies either the primary subsystem, or in the case of JES2, a secondary JES subsystem. You must use the same subsystem name in this SSIBSSNM field as you used for the original Request Job ID call (SSI function code 20).
SSIBJBID	Returned job ID You must use the job ID obtained during the previously issued Request Job ID call (SSI function code 20).
SSIBSUSE	(JES3 only) Subsystem use — the SSIBSUSE value that was returned upon completion of the Request Job ID call (SSI function code 20).

Set all other fields in the SSIB control block to binary zeros before issuing the IEFSSREQ macro.

SSRR Contents: The caller sets the following fields in the SSRR control block on input:

Field Name	Description
SSRRLEN	Length of the SSRR (SSRRSIZE) control block

SSRRVER Version of mapping for the caller. Set this field to SSRRCVER (an IBM-defined integer constant within the SSRR control block).

Note: This SSRR control block can be the same SSRR control block that was provided on the original Request Job ID call (SSI function code 20). All of the fields except the SSRRLEN field and the SSRRVER field contain binary zeros.

Output Register Information

When control returns to the caller, the general purpose registers contain:

Register	Contents
0	Used as a work register by the system
1	Address of the SSOB control block
2 — 13	Same as on entry to call
14	Return address
15	Return code

Return Code Information

The SSI places the following decimal return codes in register 15. Examine the return code to determine if the request was processed.

Return Code (Decimal)	Meaning
SSRTOK (0)	The Return Job ID call completed. Check the SSOBRETN field for specific function information.
SSRTNSUP (4)	The subsystem specified in the SSIBSSNM field does not support this function.
SSRTNTUP (8)	The subsystem specified in the SSIBSSNM field exists, but is not active.
SSRTNOSS (12)	The subsystem specified in the SSIBSSNM field is not defined to MVS.
SSRTDIST (16)	The pointer to the SSOB control block or the SSIB control block is not valid, or the function code specified in the SSOBFUNC field is greater than the maximum number of functions supported by the subsystem specified in the SSIBSSNM field.
SSRTLERR (20)	Either the SSIB control block or the SSOB control block has invalid lengths or formats.
SSRTNSSI(24)	The SSI has not been initialized.

Output Parameters

Output parameters for the function routine are:

- SSIB
- SSOBRETN

SSIB Contents: The SSIB control block no longer contains a valid job ID on output. If this address space needs subsequent JES services, issue the Request Job ID call (SSI function code 20) again.

SSOBRETN Contents: When control returns to the caller and register 15 contains a zero, the SSOBRETN field contains one of the following decimal values:

Value (Decimal)	Meaning
SSRROK (0)	Successful completion. The caller's job ID was returned to JES. This address space is not available to JES services unless a subsequent Request Job ID call (SSI function code 20) obtains a new job ID.
SSRRFRET (12)	The Return Job ID call cannot return a job ID to JES because a Request Job ID call (SSI function code 20) was not made. The job ID is not returned.
SSRRPERR (36)	The JES processing this call has returned a program error. An error can occur if the job ID returned failed internal JES validation, or if JES does not have enough virtual storage for a work area.

End of Product-sensitive programming interface

Request Subsystem Version Information Call — SSI Function Code 54

The Request Subsystem Version Information Call (SSI function code 54) provides a requesting program the ability to obtain version-specific information about a particular subsystem.

Type of Request

Directed SSI call.

Use Information

A caller issues SSI function code 54 to obtain the following information about a particular subsystem:

- Product function modification identifier (FMID)
- Product version number
- Subsystem common name (such as 'JES2')
- Network node name
- JES system member name
- Whether the subsystem supports the following functions:
  - Dynamic output
  - Restarting of initiators
  - Dynamic allocation of multiple started task (STC) and TSO/E internal readers.
  - Client print

Note that 4-digit device numbers are supported.

Issued to

- Master
- JES2/JES3
- User-supplied or vendor-supplied subsystem.

Related SSI Codes

None.

Related Concepts

You need to understand:

- ENF (event notification facility) signal 40

  JES2 can issue ENF signal 40 during initialization or orderly termination to communicate the fact that JES2 has initialized, or is ending.

  JES3 issues ENF signal 40 during initialization or when the JES3 address space is terminating (regardless of orderly shutdown or abnormal termination).

  You might need to know when JES is initializing or ending when using SSI function code 54 to obtain **relatively static** (information that is not likely to change between restarts) information about a JES subsystem. If JES ends and is restarted with a new level, or with a different functional capability, you will need to reissue this SSI request to obtain information about the new capabilities of JES. During initialization or orderly termination, JES issues event notification facility (ENF) signal 40, for which authorized callers can listen. For information about how programs can listen for ENF signals, see the description of using the ENFREQ macro in *OS/390 MVS Programming: Authorized*

Assembler Services Guide. Note that the users of ENFREQ must be authorized.

- The caller issues the IEFSSREQ with the SSVI control block used as input. The information that the subsystem returns will be contained within four sections of the SSVI control block.
  - Fixed header input section

    The user provides this information before issuing IEFSSREQ. This information is explained "Fixed Header Input Section" on page 63.
  - Fixed header output section

    Information returned by all called subsystems is returned in this section. This information is explained "Fixed Header Output Section" on page 65.
  - Installation variable output section (JES)

    As of JES2 SP 4.3 or JES3 SP 5.1.1, installations can supply their own keywords, or override one or more keywords returned in the system variable output section. This information is explained "Installation Variable Output Section" on page 67.
  - System variable output section

    The called subsystem returns subsystem-specific information in the form of keyword value specifications. This information is explained "System Variable Output Section" on page 67.

Environment

The caller (issuer of the IEFSSREQ macro) must include the following mapping macros:

- CVT
- IEFJESCT

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your program:

- IEFSSOBH
- IEFJSSIB
- IEFSSVI

The caller must meet the following requirements:

Minimum Authorization	Problem state, any PSW key
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control Parameters	The SSOB, SSIB, and SSVI control blocks can reside in storage above 16 megabytes.
Recovery	The caller should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Assembler Services Guide* for more information on an ESTAE-type recovery environment.

Figure 8 on page 62 shows the environment at the time of the call for SSI function code 54.

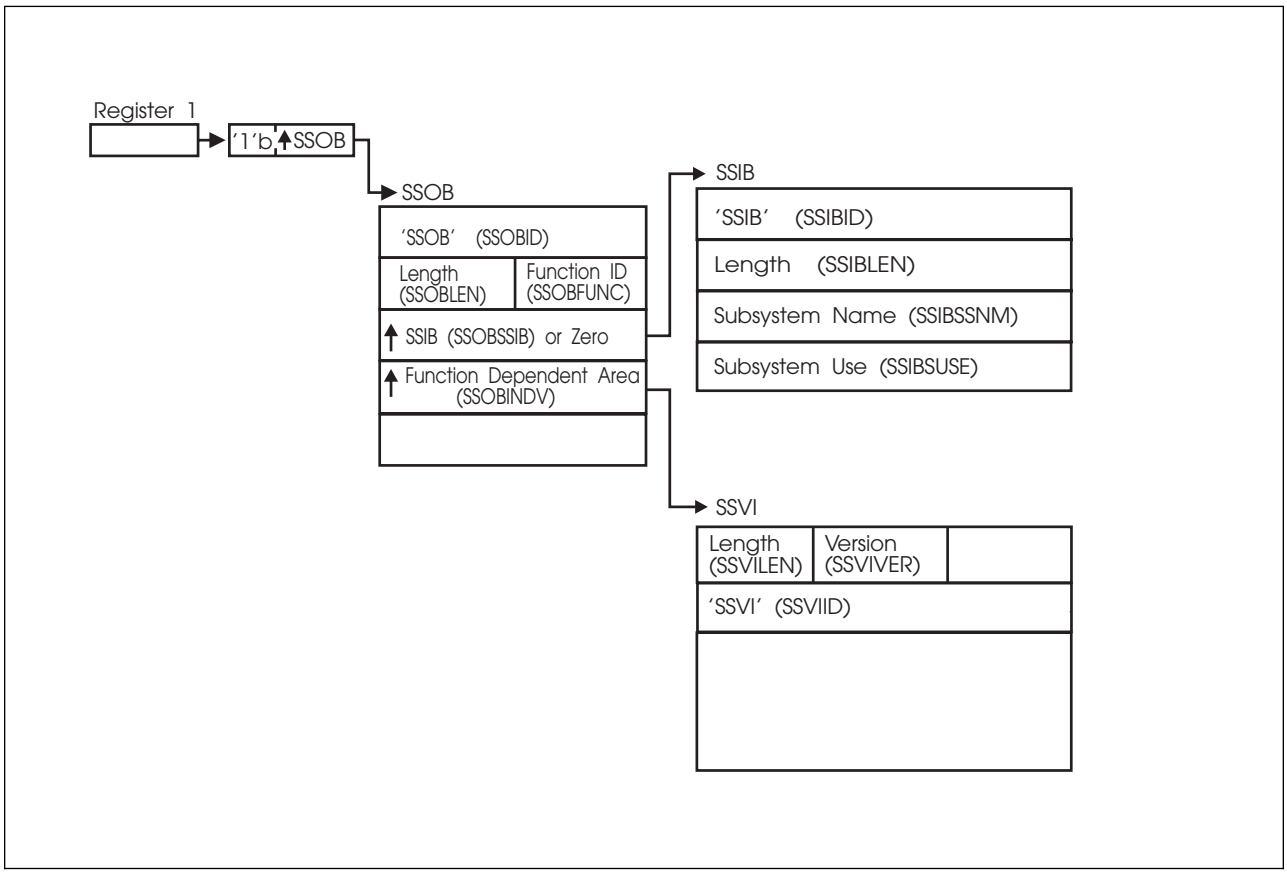

Figure 8. Environment at Time of Call for SSI Function Code 54

Input Register Information

Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

Register	Contents
1	Address of a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.
13	Address of a standard 18-word save area.

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSVI

SSOB Contents: The caller sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'

SSOBLEN	Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function Code 54 (SSOBSSVI)
SSOBSSIB	Address of the SSIB control block or zero (if this field is zero, the life-of-job SSIB is used). See “Subsystem Identification Block (SSIB)” on page 8 for more information on the life-of-job SSIB.
SSOBINDV	Address of the function dependent area (SSVI control block)

Set all other fields in the SSOB control block to binary zeros before issuing the IEFSSREQ macro.

SSIB Contents: If you don't use the life-of-job SSIB, the caller must provide an SSIB and set the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of the subsystem to which this Request Subsystem Version Information call is directed. It is either the master subsystem, a JES2 (primary or secondary) subsystem, a JES3 subsystem, or a user-supplied or vendor-supplied subsystem.

Set all other fields in the SSIB control block to binary zeros before issuing the IEFSSREQ macro.

SSVI Contents: The input information in the SSVI control block is contained in the following area mapped within the SSVI control block:

- Fixed header input section

The caller sets these fields before issuing the IEFSSREQ macro.

Fixed Header Input Section

The fixed header input section contains the information that the caller needs to provide to the subsystem on input for this Request Subsystem Version Information call.

Field Name	Description
SSVILEN	Length of entire area — Set this field to a value that is at least equal to the value of SSVIMSIZ (a constant contained within the SSVI control block). The length includes the fixed header section, plus the system variable section and the installation variable section. The caller must ensure that the length specified in the SSVILEN field is large enough to contain the requested information.
SSVIVER	Version of mapping for the caller — Set this field to SSVICVER (an IBM-defined integer constant within the SSVI control block).
SSVIID	Identifier 'SSVI'

Set all other fields in the SSVI control block to binary zeros before issuing the IEFSSREQ macro.

Output Register Information

When control returns to the caller, the general purpose registers contain:

Register	Contents
0	Used as a work register by the system
1	Address of the SSOB control block
2 — 13	Same as on entry to call
14	Return address
15	Return code

Return Code Information

The SSI places one of the following return codes in register 15. Examine the return code to determine if the request was processed.

Return Code (Decimal)	Meaning
SSRTOK (0)	Successful completion. The subsystem request completed. Check field SSOBRETN for specific function information.
SSRTNSUP (4)	The subsystem specified in the SSIBSSNM field does not support this function.
SSRTNTUP (8)	The subsystem specified in the SSIBSSNM field exists, but is not active.
SSRTNOSS (12)	The subsystem specified in the SSIBSSNM field is not defined to MVS.
SSRTDIST (16)	The pointer to the SSOB control block or the SSIB control block is not valid, or the function code specified in the SSOBFUNC field is greater than the maximum number of functions supported by the subsystem specified in the SSIBSSNM field.
SSRTLERR (20)	The SSIB control block or SSOB control block has invalid lengths or formats.
SSRTNSSI(24)	The SSI has not been initialized.

Output Parameters

Output parameters for the function routine are:

- SSOBRETN
- SSVI

SSOBRETN Contents: When control returns to the caller, the SSOBRETN field contains one of the following decimal values if general purpose register 15 was zero:

Value (Decimal)	Description
SSVIOK (0)	Successful completion. The requested information was returned. See the SSVI control block section description below for the specific format of the returned information.

SSVINSTR (8) The requesting application did not provide a storage area large enough to contain the requested information. The SSVIRLEN field indicates the total amount of storage this request requires to complete successfully.

When you receive this return code, obtain the appropriate amount of storage for a new IEFSSVI mapping macro by using the value returned in the SSVIRLEN field. Then, resubmit the request and set the SSVILEN field to the new storage size obtained from the SSVIRLEN field from the previous request.

SSVIPARM (16) The SSVI data area contains one or more of the following parameter errors:

- SSOBINDV (in the SSOB control block) did not contain the address of a valid SSVI control block
- SSVIID did not contain 'SSVI'
- SSVIVER did not specify a valid version of the SSVI control block
- SSVILEN contained a value that is less than the value of SSVIMSIZ (an IBM-defined integer constant within the SSVI control block).

When you receive this return code, fix the problem and resubmit the request.

SSVIABLG (24) An abend or logical error was encountered within the called subsystem's function code routine.

When you receive this return code, search the problem report databases for a fix to the problem. If no fix exists, contact the IBM support center.

SSVI Contents: The output information returned in the SSVI control block is contained in one or more of the following areas mapped within the SSVI control block:

- Fixed header output section
- System variable output section
- Installation variable output section

Each of these areas is described in order, followed by a description of the format of the two variable output sections.

Fixed Header Output Section

The fixed header output section contains information that the called subsystem returns to the requesting program. The called subsystem sets all fields, although they may be binary zeros.

The following shows how the master and JES subsystems set the contents of the fixed header output section:

Field Name	Description
SSVIRLEN	A 2-byte binary field that contains either the length of the storage used (if the caller's request was successful), or the length of storage required (if the request failed because the caller did not specify enough storage). To determine whether the SSVIRLEN field contains returned or required storage, check the return code in SSOBRETN, which indicates: **Decimal Value** — **Meaning** **SSVIOK (0)** — Request was successful. The SSVIRLEN field contains the length, in bytes, of the returned data. **SSVINSTR (8)** — Request failed. The caller did not specify enough storage in the SSVIRLEN field. The SSVIRLEN field contains the amount of storage, in bytes, the subsystem requires to return the requested information. Note that this field is not set when the SSOBRETN field contains return code SSVIPARM (16) or SSVIABLG (24).
SSVIRVER	A 1-byte binary field that contains the version of the SSVI control block used by the subsystem. When the caller's version of the SSVI control block does not match the version used by the called subsystem, the subsystem returns information based on the older of the two versions of the SSVI control block.
SSVIFLEN	A 2-byte integer field that contains the length of the fixed header output section of the SSVI control block the subsystem uses.
SSVIASID	A 2-byte binary field that indicates the ASID of the subsystem. A value of X'FFFF' indicates that the address space abended. This field contains valid information only if the caller-supplied version in field SSVIVER is greater than or equal to 2.
SSVIVERS	An 8-byte character field that specifies the version of the subsystem. For example, JES returns: SP 5.1.0, SP 5.2.1, or OS 1.1.0. The master subsystem returns the same value as that contained in CVTPRODN.
SSVIFMID	An 8-byte character field that specifies the FMID of the subsystem (for example, HBB5510, HJE5510, or HJS5511).
SSVICNAM	An 8-byte character field that is left-justified, and padded to the right with blanks and contains the common name of the subsystem. For example, in a poly-JES environment, the secondary JES2 subsystem (for example, JESA) returns: 'JES2 '. The master subsystem of an MVS system returns: 'MASTER '.
SSVIPLVL	This 1-byte field contains either zero or a value that indicates the relative subsystem product level. For example, with either JES, the relative subsystem product level value will increase by at least one for each subsequent release of the subsystem. For JES2 SP 3.1.3, the relative subsystem product level value is decimal '20'

and for OS/390 Release 1 JES2, the relative subsystem product level value is decimal '26'. For more information, see topic “Determining the JES2 Release Level” in *OS/390 JES2 Installation Exits*. For JES3 SP 3.1.2, the relative subsystem product level value is decimal '1' and for OS/390 Release 1 JES3, the relative subsystem product level value is decimal '6'. For more information, see topic “Determining the JES3 Release Level” in *OS/390 JES3 Customization*.

This field contains valid information only if the caller-supplied version in field SSVIVER is greater than or equal to 2.

SSVISLVL This 1-byte field indicates the relative service level of the subsystem and contains either zero or the service level of the subsystem. For example, the JES relative service level is set to zero for each new product level and will increase by at least one each time significant maintenance or function is added within a release. For additional information concerning this field, see *OS/390 JES2 Installation Exits* or *OS/390 JES3 Customization*.

This field contain valid information only if the caller-supplied version in field SSVIVER is greater than or equal to 2.

SSVIUDOF A 4-byte integer field that contains the offset from the start of the IEFSSVI DSECT, to the start of the installation variable output data section. The subsystem sets this field to zero if there is no installation variable output data section.

SSVISDOF A 4-byte integer field that contains the offset from the start of the SSVI control block, to the start of the system variable output data section. The subsystem sets this field to zero if there is no system variable output data section.

System Variable Output Section

The system variable output section contains subsystem-specific information as keyword values. For more information see “Format of the Variable Output Sections” on page 68.

The called subsystem's function routine can return keyword values to SSI function code 54 callers in the system variable output section, and, optionally for JES, the installation variable output section (defined through JES2 Exit 24, or through JES3 via IATUX63). The subsystem's function routine returns two offsets, SSVIUDOF and SSVISDOF, in the fixed header output section. Both are offsets from the start of the SSVI control block to the beginning of their corresponding data area. To indicate that an output section does not exist, the subsystem's function routine sets the offset value to zero. Each data area contains a 2-byte length field, which itself is not included in the length of the string.

Installation Variable Output Section

As of JES2 SP 4.3 or JES3 SP 5.1.1, installations can use the installation variable output data section to define their own keywords, or override one or more of the keyword values returned by the called subsystem in the system variable output section. The installation variable output data section has the same format as the system variable output data section. For more details see “Format of the Variable Output Sections” on page 68.

Installations can specify their own keyword values to be returned in the installation variable output section (through JES2 Exit 24 or JES3 via IATUX63). For more information about using JES2 Exit 24, see *OS/390 JES2 Installation Exits*. For more information about using JES3 IATUX63, see *OS/390 JES3 Customization*.

Format of the Variable Output Sections

The following is a description of the subsystem and installation variable output sections:

Field Name	Description
SSVIVLEN	A 2-byte signed hexadecimal field that contains the length of the variable output data string. The length of this field is not included in the length of the string.
SSVIDAT	A variable length character string (its length is set through SSVIVLEN) that contains a set of keywords and their respective values. When master (MSTR) or JES is the called subsystem, any, all or none of the keyword values shown in Figure 9 are returned to the SSI code 54 caller.

Procedure of Searching Data Strings: When searching the variable output data strings, IBM recommends that installations have their SSI code 54 callers search the installation variable output section, if one exists, before searching the system variable output section. (The callers would use the first instance of a searched for keyword.) By following this procedure, the installation can add its own values to those returned by the SSI, and override the system values, without actually changing the information in the system variable output section.

IBM-Defined Keywords: The following table shows the IBM-defined keywords that can be returned in the variable-length character string:

Figure 9 (Page 1 of 2). IBM-Defined Keywords

Keyword	Explanation
,JES_NODE=*'name'*	Specifies the network node name of the JES.
,JES_MEMBERNAME=*'name'*	Specifies the member name of a particular JES2 in a multi-JES configuration or the JES3 main name in a JES3 complex.
,INITIATOR_RESTART='YES|NO'	Indicates whether the subsystem supports the restarting of initiators.
,DYNAMIC_OUTPUT='YES|NO'	Indicates whether the subsystem supports the dynamic output feature.
,MULTIPLE_STCTSO='YES|NO'	Indicates whether the subsystem supports dynamic allocation of multiple started task (STC) and TSO/E internal readers.
,FOUR_DIGIT_DEVNUMS='YES|NO'	Indicates whether the subsystem supports 4-digit device numbers.
,AUTO_RESTART_MANAGER='YES|NO'	Indicates whether the subsystem supports using the automatic restart manager.
,SAPI='YES'	Indicates SAPI is supported by this JES.

Figure 9 (Page 2 of 2). IBM-Defined Keywords

Keyword	Explanation
,SAPI_VOL_SELECT='NO'	Indicates selection by volume not supported.
,SAPI_PRTY_SELECT='NO'	Indicates selection by priority not supported.
,SAPI_CHARS='NO'	Indicates selection by characters not supported.
,SAPI_IP_SELECT='NO'	Indicates selection by IP address (Internet protocol) not supported.
,SAPI_MOD_SELECT='NO'	Indicates selection by modification id not supported.
,CLIENT_PRINT='YES'	Indicates that the JES supports the creation of a client token in support of client printing.
,WTR_SYSOUT_CLASS='*classes*'	Indicates the SYSOUT class for which output is placed on the JES3 writer queue. (See note.) For JES2, classes are for non-held SYSOUT.
,TSO_SYSOUT_CLASS='*classes*'	Indicates the SYSOUT class for which output is placed on the HOLD queue, and is held for TSO/E. (See note.) For JES2, classes are held for SYSOUT.
,EXW_SYSOUT_CLASS='*classes*'	Indicates the SYSOUT class for which output is placed on the HOLD queue and is held for external writers. (See note.) This keyword is not applicable to JES2.

Note:: *class* can be a value of A through Z or 0 through 9.

No blanks or commas are returned.

For JES3, classes that are defined to have SYSOUT directed to NJE are not returned.

For JES3, classes that are defined to have zero copies created are not returned.

For JES3, classes that are defined to be held for **both** TSO as well as external writers are not returned.

The format of the data in the variable output sections is:

```
,keyword='value',keyword='value',...,keyword='value'
```

Note that each keyword value in the data string is enclosed by a pair of apostrophes and preceded by a comma. All values must be upper-case.

Restrictions for Variable Output Section: Double-byte character set information is not currently recognized for variable output section strings.

Specifying Keywords

Installations, or any subsystem that supports the Request Subsystem Version Information call, must observe the following syntax rules when specifying keywords in the SSVIDAT field:

- A comma starts the entire string, and a comma must delimit each keyword from the previous keyword. This syntax allows the caller's function routine to use an index-type function when searching for keywords. For example, an index for ",keyword='" provides a valid technique for searching for the presence of the keyword in a string.

  The length of the data string can exceed 256 characters; ensure that the caller's parsing function is coded to handle very long data strings.

  An apostrophe ('), comma (,), and equal sign (=) are not allowed as part of a keyword term. For example, the following keyword terms are not allowed:

  – KEYWORD'S='...'
  – KEY=WORD='...'
  – KEYWORD,='...'
- The prefix value USER_ is reserved for installations to pass their own information in the installation variable output section.
- The '=' sign is required.
- Not all keywords need be returned by the subsystem service.
- The combination of an equal sign followed by an apostrophe (=').. is not allowed as part of a keyword value.
- Alphabetic characters for a keyword value are assumed to be in upper case unless otherwise stated.
- If a registered keyword appears in an installation string, then the allowable values are the same as the system string definition.
- The apostrophes surrounding the value for a keyword are required.
- A null value is indicated by two apostrophes in sequence.
- To code an apostrophe within the keyword value, code two apostrophes and enclose the keyword value within apostrophes.

Additional Recommendations for Specifying Keywords:

- Define yes or no choices as 'YES' or 'NO' (not abbreviated).
- Specify any numeric values as unsigned decimal numbers.
- Avoid specifying multiple parameters per keyword. Instead, use a separate keyword for each parameter, when possible.
- Numeric values must be passed in zoned-decimal format.
- When a keyword is located in a string, the end of the keyword's value should be determined prior to performing any comparisons. This ensures that the value that is searched for is not just a substring for another value.
- A feature or function that may be activated or inactivated while a subsystem is still active may not be good candidates to include in the string. An exception to this would be if the subsystem has a mechanism to inform all potential requesters interested in the feature or function.

Example

The following is a coded example of a program that generates a Request Subsystem Version Information call.

This program is reentrant, and does not have to run in an authorized library.

```
SSIREQ54 TITLE '- ISSUE SUBSYSTEM INFORMATION SSI CALL'
SSIREQ54 AMODE 31
SSIREQ54 RMODE ANY
         SPLEVEL SET=4
*********************************************************************
* FUNCTION: THIS PROGRAM GENERATES A SUBSYSTEM VERSION INFORMATION  *
*           CALL.  IT DISPLAYS THE RETURNED INFORMATION ON THE      *
*           ON THE OPERATOR CONSOLE.  THE SUBSYSTEM CALL IS         *
*           DIRECTED TO THE MASTER SUBSYSTEM.                       *
*                                                                   *
* NAME OF MODULE: SSIREQ54                                          *
*                                                                   *
* REGISTER USE:                                                     *
*                                                                   *
*              0                  PARM REGISTER                     *
*              1                  PARM REGISTER                     *
*              2                  SSOB                              *
*              3                  SSIB                              *
*              4                  SSVI                              *
*              5                  SSVI SIZE USED                    *
*              6                  SSVI SIZE NEEDED                  *
*              7                  UNUSED                            *
*              8                  ABEND VALUE REGISTER              *
*              9                  IEFSSREQ/SSVI RETURN CODES        *
*              10                 UNUSED                            *
*              11                 UNUSED                            *
*              12                 SSIREQ54 BASE REGISTER            *
*              13                 SAVE AREA CHAIN REGISTER          *
*              14                 PARM REGISTER / RETURN ADDR       *
*              15                 PARM REGISTER / COND CODE         *
*                                                                   *
* ATTRIBUTES:  PROBLEM STATE, AMODE(31), RMODE(ANY)                 *
*                                                                   *
* NOTE:  THIS IS A SAMPLE PROGRAM.                                  *
*                                                                   *
*********************************************************************
         SPACE ,
SSIREQ54 START 0
         TITLE '- EQUATES'
*********************************************************************
*        GENERAL EQUATES                                            *
*********************************************************************
NOP      EQU   0                  NO OPERATION
FF       EQU   X'FF'              ALL BITS ON
EQUHOBON EQU   X'80000000'         HIGH ORDER BIT ON
*
*********************************************************************
*        AFTER COMPARE INSTRUCTIONS                                 *
*********************************************************************
GT       EQU   2                  A HIGH
LT       EQU   4                  A LOW
NE       EQU   7                  A NOT EQUAL B
EQ       EQU   8                  A EQUAL B
GE       EQU   11                 A NOT LOW
LE       EQU   13                 A NOT HIGH
*
```

```
*********************************************************************
*        AFTER ARITHMETIC INSTRUCTIONS                              *
*********************************************************************
OV       EQU   1                   OVERFLOW
PLUS     EQU   2                   PLUS
MINUS    EQU   4                   MINUS
NZERO    EQU   7                   NOT ZERO
ZERO     EQU   8                   ZERO
ZEROS    EQU   8                   ZERO
NMINUS   EQU   11                  NOT MINUS
NOV      EQU   12                  NOT OVERFLOW
NPLUS    EQU   13                  NOT PLUS
*
*********************************************************************
*        AFTER TEST UNDER MASK INSTRUCTIONS                         *
*********************************************************************
ALLON    EQU   1                   ALL ON
MIXED    EQU   4                   MIXED
NALLOFF  EQU   5                   ALLON+MIXED
ALLOFF   EQU   8                   ALL OFF
NALLON   EQU   12                  ALLOFF+MIXED
*
*********************************************************************
*        GENERAL PURPOSE REGISTERS                                  *
*********************************************************************
R0       EQU   0                   PARM REGISTER
R1       EQU   1                   PARM REGISTER
R2       EQU   2                   SSOB
R3       EQU   3                   SSIB
R4       EQU   4                   SSVI
R5       EQU   5                   SSVI SIZE USED
R6       EQU   6                   SSVI SIZE NEEDED
R7       EQU   7                   UNUSED
R8       EQU   8                   ABEND VALUE REGISTER
R9       EQU   9                   IEFSSREQ/SSVI RETURN CODES
R10      EQU   10                  UNUSED
R11      EQU   11                  UNUSED
R12      EQU   12                  SSIREQ54 BASE REGISTER
R13      EQU   13                  SAVE AREA CHAIN REGISTER
R14      EQU   14                  PARM REGISTER / RETURN ADDR
R15      EQU   15                  PARM REGISTER / COND CODE
*
*********************************************************************
*        ABEND EQUATES                                              *
*********************************************************************
SSVIA101 EQU   101                 IEFSSREQ MACRO RETURNED R15
*                                   NON-ZERO
SSVIA102 EQU   102                 SSOBRETN IS NON-ZERO BUT NOT
*                                   EQUAL TO SSVIERR
         TITLE '- CVT - COMMUNICATIONS VECTOR TABLE'
         CVT DSECT=YES,LIST=NO
         TITLE '- IEFJESCT - JES CONTROL TABLE'
         IEFJESCT TYPE=DSECT
         TITLE '- IEFJSSIB - SUBSYSTEM IDENTIFICATION BLOCK'
         IEFJSSIB
```

```
         TITLE '- IEFSSOBH - SUBSYSTEM OPTION BLOCK HEADER'
         IEFSSOBH
SSOBGN   EQU   *             REQUIRED IF NOT USING IEFJSSOB DEFN
         TITLE '- IEFSSVI  - SUBSYSTEM VERSION INFORMATION'
         IEFSSVI
         TITLE '- LDA - LOCAL DATA AREA DSECT'
*********************************************************************
*        THE LOCAL DATA AREA IS MAPPED IN THIS DSECT.  THIS DATA    *
*        AREA IS OBTAINED THROUGH A 'STORAGE' MACRO INSTRUCTION     *
*        IN THE PROGRAM.                                            *
*********************************************************************
         SPACE ,
LDAAREA  DSECT
LDASTART EQU   *                   START OF LOCAL DATA AREA
LDASA    DS    9D                  SAVE AREA FOR LOWER CALLERS
LDAID    DS    CL8'LDAAREA '       IDENTIFICATION OF LDA AREA
LDA@SSOB DS    F                   POINTER TO SSOB FOR IEFSSREQ'S USE
LDASSOB  DC    XL(SSOBHSIZ)'00'    AREA FOR SSOB
LDASSIB  DC    XL(SSIBSIZE)'00'       AND SSIB
LDAEND   EQU   *                   START OF LOCAL DATA AREA
LDASIZE  EQU   LDAEND-LDASTART     LENGTH OF AREA TO GETMAIN
         TITLE '- HOUSEKEEPING REENTRANT ENTRY ROUTINE'
*********************************************************************
*        HOUSEKEEPING AND GENERAL ENTRY ROUTINE (REENTRANT USING    *
*        LINKAGE-STACK METHOD)                                      *
*********************************************************************
SSIREQ54 CSECT
         BAKR  R14,0               SAVE CALLER'S ARS, GPRS, AND
*                                   RETURN ADDRESS ON LINKAGE STACK
         LR    R12,R15             SET UP PROGRAM BASE REGISTER
         USING SSIREQ54,R12        INFORM ASSEMBLER
         STORAGE OBTAIN,           GET A SAVE AREA THAT'S REENTRANT    X
               LENGTH=LDASIZE,     STANDARD SAVE AREA SIZE             X
               COND=NO             UNCONDITIONAL REQ - NO RC INFO
         SPACE ,
         LR    R13,R1              SAVE STORAGE ADDRESS
         USING LDASTART,R13        ADDRESS LOCAL DATA AREA (LDA)
         MVC   LDAID,=CL8'LDAAREA'   INDICATION OF LOCAL DATA AREA
         WTO   'SSIREQ54 EXECUTING V1', LET OP KNOW                    X
               ROUTCDE=(2,11)
         TITLE '- SSOB/SSVI PROCESSING ROUTINE'
*********************************************************************
*        SET UP SSOB, SSIB, AND SSVI CONTROL BLOCKS.                *
*********************************************************************
         SPACE 2
*********************************************************************
*        OBTAIN STORAGE FOR AN SSVI.                                *
*********************************************************************
         LA    R5,SSVIMSIZ         MINIMUM SIZE REQUIRED
TRYIT    DS    0H
         STORAGE OBTAIN,           GET A SAVE AREA THAT'S REENTRANT    X
               LENGTH=(5),         STANDARD SAVE AREA SIZE             X
               COND=NO             UNCONDITIONAL REQ - NO RC INFO
         LR    R4,R1               POINT TO THE SSVI
         USING SSVI,R4             ADDRESSABILITY
         SPACE 2
```

```
*********************************************************************
*        WHEN ISSUING THE IEFSSREQ MACRO, REGISTER 1 MUST POINT TO  *
*        A CONTROL BLOCK THAT HAS IT'S HIGH-ORDER BIT SET, AND IT'S *
*        LOW-ORDER 31 BITS POINTING TO THE SSOB FOR THE SPECIFIC    *
*        FUNCTION CALL.  THEREFORE, SET THIS CONTROL BLOCK          *
*        (LDA@SSOB) WITH THE HIGH ORDER BIT SET, AND THE LOW-ORDER  *
*        31 BITS POINTING TO LDASSOB FIELD.                         *
*********************************************************************
         SPACE ,
         LA    R2,LDASSOB          POINT TO THE SSOB
         USING SSOB,R2             ADDRESSABILITY
         O     R2,=A(EQUHOBON)     SET HIGH ORDER BIT ON
         ST    R2,LDA@SSOB         STORE FOR IEFSSREQ'S USE
*                                   LATER WHEN ISSUING MACRO
*********************************************************************
*        NOW PROCESS THE SSOB (THE SUBSYSTEM OPTION BLOCK).         *
*********************************************************************
         SPACE ,
         XC    SSOBEGIN(SSOBHSIZ),SSOBEGIN  CLEAR THE SSOB
         MVC   SSOBID,=C'SSOBID'   MOVE IDENTIFIER IN
         MVC   SSOBLEN,=Y(SSOBHSIZ)  MOVE SIZE OF THE HEADER IN
         LA    R1,LDASSIB          POINT TO THE SSIB
         ST    R1,SSOBSSIB         SAVE IN SSOB
         MVC   SSOBFUNC,=Y(SSOBSSVI) MOVE THE FUNCTION ID IN
         ST    R4,SSOBINDV         SAVE SSVI ADDRESS IN SSOB
*********************************************************************
*        DONE WITH THE SSOB - NOW WORK WITH THE SSIB.               *
*        THE SSIB IS USED TO IDENTIFY THE SPECIFIC SUBSYSTEM THAT   *
*        THIS REQUEST IS GOING TO.  WE ISSUE OUR REQUEST TO THE     *
*        MASTER SUBSYSTEM, SO WE NEED TO PROVIDE ONE RATHER THAN    *
*        USE THE LIFE-OF-JOB SSIB WHICH COULD BE USED IF RUNNING    *
*        UNDER JES2.                                                *
*********************************************************************
         SPACE ,
         LA    R3,LDASSIB          POINT TO THE SSIB
         USING SSIB,R3             ADDRESSABILITY
         XC    SSIBEGIN(SSIBSIZE),SSIBEGIN  CLEAR SSIB
         MVC   SSIBID,=C'SSIBID'   MOVE IDENTIFIER IN
         MVC   SSIBLEN,=Y(SSIBSIZE)  MOVE SIZE OF THE SSIB IN
         MVC   SSIBSSNM,=C'MSTR'   SHOW MASTER SUBSYSTEM TO BE
*                                   USED TO GET THE INFO
*********************************************************************
*        DONE WITH THE SSIB - NOW WORK WITH THE SSVI.               *
*        THE SIZE CAN BE VARIABLE, SO WE NEED TO USE DYNAMIC SIZING *
*        TECHNIQUES WHEN CLEARING IT.                               *
*********************************************************************
         SPACE ,
         LR    R15,R5              SIZE OF THE SSVI
         BCTR  R15,0               DECREMENT FOR EX
         EX    R15,CLEAR           CLEAR THE SSVI
         STH   R5,SSVILEN          SAVE THE SIZE OF THE SSVI
         MVI   SSVIVER,SSVICVER    MOVE CURRENT VERSION NUMBER IN
         MVC   SSVIID,=A(SSVICID)  SAVE THE IDENTIFIER
         TITLE '- ISSUE IEFSSREQ'  ON IT''S WAY'
```

```
*********************************************************************
*        THE SSOB, SSIB, AND SSVI BLOCKS ARE NOW FILLED IN, AND THE *
*        IEFSSREQ MACRO IS READY TO GO.                             *
*********************************************************************
         SPACE 2
*********************************************************************
*        SET REGISTER ONE SO THAT IT POINTS TO POINTER OF THE SSOB  *
*********************************************************************
         SPACE ,
         LA    R1,LDA@SSOB         R1 POINTS TO ADDRESS OF SSOB
*********************************************************************
*        ISSUE THE IEFSSREQ REQUEST TO THE SUBSYSTEM.  NOTE WE      *
*        DON'T HAVE TO MODESET TO SUPERVISOR STATE; PROBLEM STATE   *
*        IS FINE FOR THIS SUBSYSTEM VERSION INFORMATION CALL.       *
*********************************************************************
         SPACE ,
         IEFSSREQ ,                GO GET THE VERSION INFORMATION
         SPACE ,
*********************************************************************
*        NOW CHECK THE RESULTS - HOW DID WE DO?                     *
*********************************************************************
         SPACE ,
         LA    R8,SSVIA101         ASSUME R15 NON-ZERO
         LTR   R9,R15              DID R15=0?  SAVE IN REG9 AS WELL
         BC    NZERO,ABEND         NO...GO TAKE A DUMP
         LA    R8,SSVIA102         ASSUME SSOBRETN NON-ZERO
         ICM   R9,B'1111',SSOBRETN CHECK SSOBRETN
         BC    ZERO,SHOWUSER       SEEMS OK - SHOW WHAT WE GOT
         C     R9,=A(SSVINSTR)     SPECIAL NOT ENOUGH
*                                   STORAGE CASE?
         BC    NE,ABEND            NO, TAKE A DUMP
         SPACE ,
*********************************************************************
*        THE IEFSSREQ MACRO WORKED OK, BUT THERE WASN'T ENOUGH      *
*        STORAGE DEFINED TO RECEIVE ALL OF THE INFORMATION.  USING  *
*        THE INFORMATION RETURNED, LET'S TRY AGAIN.                 *
*********************************************************************
         SPACE ,
         LH    R6,SSVIRLEN         SAVE THE STORAGE NEEDED
         STORAGE RELEASE,          FREE MY INFO AREA                  X
               LENGTH=(5),         VARIABLY OBTAINED SIZE             X
               ADDR=(4)            HERE'S WHERE IT LIVES
         LR    R5,R6               NEW SIZE TO TRY AGAIN
         B     TRYIT               GO DO IT TO DO!
         DROP  R2                  SSOB
         TITLE '- EXIT ROUTINES TO MVS (BOTH GOOD AND BAD)'
*********************************************************************
*        THESE ARE GENERAL EXIT ROUTINES BACK TO MVS.               *
*        ABENDS ARE USED FOR THE ABNORMAL TERMINATIONS.             *
*********************************************************************
```

```
         SPACE 2
SHOWUSER DS    0H
         ICM   R6,B'1111',SSVIUDOF ANY USER DATA?
         BC    ZERO,SHOWSYS        NO, SHOW THE SYSTEM DATA
         LA    R7,SSVI(R6)         R7==>USER VARIABLE DATA AREA
         USING SSVIVDAT,R7         ADDRESSABILITY
         LH    R8,SSVIVLEN         GET THE LENGTH
         CH    R8,=H'125'          GREATER THAN 125 CHARS?
         BC    LE,SHOWIT1          NO, USE THE REAL LENGTH
         MVC   SSVIVLEN,=H'125'    ELSE, USE ONLY FIRST 125
SHOWIT1  DS    0H                  R8=NUMBER OF CHARS TO DISPLAY
         WTO   TEXT=SSVIVLEN,      SHOW TO THE CONSOLE               X
               ROUTCDE=(11)
         B     SHOWSYS2            BRANCH AROUND WTO
SHOWSYS  DS    0H
         WTO   'SSIREQ54 NO USER DATA PRESENT',   LET OP KNOW        X
               ROUTCDE=(2,11)
SHOWSYS2 DS    0H
         ICM   R6,B'1111',SSVISDOF ANY SYSTEM DATA?
         BC    NZERO,SHOWSYS3      YES, DISPLAY IT
         WTO   'SSIREQ54 NO SYSTEM DATA', LET OP KNOW                X
               ROUTCDE=(2,11)
         B     RETURN
SHOWSYS3 DS    0H
         LA    R7,SSVI(R6)         R7==>USER VARIABLE DATA AREA
         USING SSVIVDAT,R7         ADDRESSABILITY
         LH    R8,SSVIVLEN         GET THE LENGTH
         CH    R8,=H'125'          GREATER THAN 125 CHARS?
         BC    LE,SHOWIT2          NO, USE THE REAL LENGTH
         MVC   SSVIVLEN,=H'125'    ELSE, USE ONLY FIRST 125
SHOWIT2  DS    0H                  R8=NUMBER OF CHARS TO DISPLAY
         WTO   TEXT=SSVIVLEN,      SHOW TO THE CONSOLE               X
               ROUTCDE=(11)
         SPACE ,
         WTO   'SSIREQ54 RETURNING',      LET OP KNOW                X
               ROUTCDE=(2,11)
         SPACE ,
*********************************************************************
*        GIVE BACK THE STORAGE WE BOUGHT EARLIER.                   *
*********************************************************************
         SPACE ,
RETURN   DS    0H
         STORAGE RELEASE,          FREE MY INFO AREA                 X
               LENGTH=(5),         VARIABLY OBTAINED SIZE            X
               ADDR=(4)            HERE'S WHERE IT LIVES
         STORAGE RELEASE,          FREE MY REENTRANT SAVE AREA       X
               LENGTH=LDASIZE,     STANDARD SAVE AREA SIZE           X
               ADDR=(R13)          HERE'S WHERE IT LIVES
         SPACE ,
*********************************************************************
*        SET PROGRAM RETURN CODE.                                   *
*********************************************************************
```

```
         SPACE ,
         SLR   R15,R15             SET RETURN CODE OF ZERO
*********************************************************************
*        RETURN TO CALLER WITH ORIGINAL STATUS AND REGISTERS.       *
*********************************************************************
         SPACE ,
         PR                        RETURN TO CALLER USING STACK,       X
                                    RESET REGS 2-14, ADDRESSING MODE,  X
                                    ASC MODE, AND RETURN TO CALLER
*********************************************************************
*        ABEND ROUTINES FOLLOW                                      *
*********************************************************************
         SPACE ,
ABEND    DS    0H                  R15 NON-ZERO AFTER IEFSSREQ
         WTO   'PROGRAM HAD FATAL ERROR - SEE REGS 8 AND 9'            X
               ROUTCDE=(2,11)
         SPACE ,
         ABEND (R8),DUMP,STEP      LET THE USER IN ON THE BAD NEWS
         TITLE '- LOCAL DATA'
         SPACE ,
CLEAR    XC    0(*-*,R4),0(R4)     CLEAR SSVI - OBJ OF EXECUTE
         END   ,
```

Product-sensitive programming interface

Notify User Message Service Call — SSI Function Code 75

The Notify User Message Service Call (SSI function code 75) provides a requesting program the ability to send a message to other users who are either:

- On the same networking node
- On another node.

Type of Request

Directed SSI call.

Use Information

When a caller issues SSI function code 75 to send a message through networking facilities, the requesting program uses network job entry (NJE) services provided by MVS/JES. In an MVS environment, the TSO/E user is typically the recipient of these messages. For example, when a program reaches a particular place in its processing that the user wants to know about, the caller issues the SSI function code 75, and a message is sent to the user notifying them of this event. The text of this message is free-form.

Issued to

- The primary subsystem, either JES2 or JES3
- A secondary JES2 subsystem.

Related SSI Codes

None.

Related Concepts

None.

Environment

The caller (issuer of the IEFSSREQ macro) must include the following mapping macros:

- CVT
- IEFJESCT

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your program:

- IEFSSOBH
- IEFJSSIB
- IAZSSNU

The caller must meet the following requirements:

Minimum Authorization	Supervisor state
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts

Locks	No locks held
Control Parameters	The SSOB, SSIB, and SSNU control blocks can reside in storage above 16 megabytes.
Recovery	The caller should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on an ESTAE-type recovery environment.

Figure 10 shows the environment at the time of the call for SSI function code 75.

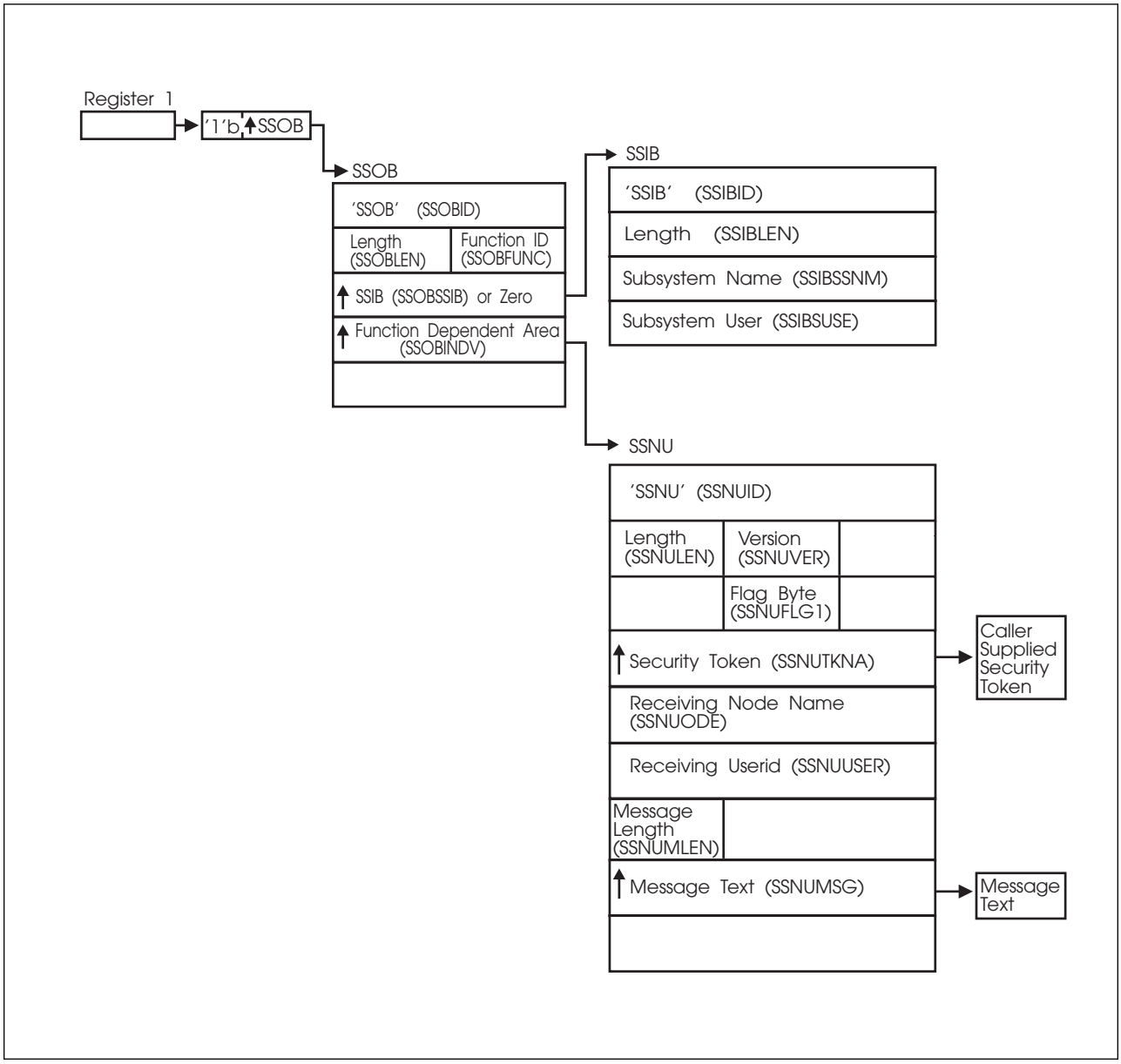

Figure 10. Environment at Time of Call for SSI Function Code 75

Input Register Information

Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

Register	Contents
1	Address of a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.
13	Address of a standard 18-word save area.

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSNU

SSOB Contents: The caller sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function code 75 (SSOBSSNU)
SSOBSSIB	Address of the SSIB control block or zero (If this field is zero, the life-of-job SSIB is used). See “Subsystem Identification Block (SSIB)” on page 8 for more information on the life-of-job SSIB.
SSOBINDV	Address of the function dependent area (SSNU control block)

Set all other fields in the SSOB control block to binary zeros before issuing the IEFSSREQ macro.

SSIB Contents: If you don't use the life-of-job SSIB, the caller must provide an SSIB and set the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of the subsystem to which this Notify User Message Service call is directed. It is usually the primary JES, or in the case of JES2, a possible secondary JES. If your routine has not been initiated from such a JES, the caller must issue a Request Job ID call (SSI function code 20) prior to this Notify User Message Service call. You must use the same subsystem name in this SSIBSSNM field as you used for the Request Job ID call.
SSIBSUSE	(JES3 only) Subsystem use — the SSIBSUSE value that was returned upon completion of the Request Job ID call (SSI function code 20).

The caller must set all other fields in the SSIB control block to binary zeros before issuing the IEFSSREQ macro.

SSNU Contents: The caller sets the following fields in the SSNU control block on input:

Field Name	Description
SSNUID	Identifier 'SSNU'
SSNULEN	Length of the SSNU (SSNUSIZE) control block
SSNUVER	Version of mapping for the caller — Set this field to SSNUCVER (an IBM-defined integer constant within the SSNU control block).
SSNUFLG1	Flag Byte • **SSNU1MLO** — logon message flag If SSNU1MLO is set, a message is issued only if the user is logged on.
SSNUTKNA	Associated security token of issuing user — this field is optional. If specified, the SSNUTKNA field must point to a valid security token, and a WRITER class call is made to validate that the user has the authority to issue messages that NJE sends.
SSNUNODE	Node that messages are sent to — If homenode is desired, use binary zeros in the SSNUNODE field (Do not use blank (X'40') characters).
SSNUUSER	Userid to which messages are sent.
SSNUMLEN	Length of the message pointed to by the SSNUMSG field. The message must be no greater than 100 characters.
SSNUMSG	Address of the EBCDIC data message that is issued.

Set all other fields in the SSNU control block to binary zeros before issuing the IEFSSREQ macro.

Output Register Information

When control returns to the caller, the general purpose registers contain:

Register	Contents
0	Used as a work register by the system
1	Address of the SSOB control block
2 — 13	Same as on entry to call
14	Return address
15	Return code

Return Code Information

The SSI places one of the following decimal return codes in register 15. Examine the return code to determine if the request was processed.

Return Code

(Decimal)	Meaning
SSRTOK (0)	The Notify User Message Service request was processed. Check the SSOBRETN field for specific function information.
SSRTNSUP (4)	The subsystem specified in the SSIBSSNM field does not support this function.
SSRTNTUP (8)	The subsystem specified in the SSIBSSNM field exists, but is not active.
SSRTNOSS (12)	The subsystem specified in the SSIBSSNM field is not defined to MVS.
SSRTDIST (16)	The pointer to the SSOB control block or the SSIB control block is not valid, or the function code specified in the SSOBFUNC field is greater than the maximum number of functions supported by the subsystem specified in the SSIBSSNM field.
SSRTLERR (20)	Either the SSIB control block or the SSOB control block has invalid lengths or formats.
SSTRNSSI(24)	The SSI has not been initialized.

Output Parameters

Output parameters for the function routine are:

- SSOBRETN
- SSNU

SSOBRETN Contents: When control returns to the caller and register 15 contains a zero, the SSOBRETN field contains one of the following decimal values:

Value (Decimal)	Meaning
SSNUOK (0)	The message was issued successfully. The SSNUERCD field contains a zero (SSNURQOK).
SSNUOKB (4)	The message was issued successfully but had a minor error. See the SSNUERCD field in the SSNU control block for the specific reason code.
SSNUERR (8)	The message was not issued. See the SSNUERCD field in the SSNU control block for the specific reason code.
SSNUNEX (12)	The functional extension to the SSNU control block was not found in the SSOB (the SSOBINDV field was zero) control block.

SSNU Contents: The SSNUERCD (error code) field in the SSNU control block contains one of the following decimal values if the SSOBRETN field was set to either SSNUOKB or SSNUERR on return from the IEFSSREQ macro:

Value (Decimal)	Meaning
SSNURQOK (0)	The request was successful.
SSNUMSGT (4)	The request was successful, but the message text was truncated because it was too long.
SSNUEXC (8)	A user exit cancelled the request (JES2 only). In JES2, exit 42 may have requested the cancellation of the message.

SSNUNUSR (12) An invalid userid was specified (blanks or zeros).

SSNUINVD (16) An invalid nodename was supplied. The message was not issued.

SSNUIVID (20) An invalid identifier (SSNUID) was supplied. The message was not issued.

SSNUIVER (24) An invalid version of the SSNU control block was supplied. The message was not issued. The value supplied in the SSNUVER field could not be processed by JES (either because it was zero, or it was at a higher level than the receiving JES can process (SSNUCVER)).

SSNUNOST (28) Storage in the processing subsystem was not available for the function. The message was not issued.

SSNUNOAU (32) The supplied token failed an NJE WRITER class authorization call. The caller is not allowed to issue messages to the specified node. The message was not issued.

SSNUMSGE (36) The supplied message address or length was not valid (address specified was zero). The message was not issued.

End of Product-sensitive programming interface

SYSOUT Application Program Interface (SAPI) — SSI Function Code 79

The SYSOUT Application Program Interface (SSI function code 79) allows JES to function as a server for applications needing to process SYSOUT data sets residing on JES spool. Use of the SAPI SSI call allows a user-supplied program to access JES SYSOUT data sets independently from the normal JES-provided functions (such as print or network). Users of this function are application programs operating in address spaces external to JES. SAPI supports multiple, concurrent requests from the applications' address spaces. Each issuer of the IEFSSREQ macro is referred to as an “application thread.”

Differences Between SSI Function Codes 1 and 79

Although both the SYSOUT Application Program Interface (SSI Function Code 79) and Process SYSOUT (SSI Function Code 1) allow applications to retrieve SYSOUT from JES spool using a variety of criteria, there are several important differences between the two function calls. IBM recommends that applications use the SAPI, as it is richer in function, as well as having better performance characteristics than the Process SYSOUT Call.

Some of the differences that SAPI provides are:

- The ability to multi-task data set selection and processing calls from within an application.
- A richer selection criteria, including the use of wildcard characters for attributes.
- A greater number of SYSOUT data set characteristics returned to the application than does Process SYSOUT.
- The application has the ability to retrieve information contained in the scheduler work blocks (SWBs)
- A greater degree of modification ability of selected SYSOUT data sets.
- A count facility that Process SYSOUT does not provide.

Requesting SAPI Processing

The IAZSSS2 (SSS2) mapping macro is used as input to the IEFSSREQ request for SAPI processing. Fields in the SSS2 macro are differentiated into input, output, and disposition fields.

- An issuer's application thread sets input fields upon each IEFSSREQ invocation.
- JES manages output fields.

  JES updates the output-defined fields in response to each IEFSSREQ invocation.
- An issuer's application thread sets the disposition fields on an obtain data set request to inform JES of the disposition processing to occur for the data set returned on the prior obtain data set request.

SYSOUT Application Program Interface Request Types

An application thread can make three types of requests with SAPI. Each is independent of, and mutually exclusive with the others. Field SSS2TYPE indicates which of these three possible types of requests the application thread is issuing:

- SSS2PUGE - indicates a SAPI PUT/GET request
- SSS2COUN - indicates a SAPI COUNT request

- SSS2BULK - indicates a SAPI BULK MODIFY request

This is the function each serves:

- PUT/GET

  Initiates data set selection, and optionally can provide disposition processing for the data set returned in the previous SAPI PUT/GET call. The SAPI PUT/GET call is described on “PUT/GET Requests” on page 88.

- COUNT

  Returns the count of entries that can be scheduled without returning a particular data set. The SAPI COUNT call is described on “COUNT Requests” on page 93.

- BULK MODIFY

  Modifies selected attributes of one or more data sets. The SAPI COUNT call is described on “BULK MODIFY Requests” on page 95.

General Programming Considerations — Applicable to All Calls

The following considerations apply to any of the three types of SAPI (SYSOUT application program interface) calls (PUT/GET, COUNT, and BULK MODIFY):

- Each unique SSOB/SSS2 pair supplied as input on the IEFSSREQ request is viewed as a separate thread by JES.

  You can multi-task these requests within your application's address space, or even issue multiple IEFSSREQ requests (supplying different SSOB/SSS2 pairs) from within a single task in your application's address space. A task that issues the original IEFSSREQ can transfer the SSOB/SSS2 control block pair to another task within your address space for subsequent IEFSSREQ requests. However, if this is done and the originating task (which JES considers to be the owner of that specific thread) fails, then JES cleanup occurs for resources associated with that SSOB/SSS2 pair. If the transferred task attempts to issue another IEFSSREQ with that same SSOB/SSS2 pair after such a termination occurs, incorrect processing occurs because JES has disconnected from that SSOB/SSS2 pair.

  The field SSS2JEST is the binding value that JES uses to associate a specific SSOB/SSS2 pair to its thread. The owner of a thread is the TCB that makes the **FIRST** request and receives a token in field SSS2JEST. After initially setting SSS2JEST to X'00's as part of the application thread's original initialization of the SSS2, the application thread cannot modify or refer to the SSS2JEST.

- The ‘output section’ of the SSS2 is initialized once by the application thread. The application thread does so by clearing the entire SSS2 with binary zeroes prior to initializing any input fields and then issuing the first IEFSSREQ request. Subsequently, JES manages all the output section fields. An application thread can only change the contents of this output section after an IEFSSREQ request has been made with the SSS2CTRL flag set. JES considers such a subsequent request as a new thread because as a result of the SSS2CTRL bit being set on the prior IEFSSREQ call, JES disassociates all JES-maintained resources held.

- Destination fields can include a single, maximum 8-character destination or a destination in the format of node.userid. For the latter case you must have an NJE-defined destination as the node. The fields are:

  - SSS2DEST (Destination - selection)
  - SSS2DES2 (New Destination - BULK MODIFY)
  - SSS2DDES (New Destination - Disposition Processing)
  - SSS2DESR (Returned Destination from a SAPI PUT/GET Call)
- When the selection destination field (SSS2DEST) is in the form of `A.B`, the `A` portion can **not** be an NJE-defined node other than the node on which the application is running.
- When the modification destination field (SSS2DES2 or SSS2DDES) is in the form of `A.B`, the `A` portion **can** be an NJE defined node. In this case, the SYSOUT is sent to user 'B' at node 'A'.
- Wildcards are valid for the following SSS2 selection fields:
  - SSS2JOBN (Job Name)
  - SSS2CREA (Owning Userid)
  - SSS2PRMO (Process Modes)
  - SSS2DEST (Destination)
  - SSS2PGMN (User Writer Name)
  - SSS2FORM (Form Numbers)

  Valid wildcards are `*` for multiple characters and `?` for a single character.
- Output field SSS2RET2 indicates which of the input selection fields were not used by JES in the selection of work.
- The SSI Function Code 54 call (Request Subsystem Version Information) can be used to determine the appropriate SYSOUT class to use when modifying the data set's SYSOUT class through the SAPI BULK MODIFY call.
- In the terminology of SAPI, the term 'null' refers to fields in the SSS2 that are either `X'40'`s (EBCDIC blanks) in the case of character data, or `X'00'`s (all zeroes) in the case of binary data.
- JES provides a minimum amount of input validity checking of an input SSS2 before a final call (SSS2CTRL) is processed. This validity checking includes:
  - Ensures a valid SSS2 eye catcher is present
  - Ensures a valid version number is present
  - Ensures a valid request type is present
  - Ensures a valid length is present
  - Ensures a valid disposition, if applicable, is present
- Data sets available for selection are those that are available at the time the search for a data set matching the selection criteria begins. Therefore, if a data set matching the selection criteria is created while a search is in progress, it is possible that the data set will not be found during that search.
- Data sets available for selection are those that are not currently being processed.
- The use of the token returned from Extended Status (SSI 80) can result in an EOD return code (SSS2EODS) returned to the user. This can happen when the SYSOUT available at the time Extended Status was used had been processed before this call was made (SSS2RENM) or is currently being processed (SSS2RENS).

PUT/GET Requests

PUT/GET request processing occurs when an application thread issues the IEFSSREQ macro to initiate data set selection. The input SSOB and SSS2 control blocks, provided by the application thread, specifies the selection criteria used to select a data set. The application thread can use a wide variety of selection criteria to select a SYSOUT data set to be processed.

Once the application thread receives a data set from the JES, you must allocate (through a dynamic allocation with the data set name that is returned from SSS2DSN) the data set to process it. During this allocation, dynamic allocation requires DALBRTKN text unit. JES performs the initialization of this text unit. The application thread must move the address from field SSS2BTOK into a text unit pointer field for the JES-provided DALBRTKN text unit. The actual processing of the SYSOUT data set depends upon your specific application. After your application thread has completed processing of the data set, it then unallocates the data set with the text unit of DUNDDNAM specifying the DDNAME of the returned data set from the original allocation. The allocation/unallocation of the data set must occur once per returned data set.

The PUT processing occurs when the application thread subsequently issues a following IEFSSREQ macro to **select another** data set. You can use fields in the optional disposition section **of the SSS2** to change certain attributes of the **previously obtained** data set from the prior IEFSSREQ call.

A difference between SAPI and Process SYSOUT (SSI Function Code 1) during unallocation is that SAPI does not process any of the unallocation text units as occurs in Process SYSOUT. The SSS2 provides specific disposition fields for JES to use during the subsequent SAPI PUT/GET call to provide for disposition processing. From a JES processing point of view, the disposition processing for the previous data set occurs prior to the processing of the selection of the next data set, but both are occurring within the same IEFSSREQ call by the application thread.

You must provide at least SAF `UPDATE` authority for the JESSPOOL resource class to the application thread to issue the SAPI PUT/GET call correctly.

If the application does not provide for multi-tasking, it must follow the protocol below. If the application does provide for multi-tasking, each application thread in the address space must follow the protocol shown in Figure 11 on page 89.

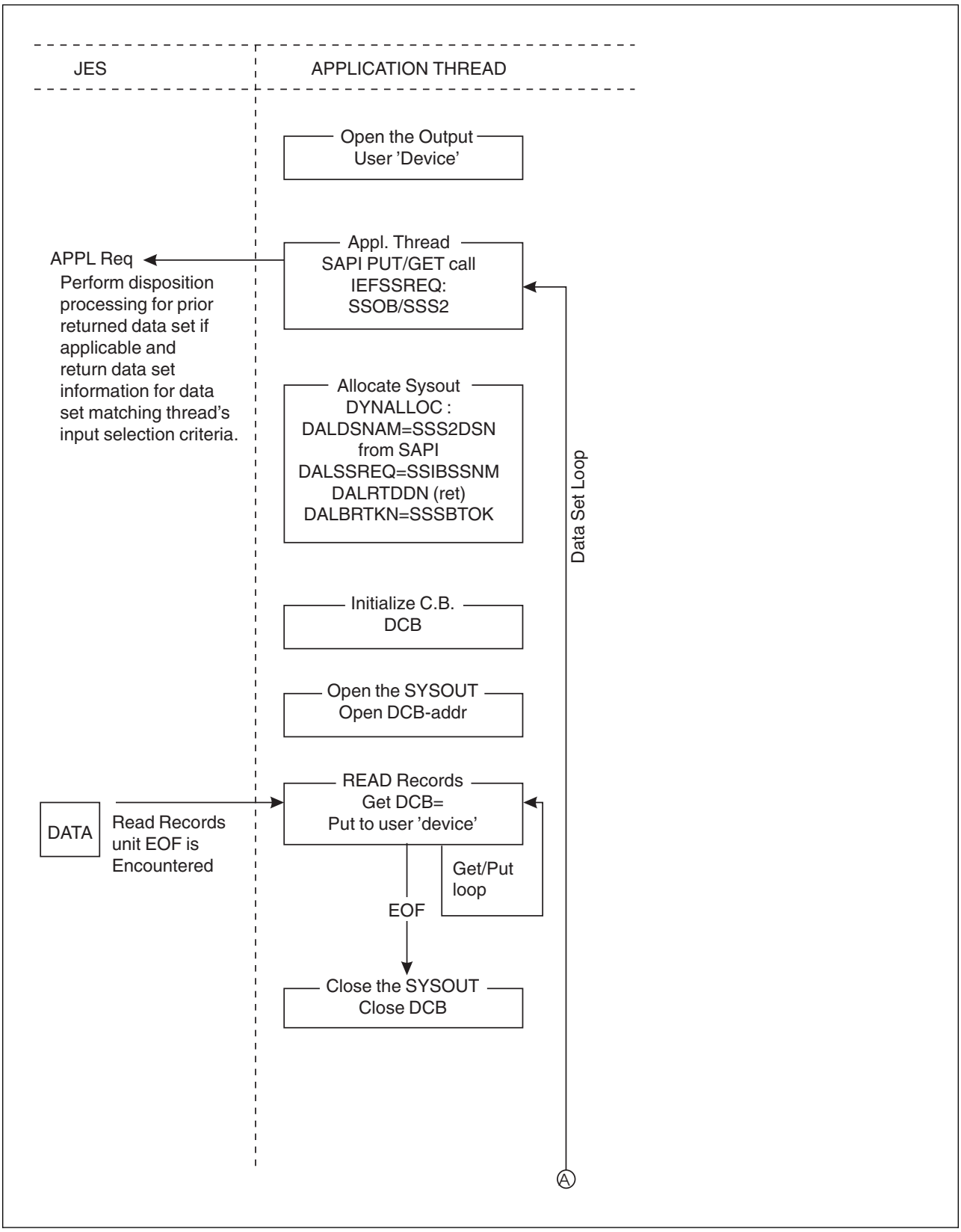

Figure 11 (Part 1 of 2). Protocol for the SAPI PUT/GET Call

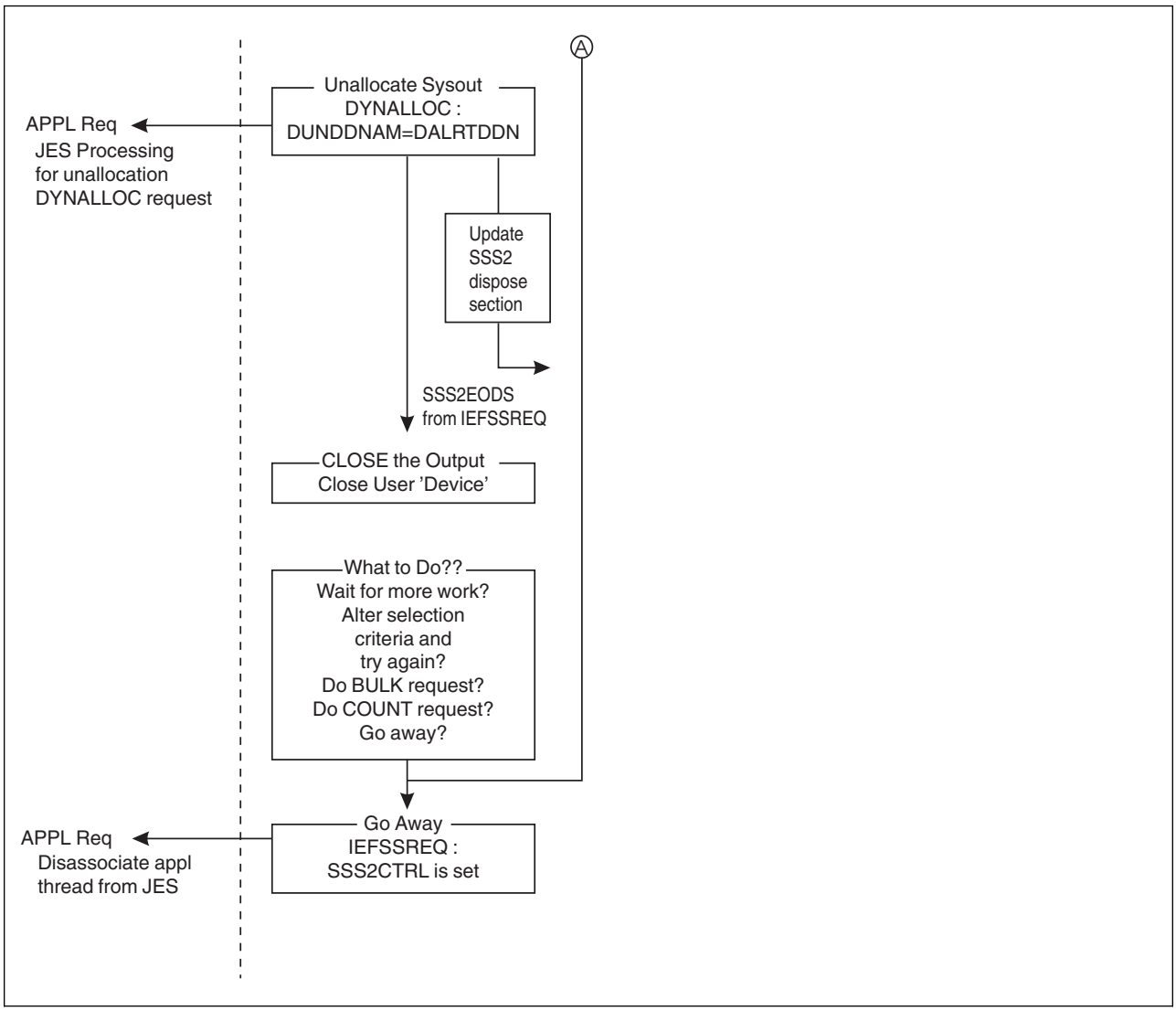

Figure 11 (Part 2 of 2). Protocol for the SAPI PUT/GET Call

Programming Considerations for PUT/GET

- The application thread must provide a pointer to an ECB in field SSS2ECBP if the application thread wants JES to post it when newly created work has characteristics matching the thread's selection criteria. This occurs after JES returns SSS2EODS for a PUT/GET request. If an ECB is not supplied, it is the responsibility of the thread to initiate an IEFSSREQ request.
- For JES3 only, once the application thread begins PUT/GET processing, a COUNT or BULK MODIFY request can not be initiated prior to receiving an SSS2EODS response to a PUT/GET request.
- SSS2CDS contains a 1 for the single returned data set in a SAPI GET/PUT call. If the data set disposition is DELETE, all copies of the data set are deleted.
- Information contained within the SYSOUT data set's scheduler work blocks (SWBs) can also be returned to the application thread. Much of the information contained within the SWB is normally not processed by JES, and therefore much more information about the data set can be retrieved from the SWB than

is returned in fields of the SSS2. Examples of such information contained within the SWB are NAME, BUILDING, ADDRESS, and so on.

The application thread needing to retrieve this SWB information, sets either SSS2FSWB or SSS2FSWT in flag byte SSS2MSC1 when issuing a PUT/GET request. The setting of SSS2FSWB implies SSS2FSWT processing as well. JES then provides the application thread the information that can be used when the application thread invokes the SJF services to retrieve this SWB information. These services are either SJFREQ REQUEST=RETRIEVE or SWBTUREQ REQUEST=RETRIEVE.

Note that the use of either settings cause JES to perform additional processing overhead to satisfy this request. Thus, the application thread should not request the SWB information unless needed by the application. Examples of this additional overhead are spool I/O to read the stored SWBTU blocks, SJF services that JES needs to invoke to prepare the environment, additional GETMAINs needed to satisfy the request.

If the application thread sets either SSS2FSWT or SSS2FSWB, JES returns in output field SSS2SWTU a single SWBTU that can be used as input to a subsequent SWBTUREQ REQUEST=RETRIEVE call made by the application thread. Mapping macro IEFSJTRP is used when issuing this SWBTUREQ request. Field SJTRSTUP can be set with the contents of SSS2SWTU when issuing this request. Set field SJTRSWBN with a binary 1 to indicate a single SWBTU block is being used for the SWBTUREQ call. The application thread does not need to explicitly provide storage for the SWBTU block or free it; that is JES's responsibility.

If the application thread sets SSS2FSWB, JES returns in output field SSS2SWBT an output descriptor token that can be used as input to a subsequent SJFREQ REQUEST=RETRIEVE call made by the application thread. This is in addition to the SSS2FSWT processing previously described. Mapping macro IEFSJREP is used when issuing this SJFREQ request. Field SJRETOKN can be set with the contents of SSS2SWBT when issuing this request. The application thread does not need to explicitly provide storage for the output descriptor token, or free it; that is JES's responsibility.

In the SSS2, reason code field SSS2WRTN contains either a value of SSS2WOK (0) or SSS2WERR (4). SSS2WOK indicates that JES processing needed for SWB retrieval was completely successful, and output fields SSS2SWBT and SSS2SWTU can be used as described above. If SSS2WRTN is set with SSS2WERR, then an error occured indicating **neither** SSS2SWTU or SSS2SWBT fields can be used. If this is the case, reason code field SSS2WRSN is set with an indicator of the type of error that prevented JES from providing the SWB information.

Note that this information provided is primarily to be used as diagnostic information, because the application thread can not affect the JES processing directly that led to the error. Accordingly, receiving such a SWB processing error does **not** affect the rest of JES processing. The data set is still able to be processed by the application thread; only the ability to issue either the SWBTUREQ or SJFREQ macro services by the application thread is affected and must not be attempted.

See *OS/390 MVS Programming: Authorized Assembler Services Reference SET-WTO* for additional information concerning the use of the SJFREQ and

SWBTUREQ services to retrieve the information in the SWB by either, or both, of the methods described.

- It is the responsibility of the application thread to understand the implications of disposing a data set as KEEP. Because of the potential to process the data set again, the application thread must ensure a loop condition does not arise.
- An EOD (SSOBRETN=SSS2EODS) response is a possible return only for PUT/GET processing. When SAPI returns SSS2EODS to the application thread, the application thread can do one of the following:
  - Wait on its supplied ECB for a post from JES. This post indicates SYSOUT has just been generated that contains characteristics matching the application thread's selection criteria.

    The application can then issue another IEFSSREQ to obtain this data set from the JES. Since multiple applications can be posted from the single piece of work appearing on the queue, there is no guarantee that once posted, a thread will not receive an immediate SSS2EODS return again (that is, another thread received the work).
  - Issue another IEFSSREQ request after changing its selection criteria.
  - Issue another IEFSSREQ request with the SSS2CTRL flag set indicating the application thread is terminating.
  - Issue a COUNT request.
  - Issue a BULK MODIFY request.
- The application must provide DALSSREQ (supplying the JES subsystem name (for example, `JES2` or `JESA` or `JES3`)) and a dynamic allocation text unit pointer that contains the address supplied in SSS2BTOK. In addition, your application thread must supply a text unit with DALDSNAM that uses the data set name returned in SSS2DSN.

  The subsequent dynamic allocation call is depicted in Figure 12 on page 93.

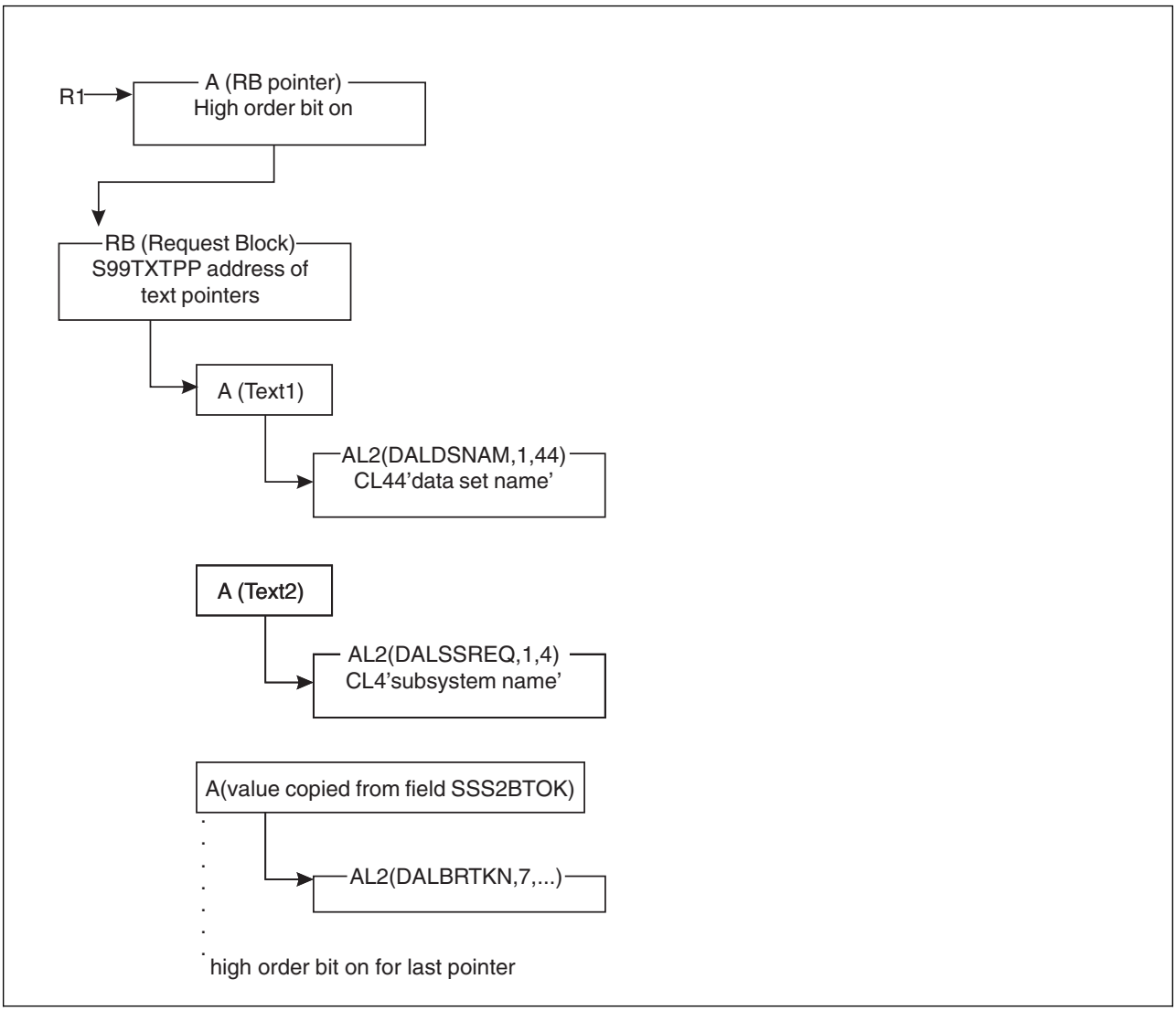

Figure 12. Control Blocks of DYNALLOC Call for SAPI-Provided Data Set

COUNT Requests

JES counts the number of schedulable elements (OSEs/JOEs) matching the input selection criteria and returns the count to the application thread in field SSS2CDS. An application thread does not receive a data set in the SAPI COUNT call. Included in the information returned are the total byte count, record count, line count, and page count.

There is **no** posting of the ECB after a COUNT request has been processed by JES.

If the application does not provide for multi-tasking, it must follow the protocol shown in Figure 13 on page 94. If the application does provide for multi-tasking, each thread in the application address space must follow the protocol shown in Figure 13 on page 94.

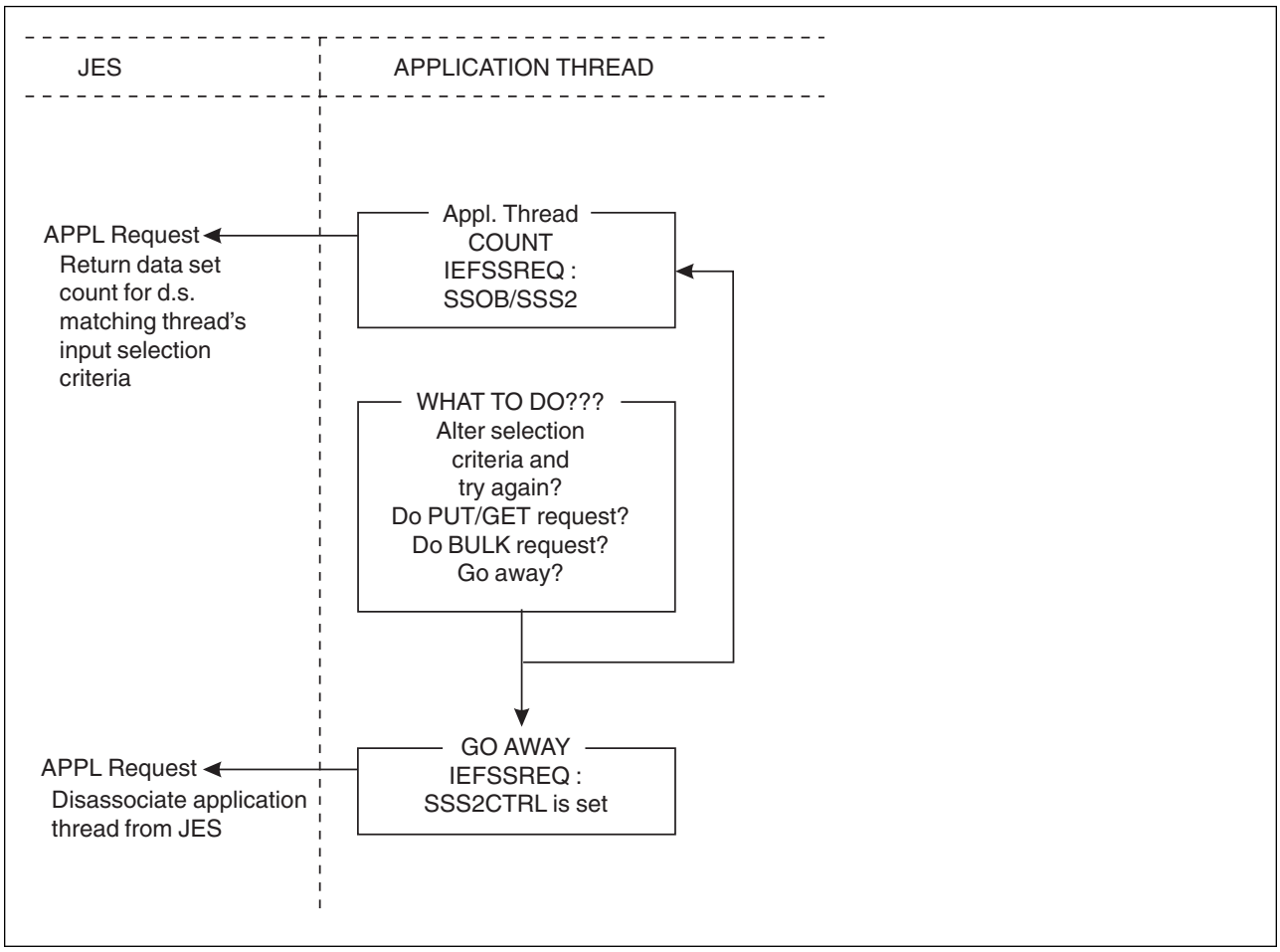

Figure 13. Protocol for the SAPI COUNT Call

Programming Considerations for COUNT

- Supplying an ECB address in field SSS2ECBP does not result in the posting of the ECB by JES for a COUNT request.
- A COUNT request can be initiated after the application thread initialization is complete, immediately following a prior COUNT request, immediately following a BULK MODIFY request or immediately following receiving an EOD response to a PUT/GET request.
- After JES returns to the thread after processing the COUNT request, the thread can do one of the following:
  - Issue another IEFSSREQ request, possibly after changing its selection criteria
  - Issue another IEFSSREQ request with the SSS2CTRL flag set indicating the application thread is terminating
  - Issue a BULK MODIFY request
  - Issue a PUT/GET request

BULK MODIFY Requests

With a BULK MODIFY request, the application thread can select SYSOUT data set(s) for modifications. Modification of data sets matching the input selection criteria occurs with the setting of information in flag byte SSS2UFLG.

- **SSS2SETC** - class update

  The class of each data set is changed to the specified class in the SSS2CLAS field.

- **SSS2DELC** - delete processing

  Each data set is deleted.

- **SSS2ROUT** - destination update

  The destination of each data set is changed to the specified destination in the SSS2DES2 field.

- **SSS2RLSE** - release processing

  Each data set is moved to the WRITER queue in JES3, and marked non-held in JES2.

  Release processing is applicable only to data sets on the JES3 Output Service HOLD queue, or for those data sets with dispositions of HOLD or LEAVE for JES2.

Processing for a BULK MODIFY request occurs for each data set matching the application thread's selection criteria. It is important to understand job boundaries can be crossed.

There is **NO** posting of the ECB after a BULK MODIFY request has been processed by JES.

You must provide at least SAF `UPDATE` authority for the JESSPOOL resource class to the application thread in order to correctly issue the SAPI BULK MODIFY call.

If the application does not provide for multi-tasking, it must follow the protocol shown in Figure 14 on page 96. If the application does provide for multi-tasking, each thread in the application address space must follow the protocol shown in Figure 14 on page 96.

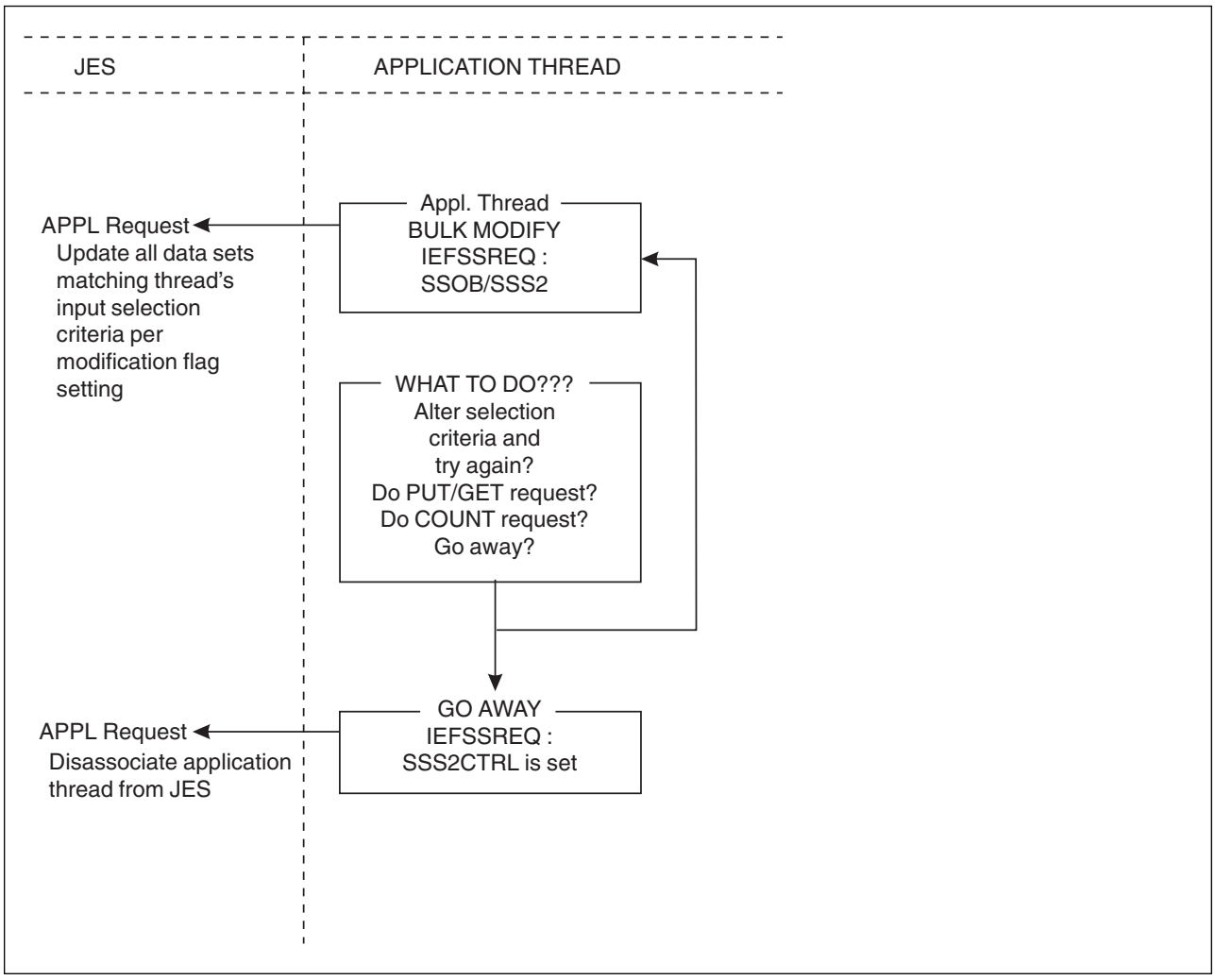

Figure 14. Protocol for the SAPI BULK MODIFY Call

Programming Considerations for BULK MODIFY

- Supplying an ECB address in field SSS2ECBP does not result in the posting of the ECB by JES for a BULK MODIFY request.
- A BULK MODIFY request can be initiated after the application thread initialization is complete, immediately following a prior BULK MODIFY request, immediately following a COUNT request or immediately following receiving an EOD response to a PUT/GET request.
- After JES returns to the application thread after processing the BULK MODIFY request, the application thread can do one of the following:
  - Issue another IEFSSREQ request, possibly after changing its selection criteria.
  - Issue another IEFSSREQ request with the SSS2CTRL flag set indicating the application thread is terminating.
  - Issue a COUNT request.
  - Issue a PUT/GET request.

Use of the Client Token

The contents of the token pointed to by field SSS2CTKN are created by JES. Using the token reduces the time to find the associated data set. Don't compare or otherwise use the tokens except on SAPI or Extended Status calls. Two different tokens obtained by different means may point to the same data set.

There are several ways to have obtained a token:

- A previous Extended Status request (see field STSTCTKN)
- As the output of a PUT/GET request (in field SSS2DSTR)
- Dynamic Allocation specified the DALRTCTK text unit.

The content of this SSS2CTKN field is used in addition to any other specified parameters. This way you can make sure the output data set still has the characteristics you would expect and have not been modified. If these characteristics are unimportant to you, specify SSS2CTKN as the only input parameter.

Keeping Processed Data Sets

SSS2RNPR on means that the JES will not return the data set to the application address space again. The application should treat this as a suggestion (not iron clad) to the JES. The data set could be seen again by the application if:

- The JES is restarted
- The application is restarted
- The operator or another application changes some characteristic.

SSS2RNPT on means that the JES will not return the data set to the application thread again. A thread begins with the first receipt of a token in field SSS2JEST and ends when the thread calls JES with the SSS2CTRL flag set. Other threads will be able to obtain the data set, provided their selection criteria allow it. The application should treat this as a suggestion (not iron clad) to the JES. The data set could be seen again by the thread if:

- The JES is restarted
- The operator or another application changes some characteristic
- Selection by token is requested.

This SSS2RNPT may be useful for applications that need to hold on to a data set or group of data sets until the data is processed by the requester. It allows for building a "pipeline" of work that is directed to the same processing device or user.

Another way to use the function may be in situations where the system needs to present a list of data sets (from the same job) and keep those data sets on SPOOL for later final inspection. An end user might want to browse all data sets from a job, regardless of output characteristic groupings. If only the KEEP disposition is specified, the same data set may eventually be shown to the application again, thus creating a never ending loop.

Type of Request
Directed SSI call.

Use Information
An application thread uses SSI function code 79 to retrieve and update JES-managed SYSOUT data sets, allowing the individual application thread to select SYSOUT from JES and process it in the manner the application thread desires.

Issued to
JES2 or JES3.

Related SSI Codes
54

Related Concepts
You should know how to use:

- Dynamic allocation (DYNALLOC) services to allocate/deallocate the JES-supplied data set.
- Sequential access method (SAM) to read the allocated SYSOUT data set.
- Other standard MVS services, such as WAIT and POST logic.

Environment
Your application thread must include the following mapping macros:

- CVT
- IEFJESCT

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your program:

- IEFSSOBH
- IEFJSSIB
- IAZSSS2

Your application thread must meet the following requirements:

Minimum Authorization	Supervisor state
Dispatchable unit mode	Task
AMODE	31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control Parameters	The SSOB and SSS2 control blocks can reside in storage above 16 megabytes.
Recovery	The application thread should provide an ESTAE-type recovery environment for each task. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on an ESTAE-type recovery environment.

Figure 15 on page 99 shows the environment at the time of the call for SSI function code 79.

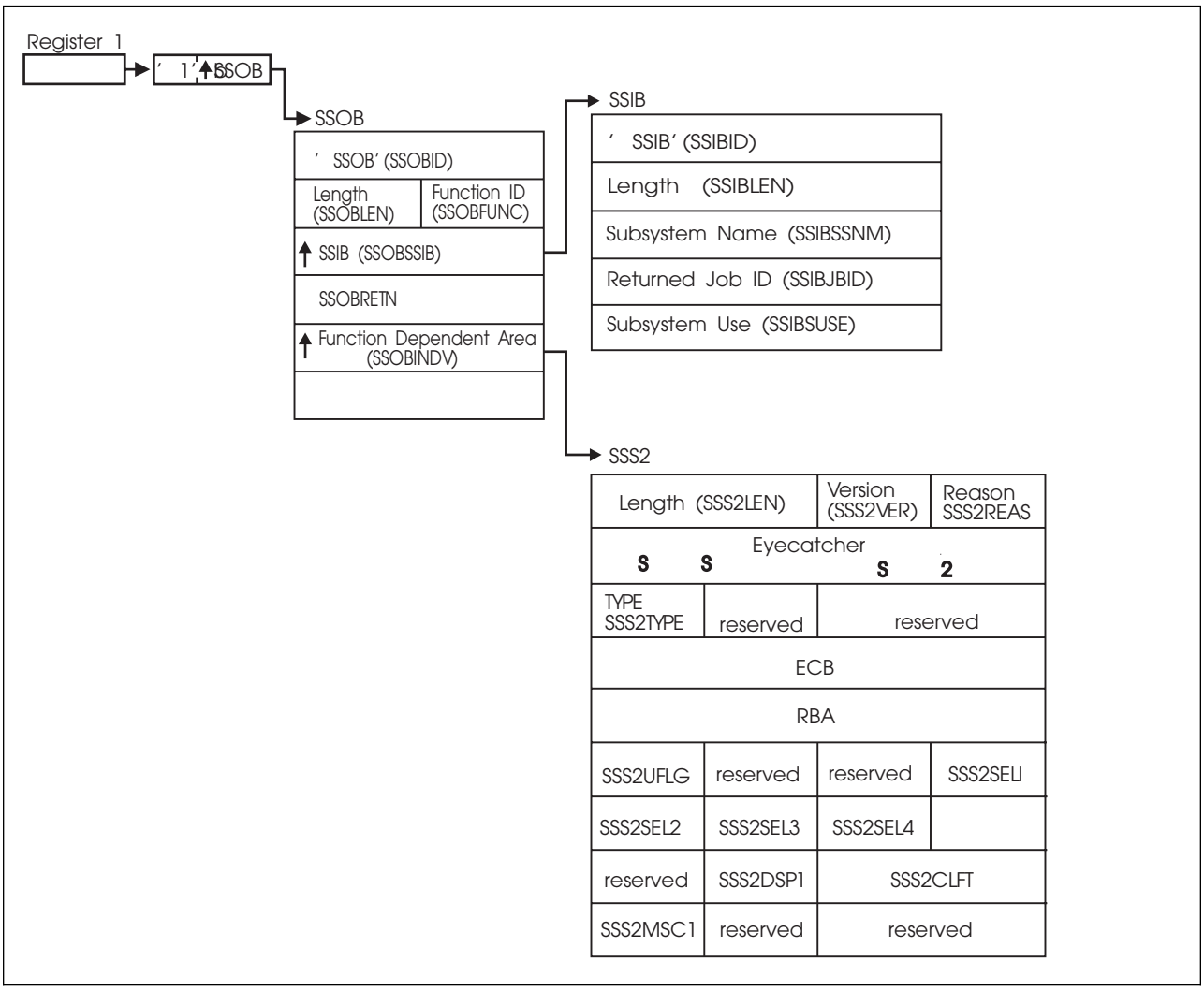

Figure 15. Environment at Time of Call for SSI Function Code 79

Input Register Information

Before issuing the IEFSSREQ macro, your application thread must ensure that the following general purpose registers contain:

Register	Contents
1	Address of a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.
13	Address of a standard 18-word save area.

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSS2

SSOB Contents: Your application thread sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier SSOB
SSOBLEN	Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function code 79 (SSOBSOU2)
SSOBSSIB	Address of an SSIB control block or zero. (If this field is zero, the life-of-job SSIB is used.) See "Subsystem Identification Block (SSIB)" on page 8 for more about the life-of-job SSIB.
SSOBRETN	Return code from JES
SSOBINDV	Address of the function dependent area (SSS2 control block)

Your application thread must set all other fields in the SSOB control block to binary zeros before issuing the IEFSSREQ macro.

SSIB Contents: If you don't use the life-of-job SSIB, your application thread must provide an SSIB and set the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier SSIB
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of the subsystem to which this SYSOUT Application Program Interface SSI call is directed. It is usually the primary JES, or in the case of JES2, possibly a secondary JES. If your routine has not been initiated from such a JES, your application thread must issue a Request Job ID call (SSI function code 20) prior to this SAPI call. You must use the same subsystem name in this SSIBSSNM field as you used for the Request Job ID call.
SSIBJBID	Job identifier — the job ID that was returned upon completion of the Request Job ID call (SSI function code 20).
SSIBSUSE	(JES3 only) Subsystem use — the SSIBSUSE value that was returned upon completion of the Request Job ID call (SSI function code 20).

Your application thread must set all other fields in the SSIB control block to binary zeros before issuing the IEFSSREQ macro.

SSS2 Contents: An application thread sets the following fields in the SSS2 control block on input:

Field Name	Description
SSS2LEN	Input field The length of the SSS2, set with the value SSS2SIZE
SSS2VER	Input field Set with the current value of SSS2CVER

SSS2EYE Input field

Eye catcher

Set with the character string SSS2

SSS2TYPE Input field

Type of call. Set with either SSS2PUGE, SSS2COUN, or SSS2BULK below.

SSS2PUGE Request type of PUT/GET.

Find a data set matching the selection criteria.

SSS2COUN Request type of Count.

Find data sets matching the selection criteria and count the number of data sets and the number of lines, pages, bytes, and records in those data sets.

SAF checks are not made for the data sets.

Counts are only a snapshot at the time the JES processes the request.

SSS2BULK Bulk modify request.

Find data sets matching the selection criteria and dispose of them as indicated in flag SSS2UFLG. No data sets are made available to the caller.

Input-Only Fields (Optional)

These fields, designated 'Optional Input-Only Fields', are used for the application to convey certain information about the particular call to the JES. Individual fields are set depending on the particular SAPI call being made at the time. Although these fields are designated 'optional', they must be set properly to effect the desired result of any particular SAPI request. For example, if the application thread needs to be posted when available work appears on the queue matching the selection criteria, then optional input field SSS2ECBP must have been set with the address of the caller-supplied ECB.

SSS2APPL For application use.

Either leave as binary zeros or supply an EBCDIC value that can be used for display purposes should you wish to view the SSS2 if performing diagnostics. An example might be to uniquely identify a particular thread's SSS2 in a storage dump.

SSS2APL1 For application use

SSS2ECBP Input field

Address of an ECB to be POSTed when work is available satisfying the selection criteria. The ECB is POSTed only if a prior PUT/GET request has returned with a reason code of SSS2EODS. The ECB is provided by the user.

The caller is allowed to free the memory for this ECB only after making a call with SSS2CTRL on in SSS2MSC1.

SSS2RBA Input/Output field

Relative byte address (RBA) of first record to be read.

Only valid if bit SSS2CHKP is on.

It is expected that SSS2RBA with the attendant SSS2CHKP bit be used by applications as a mechanism for interrupting the normal processing of a group of data sets. The most JES-efficient use of this approach is to process and delete data sets and to use the RBA mechanism only when the application wants to defer processing to a later time.

SSS2UFLG Input field

Specifies the modification processing to occur to the selected data sets.

SSS2UFLG is meaningful only if SSS2BULK is specified in SSS2TYPE (that is, this is a SAPI BULK MODIFY call).

SSS2SETC Use SSS2CLAS as the new class

SSS2DELC Delete selected data set(s)

SSS2ROUT Use SSS2DEST as the new data set destination

SSS2RLSE Release selected data sets

SSS2SEL1 Input field

Used for selection of new data sets.

You can specify selection from one, two, or three queues. The order of output with respect to the writer queues and with respect to held and non-held state is not predictable.

SSS2SHLD Select “HOLD/LEAVE” output (JES2); select “hold for TSO” output (JES3)

SSS2SXWH Select “hold for XWTR.” In a JES2 environment, this has the same meaning as SSS2SHLD.

SSS2SHOL Select from the hold queue. Specifying this setting guarantees that held output is returned regardless of the JES servicing this request.

SSS2SWTR Select “WRITE/KEEP” output (JES2); select from the writer queue if JES3. If none of the three bits are set, then the request is handled as if SSS2SWTR was specified.

SSS2SAWT Select from all the above

SSS2SCLS Use SSS2CLSL as the class selection list

SSS2SDST Use SSS2DEST as a filter

SSS2SJBN Use SSS2JOBN as a filter

SSS2SDUP Use SSS2JOBN as a filter, but give a reason code of SSS2DUPJ if duplicate jobs. This setting is meaningful only if SSS2JOBN has no wildcard characters. The setting is not used for

| | | a bulk modify (SSS2BULK) or count (SSS2COUN) request.
| | | |
| | **SSS2SDU2** | Give a reason code of SSS2DUPJ if duplicate job. This setting is only meaningful if SSS2JOBN is also set.
| | **SSS2SJBI** | Use SSS2JBIL and SSS2JBIH as filters
| **SSS2SEL2** | Input field | |
| | Used for selection of new data sets. | |
| | **SSS2SPGM** | Use SSS2PGMN as a filter
| | **SSS2SFRM** | Use SSS2FORM as a filter
| | **SSS2SCRE** | Use SSS2CREA as a filter
| | **SSS2SPRM** | Use SSS2PRMO as a filter
| | **SSS2SIPA** | Only select output that has an Internet Protocol (IP) address
| | **SSS2SIPN** | Only select output that has no IP address. This setting is mutually exclusive with SSS2SIPA
| | **SSS2SFCB** | Use SSS2FCB as a filter
| | **SSS2SUCS** | Use SSS2UCS as a filter
| **SSS2SEL3** | Input field | |
| | Used for selection of new data sets. | |
| | **SSS2SSTC** | Select Started Tasks (STCs) (see note in SSS2STYP)
| | **SSS2STSU** | Select Time Sharing Users (TSUs) (see note in SSS2STYP)
| | **SSS2SJOB** | Select batch jobs (JOBs) (see note in SSS2STYP)
| | **SSS2SAPC** | Select APPC output (see note in SSS2STYP)
| | **SSS2STYP** | If none of these bits is on, then selection is as if **all** of the bits are on.
| **SSS2SEL4** | Input field | |
| | Used for selection of new data sets. | |
| | **SSS2SMOD** | Use SSS2MOD as a filter
| | **SSS2SFLS** | Use SSS2FLSH as a filter
| | **SSS2SAGE** | Data sets selected must be at least as old as the value in SSS2AGE.
| | **SSS2SLIN** | Use minimum and maximum line counts specified in SSS2LMIN and SSS2LMAX as a data set group filter
| | **SSS2SPAG** | Use minimum and maximum page counts specified in SSS2PMIN and SSS2PMAX as a data set group filter

SSS2SPRI Select output based on priority

SSS2SVOL Select output based on the volume serial list in SSS2VOL (SSS2NVOL in SSS2RET2 on if the JES does not support)

SSS2SCHR Use Printer translation tables in SSS2CHAR as a filter (SSS2NCHR in SSS2RET2 on if the JES does not support)

SSS2SEL5 Input field

Used for selection of new data sets.

SSS2SCPN Select data set having no CPDS.

SSS2SCTK Select by client token. Mutually exclusive with SSS2SJBI. You can use this filter as the only input or in conjunction with additional filters. If you use other filters, they must all match the SYSOUT attributes.

SSS2SBRO Use SAPI as a "browse" facility rather than a "processing" facility.

SSS2SODS Use SSS2ODST as a filter.

SSS2MSC1 Input filed

Used for selection of new data sets.

SSS2CTRL

On Processing complete. JES disassociates all its held resources on behalf of the calling thread.

Off Normal processing is to occur depending on SSS2TYPE's value (that is, a SAPI PUT/GET, SAPI COUNT, or SAPI BULK MODIFY call).

SSS2FSWB Return token for SJFREQ calls in field SSS2SWBT. This also means that the address of the SWBTUREQ buffer is returned in field SSS2SWTU

SSS2FSWT Return address of SWBTUREQ buffer in field SSS2SWTU

SSS2NJEH Return address of NJE data set and job headers if available (SSS2NJED for data set header; SSS2NJEJ for job header) (SSS2NNHD in SSS2RET2 on if the JES does not support)

SSS2JOBN Input field

Used for selection of new data sets. Supports wildcards.

Jobname used for selection (if SSS2SJBN on)

To influence the type of job selected, use the settings in SSS2SEL3.

SSS2JBIL Input field

Used for selection of new data sets.

Low jobid used for selection (if SSS2SJBI on).

Jobid's are of the form xxxnnnnn where xxx is JOB, JO, or J, and nnnnn is one to seven digits. Embedded and trailing blanks are acceptable. The maximum length of the jobid is eight characters.

To influence the type of job selected, use the settings in SSS2SEL3.

SSS2JBIH Input field

Used for selection of new data sets.

High jobid used for selection (if SSS2SJBI on). This value must be null or at least as high as SSS2JBIL.

Jobid's are of the form xxxnnnnn where xxx is JOB, JO, or J, and nnnnn is one to seven digits. Embedded and trailing blanks are acceptable. The maximum length of the jobid is eight characters.

SSS2CREA Input field

Used for selection of new data sets. Supports wildcards.

Creator userid used for selection (if SSS2SCRE on).

SSS2PRMO Input field

One to four values used for selection of new data sets. Supports wildcards.

One to four PRMODEs used for selection (if SSS2SPRM is on).

This list must contain null entries (X'40's) for any of the elements not containing a selection parameter.

SSS2DEST Input field

Used for selection of new data sets. Supports wildcards.

Destination value used for selection (if SSS2SDST on).

SSS2DES2 Input field

Specifies the new destination of data sets selected for bulk modify requests.

New destination if SSS2ROUT is on.

SSS2PGMN Input field

Used for selection of new data sets. Supports wildcards.

User writer name used for selection (if SSS2SPGM is on).

SSS2FORM Input field

One to eight values used for selection of new data sets. Supports wildcards.

One to eight form numbers used for selection (if SSS2SFRM is on).

This list must contain null entries (X'40's) for any of the elements not containing a selection parameter.

SSS2CLSL Input field

Used for selection of new data sets.

SYSOUT class list used for selection (if SSS2SCLS is on). List is terminated by X'40'.

If multiple classes are listed for the initial PUT/GET request, JES searches all data sets for the first class specified before searching for the second class specified, and so on until all classes have been searched.

For JES3 only, due to searching algorithms, it is suggested that an application thread, if using multiple SYSOUT classes in SSS2CLSL, set the value to the single, returned class (from SSS2CLAR) after a data set is returned from a SAPI PUT/GET call if the application thread wishes additional data sets of this class to be returned. This prevents JES3 from excessive queue searches. After the SSOBRETN value of SSS2EODS has been returned, the application can then re-supply SSS2CLSL with the original, multi-class list to continue to search for additional data sets on subsequent SAPI PUT/GET calls.

SSS2CLAS Input field

Specifies the new class for data sets modified through bulk modify.

New class if SSS2SETC is on.

SSS2LMIN Input field

Used for selection of new data sets.

Minimum line count for data set group (if SSS2SLIN is on)

SSS2LMAX Input field

Used for selection of new data sets.

Maximum line count for data set group (if SSS2SLIN is on)

SSS2PMIN Input field

Used for selection of new data sets.

Minimum page count for data set group (if SSS2SPAG is on)

SSS2PMAX Input field

Used for selection of new data sets.

Maximum page count for data set group (if SSS2SPAG is on)

SSS2FCB Input field

Used for selection of new data sets.

FCB image name used for selection (if SSS2SFCB is on)

SSS2UCS Input field

Used for selection of new data sets.

UCS image name used for selection (if SSS2SUCS is on)

SSS2CHAR Input field

One to four values used to select new data sets (JES3 only).

One to four printer translate tables used for selection (if SSS2SCHR is on).

This list must contain null entries (X'40's) for any of the elements not containing a selection parameter.

SSS2MOD Input field

Used for selection of new data sets.

Modify image used for selection (if SSS2SMOD is on)

SSS2FLSH Input field

Used for selection of new data sets.

Flash cartridge ID for selection (if SSS2SFLH is on)

SSS2SECT Input field

Used for selection of new data sets.

Address of the security token or zero. If the application thread provides the address of the token to be returned, then the application thread must set the length in the first byte of the area, and the version in the second byte of the area prior to issuing the SAPI PUT/GET call.

SSS2AGE Input field

Used for selection of new data sets.

Minimum age of data sets to be selected (if SSS2SAGE is on). The low order bit represents 1.048576 seconds (that is, the high order word of the TOD clock).

SSS2VOL Input field

One to four values used to select new data sets (JES2 only).

One to four SPOOL volume serial numbers. Jobs are selected if and only if the job has output on at least one of the volumes listed and SSS2SVOL is on.

This list must contain null entries (X'40's) for any of the elements not containing a selection parameter.

SSS2CTKN Input field

| Address of client token used for selection (if SSS2SCTK is on).

| Mutually exclusive with SSS2SJBI.

SSS2ODST Input field

Specifies the eight-character origination node name from which the job was submitted.

Valid only if SSS2SODS in SSS2SEL5 is on.

Input Disposition Fields (Optional): These input disposition fields (optionally specified by the application thread) are used to determine what is to be done with the data set that was last returned to the application and that is now being

disposed of. If this is the first SAPI PUT/GET call, then there is no “last” data set; therefore the following information is ignored.

SSS2DSP1 Input field

Flags describing the disposition for the data set whose name is currently in SSS2DSN.

Settings in SSS2DSP1 and other dispositions are honored if and only if the keep bit (SSS2DKPE) is **on**. The absence of the keep bit implies that the data set will be deleted.

If SSS2DKPE is **off**, the data set is deleted regardless of other disposition settings in this section.

SSS2DKPE Keep the data set

SSS2RHLD Keep the data set and make it non-selectable (system hold)

SSS2RNPR Keep the data set and leave it selectable, but never return to this SAPI address space

SSS2RNPR on means that the JES does not return the data set to the application address space again. The application must treat this as a “suggestion” to the JES. The data set could be seen again by the application if:

- The JES is restarted
- The application is restarted
- Some characteristic is changed by the operator or another application

SSS2DHLD Hold the data set

This bit is mutually exclusive with SSS2DRLS.

SSS2DRLS Release the data set

This bit is mutually exclusive with SSS2DHLD.

SSS2CHKP Use SSS2RBA to checkpoint the data set position. Next data set returned will have SSS2DSF on.

SSS2DNWR Set writer name to a null value (all X'40's).

SSS2RNPT Leave the data set selectable, but never return to this sysout API thread again.

The following fields (SSS2DCLS, SSS2DFOR, SSS2DPGM, SSS2DDES, and SSS2CLFT) are used to change a subset of the data set characteristics. These only have meaning if the data set is kept (SSS2DKPE on in SSS2DSP1).

A null value indicates that no override is desired for SSS2DCLS, SSS2DFOR, SSS2DPGM, SSS2DDES, and SSS2CLFT.

SSS2DCLS Input field

New class.

SSS2DFOR Input field

New forms.

SSS2DPGM Input field

New user writer name.

SSS2DDES Input field

New destination.

SSS2CLFT Input field

Number of copies left to process. Values > 255 are treated as 255.

Output Register Information

When control returns to your application, the general purpose registers contain:

Register	Contents
0	Used as a work register by the system
1	Address of the SSOB control block
2 — 13	Same as on entry to call
14	Return address
15	Return code

Return Code Information

The SSI places one of the following decimal return codes in register 15. Examine the return code to determine if the request was processed.

Return Code (Decimal)	Meaning
SSRTOK (0)	The SAPI call completed. Check the SSOBRETN field for specific function information.
SSRTNSUP (4)	The subsystem specified in the SSIBSSNM field does not support this function.
SSRTNTUP (8)	The subsystem specified in the SSIBSSNM field exists, but it is not active.
SSRTNOSS (12)	The subsystem specified in the SSIBSSNM field is not defined to MVS.
SSRTDIST (16)	The pointer to the SSOB control block or the SSIB control block is not valid, or the function code specified in the SSOBFUNC field is greater than the maximum number of functions supported by the subsystem specified in the SSIBSSNM field.
SSRTLERR (20)	Either the SSIB control block or the SSOB control block has incorrect lengths or formats.
SSRTNSSI(24)	The SSI has not been initialized.

Output Parameters

Output parameters for the function routine are SSOBRETN and SSS2.

SSOBRETN Contents: When control returns to your application thread and register 15 contains a zero, the SSOBRETN field contains one of the following decimal values:

Value (Decimal)	Meaning
SSS2RTOK (0)	Successful completion; for SAPI PUT/GET calls, a data set was returned whose name is in SSS2DSN
SSS2EODS (4)	No more data sets to select
	See the reason codes defined for SSS2REAS at "Reason Codes for SSOBRETN being SSS2EODS" on page 112.
SSS2INVA (8)	Invalid search arguments
SSS2UNAV (12)	Unable to process now
SSS2DUPJ (16)	Duplicate jobnames. (This reason code can occur only if SSS2SDUP is on.) The duplicate job may or may not have characteristics matching the SSS2 filter set.
SSS2IDST (20)	Invalid destination specified
SSS2TKNM (28)	Token map failed. Application will not be allowed to allocate to data set and DISP=(,KEEP) will be forced
SSS2LERR (32)	Logic error See the reason codes defined for SSS2REAS at "Reason Codes for SSOBRETN being SSS2LERR."
SSS2ICLS (36)	SSS2CLAS not A-Z and not 0-9
SSS2BDIS (40)	Disposition settings incorrect See the reason codes defined for SSS2REAS at "Reason Codes for SSOBRETN being SSS2BDIS" on page 111.
SSS2CLON (44)	Disposition for data set group not uniform (See SSS2DSH). DISP=(,KEEP) is forced with no override disposition information honored

SSS2 Contents: The SSS2 control block contains the following information about the data set returned from your application's request:

Reason Codes for SSOBRETN being SSS2LERR: If field SSOBRETN contains SSS2LERR, then field SSS2REAS will contain one of the following reason codes:

Value (Decimal)	Meaning
SSS2RENI (4)	SSS2JEST zero, but SSS2DSN not null
SSS2REIP (8)	SSS2SIPA and SSS2SIPN are mutually exclusive
SSS2RALO (12)	Prior data set still allocated
SSS2RDUP (16)	SSS2SDUP on in SSS2SEL1 and wildcards used in SSS2JOBN

SSS2RJBI (20) SSS2JBIH < SSS2JBIL and SSS2SJBI on

SSS2RCRE (24) SSS2CREA has error and SSS2SCRE on

SSS2RLEN (28) SSS2LEN is less than SSS2SIZE

SSS2RTYP (32) SSS2TYPE is not valid

SSS2RDES (36) SSS2DEST has error and SSS2SDST on

SSS2RJNM (40) SSS2JOBN has error and SSS2SJBN on

SSS2RFRM (44) SSS2FORM has error and SSS2SFRM on

SSS2RPGM (48) SSS2PGMN has error and SSS2SPGM on

SSS2RPRM (52) SSS2PRMO has error and SSS2SPRM on

SSS2RCLS (56) SSS2CLSL has error and SSS2SCLS on

SSS2RFCB (60) SSS2FCB has error and SSS2SFCB on

SSS2RUCS (64) SSS2UCS has error and SSS2SUCS on

SSS2RCHR (68) SSS2CHAR has error and SSS2SCHR on

SSS2RMO (72) SSS2MOD has error and SSS2SMOD on

SSS2RFL (76) SSS2FLSH has error and SSS2SFLS on

SSS2RLPM (80) SSS2LMIN or SSS2LMAX is negative and SSS2SLIN is on -- or -- SSS2PMIN or SSS2PMAX is negative and SSS2SPAG is on

SSS2RLPG (84) SSS2LMIN > SSS2LMAX and SSS2SLIN on -- or -- SSS2PMIN > SSS2PMAX and SSS2SPAG on

SSS2RDE2 (88) SSS2DES2 has error and SSS2TYPE is SSS2BULK and SSS2ROUT on

SSS2RVOL (92) SSS2VOL has error and SSS2SVOL on

SSS2REYE (96) SSS2EYE does not have SSS2

SSS2RCTK (100) SSS2SCTK is on but SSS2CTKN is not specified or not valid.

SSS2RBRO (104) SSS2SBRO is on but Bulk Modify or Count was requested.

SSS2RECJ (108) SSS2SCTK and SSS2SJBI are mutually exclusive.

SSS2RODS (112) SSS2ODST has error and SS2SODS on

The remainder of the reason codes up through 180 are reserved for SSS2LERR.

Reason Codes for SSOBRETN being SSS2BDIS: If field SSOBRETN contains SSS2BDIS, then field SSS2REAS will contain one of the following reason codes:

Value (Decimal)	Meaning
SSS2RDCL (184)	SSS2DCLS has error
SSS2RDFR (188)	SSS2DFOR has error
SSS2RDPG (192)	SSS2DPGM has error
SSS2RDDS (196)	SSS2DDES has error
SSS2RDHR (200)	Both SSS2DHLD and SSS2DRLS specified

Reason codes 204 through 236 are reserved for SSS2BDIS.

Reason Codes for SSOBRETN being SSS2EODS: The following SSS2EODS reason codes are applicable only when SSS2CTKN is used as a filter:

Value (Decimal)	Meaning
SSS2RENM (240)	No matching output
SSS2RENS (244)	Matching output not selectable

Reason codes 248 through 252 are reserved for SSS2EODS.

Output-Only Fields

These fields are returned to the application thread with information managed by the JES. Once the initial SSS2 control block has been set to X'00's (or after a previous IEFSSREQ request with SSS2CTRL having been set), the application thread must not modify the contents of any of these 'Output-Only' fields.

SSS2REAS Output field

Reason code associated with SSOBRETN value of SSS2LERR or SSS2BDIS. See the explanation at "Reason Codes for SSOBRETN being SSS2LERR" on page 110 and "Reason Codes for SSOBRETN being SSS2BDIS" on page 111.

SSS2JEST Output field

JES token associated with this SAPI request. A zero value here implies that this is a new request. A new request implies that the SSS2DSN is null.

The application, once originally initializing this field to X'00's, must not modify or subsequently reference this field.

SSS2BTOK Output field

Address of a JES initialized data area (a dynamic allocation text unit). This value must be copied to a dynamic allocation text unit pointer by the application thread prior to the dynamic allocation of the returned data set.

See topic 90 for information concerning programming considerations related to the use of SSS2BTOK.

SSS2COPY Output field

Total number of copies requested by creator. A data set is returned through this interface only once no matter how many copies were requested by the creator.

SSS2CPYG Output field

Copy groups

SSS2JOBR Output field

Jobname of selected job

SSS2JBIR Output field

Job ID of selected job

SSS2OJBI Output field

Original jobid of selected job. (Original id may be different from current jobid.) (JES3 always returns blanks.)

SSS2CRER Output field

Creator userid of data set selected

SSS2JDVT Output field

JCL definition vector table

SSS2PRMR Output field

PRMODE of data set selected

SSS2DESR Output field

Destination of selected data set

SSS2PGMR Output field

Writer name of selected data set

SSS2FORR Output field

Form number of selected data set

SSS2TJN Output field

APPC Transaction Program Jobname that created this data set

SSS2TJID Output field

APPC Transaction Program Job ID that created this data set

SSS2DSN Output field

Data set name of selected data set. Must be blanks or zeros if SSS2JEST is zero.

You must not make assumptions regarding the format of the data set name.

SSS2SEGM Output field

Segment id (zero if data set not segmented)

SSS2WRTN Output field

SWB processing error - return code given:

SSS2WOK (0) Processing successful

SSS2WERR (4) Processing failed

Note that reason code field SSS2WRSN is also set.

SSS2WRSN Output field

SWB processing error - reason code set to non-zero only if SSS2WRTN is non-zero

SSS2WRSN has the following value: `SSSSCCRR` where `SSSSCCRR` is defined as:

SSSS Reason code from SJF service RR or a qualifier for a JES service error

CC Return code from SJF service RR

CC is '00' if RR is 4 or 8

RR indicates the SJF service or JES service

4 = JES SPOOL I/O Error
8 = JES Memory management error
12 = SJFREQ REQUEST=SWBTU_MERGE
16 = SJFREQ REQUEST=PUTSWB
20 = SJFREQ REQUEST=JDTEXTRACT
24 = SWBTUREQ REQUEST=RETRIEVE

SSS2CLAR Output field

SYSOUT class of selected data set

SSS2MLRL Output field

Maximum logical record length (LRECL)

SSS2DSID Output field

DSID for the selected data set

SSS2RET1 Output field

SSS2GNVA JES returned an output group name in SSS2OGNM (JES2 only).
SSS2DSCL Line count, page count, byte count, and record count (SSS2LNCT, SSS2PGCT, SSS2BYCT, and SSS2RCCT) are accurate. This bit will not be on if there was an abnormal termination or the data was created on a different node.
SSS2DSF First data set in output group
SSS2DSC Output group being continued
SSS2DSL Last data set in output group
SSS2IP An Internet Protocol (IP) destination is available in the SJF data. See SSS2SWBT and SSS2SWTU.
SSS2BRST BURST=YES specified
SSS2OPTJ OPTCD=J specified

SSS2RET2 Output field

SSS2NCHR Selection using printer translation tables not supported
SSS2NVOL Selecting output based on a volume serial list not supported
SSS2NNHD Returning addresses of NJE headers not supported
SSS2NMOD Selecting output based on a modification is not supported
SSS2NPRI Selecting output in priority order is not supported
SSS2NIPA IP address selection not supported. Turned on if JES does not support and SSS2SIPA or SSS2SIPN is on

SSS2RET3 Output field

SSS2RSTC Data set created by started task
SSS2RTSU Data set created by time sharing user
SSS2RJOB Data set created by batch job

SSS2RET4 Output field

SSS2CPDS Data set has page mode data

SSS2SPUN Data set was spun at close
SSS2SDSH All data sets in group must be unallocated identically.

SSS2RET5 Output field Queue where the data set resides

SSS2RHLV Data set on "HOLD/LEAVE" queue (JES2) or "Hold for TSO" queue (JES3)
SSS2RXWH Data set on "Hold for XWTR" queue. This will never be true in a JES2 environment.
SSS2RHOL Data set on one of the held queues.
SSS2RWTR Data set on "Write/Keep" queue (JES2) or "Writer" queue (JES3).

The following count fields (SSS2LNCT, SSS2PGCT, SSS2BYCT, and SSS2RCCT) are valid only if SSS2DSCL is on in SSS2RET1.

The fields represent counts for the single data set returned if SSS2TYPE is SSS2PUGE. The fields represent the total for all data sets selected if SSS2TYPE is SSS2COUN.

For a SAPI PUT/GET request, these field values are for the single returned data set. For a SAPI COUNT request, these values represent the sum of all the data sets that have been selected for the count, not taking individual copies requested of these data sets into effect.

SSS2LNCT Output field

Line count

For a PUT/GET request, this value is for the single returned data set. For a COUNT request, this value represents the sum of all the data sets that have been selected for the count, not taking individual copies requested of these data sets into effect.

SSS2PGCT Output field

Page count

For a PUT/GET request, this value is for the single returned data set. For a COUNT request, this value represents the sum of all the data sets that have been selected for the count, not taking individual copies requested of these data sets into effect.

SSS2BYCT Output field

Byte count after blank truncation

For a PUT/GET request, this value is for the single returned data set. For a COUNT request, this value represents the sum of all the data sets that have been selected for the count, not taking individual copies requested of these data sets into effect.

SSS2RCCT Output field

Spool record count (JES3 only)

For a PUT/GET request, this value is for the single returned data set. For a COUNT request, this value represents the sum of all the data sets that have been selected in the count, not taking individual copies of these data sets into effect.

SSS2PRCD Output field

Procname for the step creating this data set

SSS2STPD Output field

Stepname for the step creating this data set

SSS2DDND Output field

DDNAME for the data set creation

SSS2SWBT Output field

Token used for SJFREQ services. This field is filled in if flag SSS2FSWB is set.

See topic 90 for programming considerations concerning the use of the SSS2SWBT field.

SSS2SWTU Output field

Address of the SWBTU block. This field is filled in if flag SSS2FSWT or SSS2FSWB is set.

See topic 90 for programming considerations concerning the use of the SSS2SWTU field.

SSS2PRIV Input/Output field

Copied to and from SAPPRIV if JES2, COWPRIV if JES3.

SSS2CHR1 Output field

Printer translate table 1

SSS2CHR2 Output field

Printer translate table 2

SSS2CHR3 Output field

Printer translate table 3

SSS2CHR4 Output field

Printer translate table 4

SSS2OGNM Output field

JES2 output group name

The data set returned with a given output group name will not necessarily continue to have the given output group name if this request keeps the data set.

This field is valid only if SSS2GNVA is on in SSS2RET1.

SSS2RMOD Output field

Printer copy modify image

SSS2MODT Output field

Printer table reference character

SSS2RFLS Output field

Printer flash cartridge ID

SSS2FLSC Output field

Number of flash copies

SSS2PRIO Output field

Data set priority

SSS2LINC Output field

Lines/page (JES2 only)

SSS2TOD Output field

Date and time of data set availability in TOD format (that is, this value is the high order word of the TOD clock obtained through a STCK machine instruction.)

SSS2CDS Output field

Count of work units (JOEs/OSEs) that match the selection criteria.

SSS2NJED Output field

Address of NJE data set header. This field is non-zero if a data set header is available and the SSS2NJEH flag is on.

SSS2FCBR Output field

Forms control buffer (FCB). Set to asterisks ('****') if the default FCB is returned.

SSS2UCSR Output field

Universal character set (UCS). Set to asterisks ('****') if the default UCS is returned.

SSS2DSTR Output field

Address of a token that can be used in a subsequent PUTGET call (SSS2PUGE in SSS2TYPE) to get the same data set. To refetch this data set, place SSS2DSTR in field SSS2CTKN and turn on bit SSS2SCTK in flag SSS2SEL5.

Job-Level Output-Only Fields

Similar to the prior 'Output-Only' section, but these fields are applicable to **all** data sets from a single job. The information contained within is on a job-level basis, not on an individual data set-level basis.

SSS2PNAM Output field

Programmer name from the JOB statement

SSS2ROOM Output field

Job level room number

SSS2NOTN Output field

Job notify node

SSS2NOTU Output field

Job notify userid

SSS2ACCT Output field

Address of encoded accounting information

Accounting information is provided in 'SMF' format, just as if it is in type 5 and type 30 SMF records.

AL1(number-of-pairs-that-follow) followed by 0 or more pairs of the form:
AL1(length),CLlength'string' A length of 0 indicates an omitted field

Example: Accounting information of (X3600,42,,ANDY):

```
DC  AL1(4)              Nr of fields
DC  AL1(5),CL5'X3600'   field 1
DC  AL1(2),CL2'42'      field 2
DC  AL1(0)              field 3 (null)
DC  AL1(4),CL4'ANDY'    field 4
```

SSS2XEQ Output field

Node where job executed

SSS2ORG Output field

Node where job entered system

SSS2TIME Output field

Time on input processor for the selected job. This is in hundredths of seconds since midnight.

The time field is local, not UCT/GMT.

SSS2DATE Output field

Date on input processor for the selected job. This is in the form 0cyydddF.

The date field is local, not UCT/GMT.

SSS2SYS Output field

System name of the MVS image where the job output was created.

This field is not available if the SYSOUT came from a network node or the job was reloaded.

SSS2MBR Output field

Member name of the JES2 image where the job output was created.

This field is not available if the SYSOUT came from a network node or the job was reloaded.

SSS2NJEJ Output field

Address of NJE job header. This field is non-zero if the job header is available and SSS2NJEH flag is on.

SSS2NACT Output field

Network account number.

In JES2, this information is retrieved from the /*NETACCT JECL statement.

In JES3, this information is retrieved from the //*NETACCT JECL statement.

Extended Status Function Call — SSI Function Code 80

The extended status function call (SSI function code 80) allows a user-supplied program to obtain detailed status information about jobs and SYSOUT in the JES queue. Both JES2 and JES3 subsystems support job status information. Only JES2 subsystems support SYSOUT status information.

Extended Status Request Types

The extended status interface is designed to be a general purpose interface to obtain information from JES. Callers use the STATTYPE field to indicate the type of data they require. This SSI call can only return job information and SYSOUT status information that is readily available (that is, no I/O required).

In addition to the type of data being requested, there is a memory management call type (STATTYPE set to STATMEM). The extended status function SSI manages the storage needed to return data to the caller. Once the caller completes processing the returned data, a memory management call is required to free the data areas.

Type of Request

Directed or broadcast SSI call.

Use information

To use the extended status SSI, a caller must first choose the type of data to request. Job level data with or without SYSOUT level data can be requested. (JES3 subsystems currently do not support returning SYSOUT data.) If only job level data is requested, one output element is created for each job. When SYSOUT data is requested, one output element is created for each job with output and one element for each piece of SYSOUT. For example, a job with four pieces of SYSOUT that matched all selection criteria would return one job level data element and four SYSOUT level data elements.

Next, the caller must decide what filters to use to select which data elements are returned. A filter is an attribute that a job or SYSOUT must possess to be returned by the interface. Filters are either at the job or SYSOUT level. JES3 subsystems currently do not support SYSOUT level filters.) Use of filters is not dependent on the type of data being requested. If only job level data is being requested and a SYSOUT filter is specified, then only jobs that have SYSOUT which passes the SYSOUT filter will be returned. Only one job level data element per matching job is returned.

A typical filter has some value associated with it, such as JOBNAME with value of TOMW. However, some filters do not have values associated with them, such as jobs that are held. If no filters are applied, the the extended status function call returns information on all jobs or all SYSOUT. Because the number of jobs and SYSOUT in the system can be great, it is recommended that if information on all jobs or SYSOUT is not required, a filter be specified to limit the returned data.

All returned data will match all filters requested. If you need to limit (filter) the data based on two different values (such as a JOBNAME of PAULAK or ZOOT), you can make multiple calls to the extended status SSI before processing the results. None of the output areas set by the subsystem will be cleared until the memory management call is made. This allows a second SSI call to append its results to

the results of the first call. For example, if all jobs that are owned by userid PAULAK or ZOOT are needed, use the following series of calls:

1. Request job data with an owner filter of PAULAK
2. Request job data with an owner filter of ZOOT
3. Process all data elements returned
4. Issue memory management request to return data areas.

This is preferable to requesting information on all jobs and then selecting for processing only those data elements for jobs owned by PAULAK or ZOOT.

When information is obtained through multiple calls, it is the caller's responsibility to eliminate duplicate data. The extended status SSI makes no checks on subsequent calls to ensure information for the same job is not returned multiple times. In a JES2 environment, if the SSI is broadcast to all subsystems, JES2 suppresses replies from secondary JES2s in the same MAS as a subsystem that has already replied.

For JES2 subsystems, information returned through this SSI is obtained from a local copy of JES2's work queues. As such, it might not reflect the current state of a job or SYSOUT element. The information can be as much as a few seconds old. If your application must have the most current job status, then use some other interface (such as operator commands) to obtain the information.

For JES3 subsystems, information returned through this SSI is obtained from work queues on the JES3 global. As such, the information reflects the current status of the job or SYSOUT at the time of the request.

The order of information returned is dependent on the filters requested and the subsystem responding. The only ordering that can be assumed is that as subsystems add data to the output area, that information is added to the beginning of the output area. For example, in a series of two calls, the results from the second call will appear on the chain of output areas before the results of the first call. Similarly, if the call is broadcast to all subsystems, the output of the primary subsystem appears after the output of any secondary subsystems.

Issued to

- A JES2 subsystem (either primary or secondary) or a JES3 subsystem for a directed request.
- The master subsystem for a broadcast request.

Related SSI Codes

None.

Related Concepts

None.

Environment

The caller (issuer of the IEFSSREQ macro) must include the following mapping macros:

- CVT
- IEFJESCT

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your program:

- IEFSSOBH
- IEFJSSIB
- IAZSSST

The caller must meet the following requirements:

Minimum Authorization	Supervisor State, any PSW key
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control Parameters	The SSOB, SSIB, and IAZSSST control blocks can reside above or below 16 megabytes in virtual storage.
Recovery	The caller should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Assembler Services Guide* for more information on an ESTAE-type recovery environment.

Figure 16 shows the environment at the time of the call for SSI function code 80.

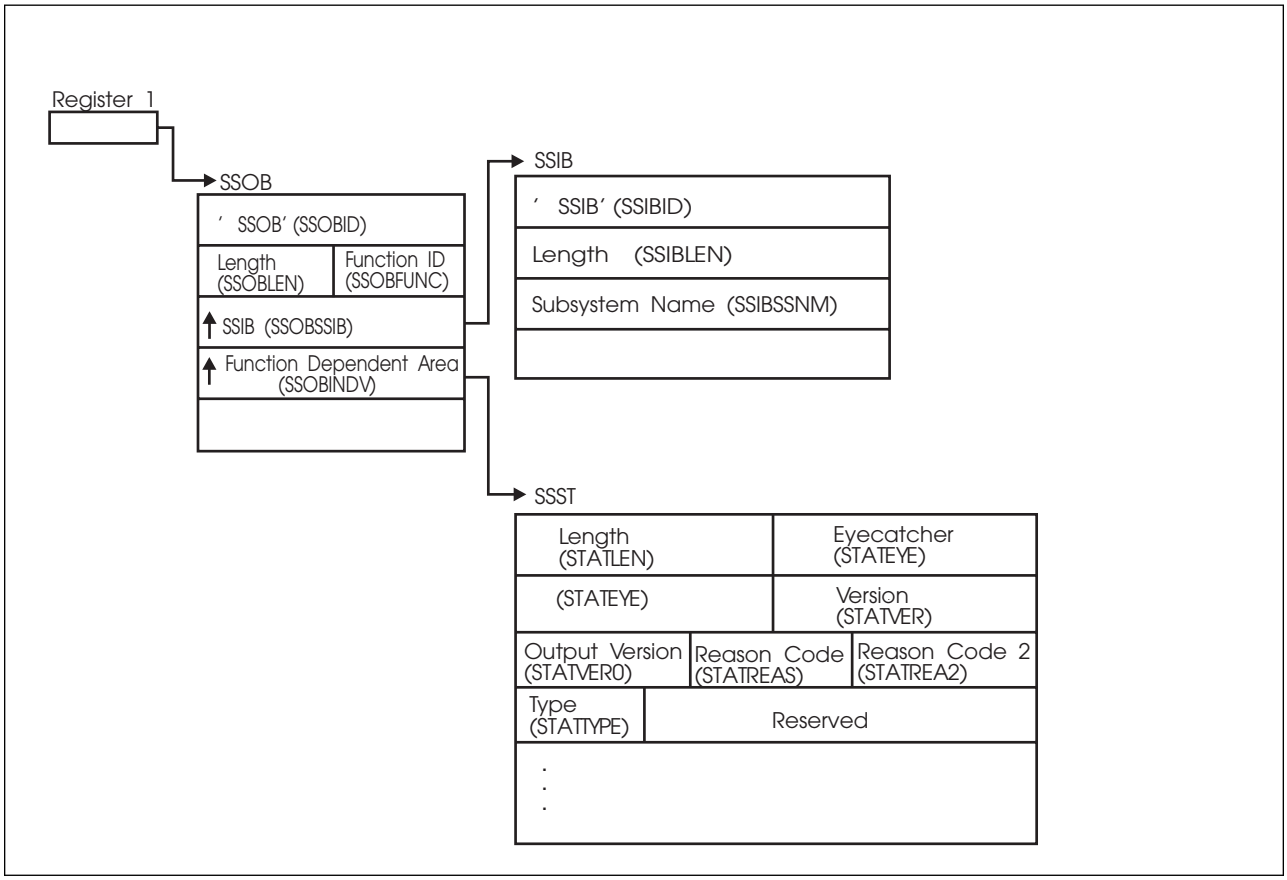

Figure 16. Environment at Time of Call for SSI Function Code 80

Input Register Information

Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

Register	Contents
1	Address of a 1-word parameter list that has the high-order bit on and a pointer to the SSOB control block in the low-order 31 bits.
13	Address of a standard 18-word save area

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- IAZSSST

SSOB Contents: The caller sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function code 80 (SSOBESTA)
SSOBSSIB	Address of an SSIB control block or zero (if this field is zero, the life-of-job SSIB is used). See "Subsystem Identification Block (SSIB)" on page 8 for more information on the life-of-job SSIB.
SSOBINDV	Address of the function-dependent area (IAZSSST control block).

Set all other fields in the SSOB control block to binary zeros before issuing the IEFSSREQ macro.

SSIB Contents: If you do not use the life-of-job SSIB, the caller must provide an SSIB and set the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of the subsystem to which this extended status function call is directed (or MSTR if it is to be broadcast).

Set all other fields in the SSIB control block to binary zeros before issuing the IEFSSREQ macro.

IAZSSST Contents: The caller must set the following fields in the IAZSSST control block on input:

Field Name	Description
STATLEN	Length of the IAZSSST (STATSIZE) control block. For STATVER, set to STATV010 or STATV020, a length of at least STATSIZ1 or STATSIZ2 is required. For STATVER set to STATV030 or greater, a length of at least STATSIZ3 is required. STATSIZE is always equated to the largest length of the IAZSSST control block and in general should be used to obtain storage for the IAZSSST and to set STATLEN.
STATEYE	Eyecatcher for the control block (set to C'STAT')
STATVER	Input version of the IAZSSST control block (Set to STATV010 for the initial version of the control block, STATV020 for OS/390 Version 2 Release 4, STATV030 for OS/390 Version 2 Release 5.)
STATTYPE	Function to be performed on this request. Valid functions are:

Field Value	Description
STATTERS	Requests obtaining basic data for jobs. The data returned on this call does not require large amounts of system overhead.
STATOUTT	Requests obtaining basic data for SYSOUT (including job level information). Data returned on this call does not require large amounts of system overhead. This request type is only supported by JES2 subsystems. STATVER must be set to at least STATV030 for this request to be valid.
STATMEM	Return memory from a previous request. After one or more requests for data, the memory obtained must be returned using this function.

The caller can also set the following fields in the IAZSSST control block on input to limit (or select) the jobs for which data will be returned:

Field Name	Description
STATSEL1	Flag byte which describes the filters to use to select jobs. Each bit corresponds to a filter field which must match any job returned.

Bit Name	Description
STATSCLS	Apply job class filter in STATCLSL.
STATSDST	Apply default destination filter in STATDEST.
STATSJBN	Apply job name filter in STATJOBN.
STATSJBI	Apply job ID filters in STATJBIL and STATJBIH. STATSJBI cannot be specified with STATSCTK.
STATSOJI	Apply original job ID filter in STATOJBI.
STATSOWN	Apply current owner filter in STATOWNR.
STATSSEC	Apply current SECLABEL filter in STATSECL.

STATSEL2 Flag byte which describes the type of jobs for which data is requested. All type bits set on (STATSTYP) **or** all bits set off select all job types.

Bit Name	Description
STATSSTC	Started tasks are selected.
STATSTSU	Time sharing users are selected.
STATSJOB	Batch jobs are selected.
STATSAPC	APPC initiators are selected. Because APPC initiators are also started tasks they are also returned if STATSSTC is specified. Use only STATSAPC to select only APPC initiators.

STATSEL3 Flag byte which describes the filters to use to select jobs. Each bit either corresponds to a filter field which must match any job returned or is a criteria for selecting jobs to return.

Bit Name	Description
STATSPRI	Apply JES job priority filter in STATPRIO.
STATSVOL	Apply SPOOL volume filters in STATVOL (this is valid only when requesting data from a JES2 subsystem).
STATSPHZ	Apply current job phase in STATPHAZ.
STATSHLD	Select jobs that are currently held. Setting both STATSHLD and STATSNHL on is the same as setting both bits off.
STATSNHL	Select jobs that are not currently held. Setting both STATSNHL and STATSHLD on is the same as setting both bits off.
STATSSYS	Only jobs active on the system listed in STATSYS are returned.
STATSMEM	Only jobs active on the JES member listed in STATMEMB are returned. (Only supported by JES2.)

STATSPOS	Include jobs queue position information for jobs awaiting execution on WLM service class queues. Setting this bit causes the fields STSCQPOS, STSCQNUM, and STSCQACT to be set if available. Calculating queue position will increase the processing overhead associated with a request. This filter is only supported by JES2 subsystems. STATVER must be set to STATV020 or greater to use this filter.

STATSEL4 Flag byte which describes the filters to use to select jobs. Each bit corresponds to a filter field which must match any job returned.

Bit Name	Description
STATSORG	Apply origin node filter in STATORGN.
STATSXEQ	Apply execution node filter in STATXEQN.
STATSSRV	Apply WLM service class filter in STATSRVC. When filtering by service class and not filtering by job number (STATSJBI) nor job phase (STATSPHZ), only jobs on the service class queue specified in STATSRVC are returned. When filtering on job number or job phase, any job assigned the service class specified in STATSRVC is returned (even if the job is not in a WLM-managed job class). Service classes are only available if the job has completed conversion processing and has not completed execution processing. This filter is only supported by JES2 subsystems. STATVER must be set to STATV020 or greater to use this filter.
STATSSEN	Apply scheduling environment filter in STATSENV.

STATSSL1 Flag byte which describes the SYSOUT filters to use to select data to return. Each bit corresponds to a filter field which must match for data to be returned. If JOB data is requested (STATTERS) then only jobs with SYSOUT that match the specified filters are returned. If SYSOUT data is requested, then data for SYSOUT that matches these filters is returned along with the corresponding job level data. These filters are only supported by JES2 subsystems. STATVER must be set to STATV030 or greater to use these filters.

Bit Name	Description
STATSCTK	Use the SYSOUT token in STATCTKN as a filter. SYSOUT tokens can be obtained from dynamic allocation or field STSTCTKN from a previous extended status request. STATSCTK and STATSJBI cannot both be specified.
STATSSOW	Apply the SYSOUT owner filter in STATSCRE.
STATSSDS	Apply the SYSOUT destination filter in STATSDES.
STATSSCL	Apply the SYSOUT class filter in STATSCLA.
STATSSWR	Apply the SYSOUT external writer filter in STATSWTR.
STATSSHL	Select SYSOUT that is currently held. This is the type of hold created by specifying HOLD=YES on the DD statement or OUTDISP=HOLD on the output card. Setting both STATSSHL and STATSSNH on is the same as setting both bits off.
STATSSNH	Select SYSOUT that is not currently held. Setting both STATSSHL and STATSSNH on is the same as setting both bits off.

STATJOBN Job name filter (used if STATSJBN is set). The name is 1-8 characters, left justified, and padded on the right with blanks. The generic characters '*' and '?' are allowed.

STATJBIL Low job ID value (used if STATSJBI is set). The job ID is 2-8 characters, left justified, and padded on the right with blanks. The JOBID must start with either the character 'J' or 'JOB' and is followed by the low job number.

STATJBIH High job ID value (used if STATSJBI is set). If this field is not specified, then information is only returned for the single job ID specified in STATJBIL. The job ID is 2-8 characters, left justified, and padded on the right with blanks. The JOBID must start with either the character 'J' or 'JOB' and is followed by the high job number.

STATOJBI Job ID value originally assigned to the job (used if STATSOJI is set). The original job ID can differ from the current job ID if the job was sent using NJE. The job ID is 2-8 characters, left justified, and padded on the right with blanks. The JOBID must start with either the character 'J' or 'JOB' a is followed by the original job number.

STATOWNR Current userid that the security product has assigned as owner of the job (used if STATSOWN is set). The owner is 1-8 character, left justified, and padded on the right with blanks. The generic characters '*' and '?' are allowed.

STATSECL Current SECLABEL that the security product has assigned to the job (used if STATSSEC is set). The SECLABEL is 1-8 character, left justified, and padded on the right with blanks. The generic characters '*' and '?' are allowed.

STATDEST Default print or punch destination assigned to the job (used if STATSDST is set). The destination 1-18 character, left justified, and padded on the right with blanks. The format of the destination is the same as that allowed on DEST= on the OUTPUT statement.

In JES2, the userid portion of the destination can contain the generic characters '*' and '?'. This can match jobs with a default print route code that contains a corresponding userid routing. However, destinations of the format 'R*', 'RM*', 'RMT*', 'U*', and 'N*' will not match jobs with a default print route code of remote, special local, or NJE.

STATORGN NJE node where the job originated (used if STATSORG is set). The origin node is 1-8 character, left justified, and padded on the right with blanks.

STATXEQN NJE node where the job is to, or was, executed (used if STATSORG is set). The execution node is 1-8 character, left justified, and padded on the right with blanks.

STATCLSL The job class associated with the job (used if STATSCLS is set). The job class is 1-8 character, left justified, and padded on the right with blanks.

In JES2, the job class can be only 1 character long. The special job classes of '$' for started tasks (STCs) and '@' for time sharing users (TSUs) are also supported.

STATVOL This keyword is supported when requesting information from a JES2 subsystem only. This field contains a list of up to four VOLSERs associated with SPOOL. A job is selected only if it has space on at least one of the specified SPOOL volumes (used if STATSVOL is set). The SPOOL VOLSERs are each 1-6 character, left justified, and padded on the right with blanks. Unused entries can be set to blanks or zero.

STATSYS The name of the MVS system on which the job must be active (used if STATSSYS is set). The job can be actively executing or active on a device on that system. The system name is 1-8 character, left justified, and padded on the right with blanks. The generic characters '*' and '?' are allowed.

STATMEMB The name of the JES member on which the job must be active (used if STATSMEM is set). The job can be actively executing or active on a device on that member. The member name is 1-8 character, left justified, and padded on the right with blanks. The generic characters '*' and '?' are allowed.

STATPRIO The 1-byte binary priority associated with the job (used if STATSPRI is set). The job's priority must match exactly to be selected.

In JES2, valid priorities are 0 to 15.

STATPHAZ The current job processing phase (used if STATSPHZ is set).

In JES2, the valid values for STATPHAZ are:

Phase Value	Description
STAT_INPUT	Job is active in input processing
STAT_WTCONV	Job is queued for conversion

STAT_CONV	Job is actively converting
STAT_VOLWT	Job is queued for SETUP (not currently used by JES2 code)
STAT_SETUP	Job is active in SETUP (not currently used by JES2 code)
STAT_SELECT	Job is queued for execution
STAT_ONMAIN	Job is actively executing
STAT_SPIN	JES2 is processing SPIN data sets for the JOB
STAT_WTBKDN	Job is queued for output processing
STAT_BRKDWN	Job is active in output processing
STAT_OUTPT	Job is on the hard copy queue
STAT_WTPURG	Job is queued for purge
STAT_PURG	Job is currently being purged
STAT_RECV	Job is active on an NJE SYSOUT receiver
STAT_WTXMIT	Job is queued for execution on another NJE node
STAT_XMIT	Job us active on an NJE JOB transmitter
STAT_EXEC	Job has not completed execution (combines multiple states in one phase request)
STAT_POSTEX	Job has completed execution (combines multiple states in one phase request)

In JES3, the valid values for STATPHAZ are:

Phase Value	**Description**
STAT_NOSUB	No subchain exists
STAT_FSSCI	Job is active in conversion/interpretation in an FSS address space
STAT_PSCBAT	Job is awaiting postscan (batch)
STAT_PSCDSL	Job is awaiting postscan (demand select)
STAT_FETCH	Job is awaiting volume fetch
STAT_VOLWT	Job is awaiting start setup

STAT_SYSSEL Job is awaiting or active in MDS system select processing

STAT_ALLOC Job is awaiting resource allocation

STAT_VOLUAV Job is awaiting unavailable volume(s)

STAT_VERIFY Job is awaiting volume mount(s)

STAT_SYSVER Job is awaiting or active in MDS system verification processing

STAT_ERROR Job encountered an error during MDS processing

STAT_SELECT Job is awaiting selection on main

STAT_ONMAIN Job is scheduled on main

STAT_BRKDWN Job is awaiting breakdown

STAT_RESTART Job is awaiting MDS restart processing

STAT_DONE Main and MDS processing complete for job

STAT_OUTPT Job is awaiting output service

STAT_OUTQUE Job is awaiting output service writer

STAT_OSWAIT Job is awaiting rsvd services

STAT_CMPLT Output service complete for job

STAT_DEMSEL Job is awaiting selection on main (demand select job)

STAT_EFWAIT Ending function request waiting or I/O completion

STAT_EFBAD Ending function request not processed

STAT_MAXNDX Maximum request index value

STATSRVC The name of the WLM service class assigned to the job (used if STATSSRV is set). Jobs only have service classes assigned to them if they have completed conversion processing and have not completed execution processing. The service class is 0-8 characters, left justified, and padded on the right with blanks.

STATSENV The name of scheduling environment (SCHENV= from the JOB statement) required by a job. (used if STATSSEN is set). Jobs only

have scheduling environments assigned to them if they have completed conversion processing and have not completed execution processing. The scheduling environment is 0-16 characters, left justified, and padded on the right with blanks. The generic characters '*' and '?' are allowed.

STATCTKN Pointer to the SYSOUT token to be used for selection (used if STATSCTK is set). The token can only be obtained from dynamic allocation or from a previous extended status request.

STATSCRE Userid that was in control when the SYSOUT data set was allocated (used if STATSSOW is set). The userid is 1-8 characters, left justified, and padded on the right with blanks. The generic characters '*' and '?' are allowed.

STATSDES Destination to which the SYSOUT is routed (used if STATSSDS is set). The destination is 1-18 characters, left justified, and padded on the right with blanks. The format of the destination is the same as that allowed on DEST= on the OUTPUT statement. IP addresses are not allowed.

In JES2, the userid portion of the destination can contain the generic characters '*' and '?'. This can match SYSOUT with a route code that contains a corresponding userid routing. However, destinations of the format 'R*', 'RM*', 'RMT*', 'U*', and 'N*' will not match SYSOUT with a route code of remote, special local, or NJE.

STATSCLA The class associated with the SYSOUT (used if STATSSCL is set). The class is 1-8 characters, left justified, and padded on the right with blanks.

Currently, only 1 character SYSOUT classes are valid.

STATSWTR The external writer name associated with the SYSOUT (used if STATSSWR is set). The external writer name is 1-8 characters, left justified, and padded on the right with blanks. The generic characters '*' and '?' are allowed.

Set all other fields in the IAZSSST control block to binary zeros before issuing the first in a series of IEFSSREQ macro calls. A memory management call (STATTYPE set to STATMEM) is required before updating output fields.

Output Register Information

When control returns to the caller, the general purpose registers contain:

Register	Contents
0	Used as a work register by the system
1	Address of the SSOB control block
2 — 13	Same as on entry to call
14	Return address
15	Return code

Return Code Information

The SSI places one of the following decimal return codes in register 15. Examine the return code to determine if the request was processed.

Return Code (Decimal)	Meaning
SSRTOK (0)	The extended status function call has completed. Check the SSOBRETN field for specific function information.
SSRTNSUP (4)	The subsystem specified in the SSIBSSNM field does not support the extended status function call.
SSRTNTUP (8)	The subsystem specified in the SSIBSSNM field exists but is not active.
SSRTNOSS (12)	The subsystem specified in the SSIBSSNM field is not defined to MVS.
SSRTDIST (16)	The pointer to the SSOB control block or the SSIB control block is not valid, or the function code specified in the SSOBFUNC field is greater than the maximum number of functions supported by the subsystem specified in the SSIBSSNM field.
SSRTLERR (20)	Either the SSIB control block or the SSOB control block has incorrect lengths or formats.
SSRTNSSI(24)	The SSI has not been initialized.

Output Parameters

Output parameters for the function routine are:

- SSOBRETN
- STATREAS
- STATREA2
- IAZSSST

SSOBRETN Contents: When control returns to the caller and register 15 contains a zero, the extended status function places one of the following decimal values in the SSOBRETN field:

Value (Decimal)	Meaning
STATRTOK (0)	Input parameters were valid, check STATJOBF for output.

STATINVA (4) The search arguments, though syntactically valid, cannot be used (for example, specifying a volume serial in STATVOL that is not being used as a SPOOL volume).

STATLERR (8) Logic error in one of the search arguments. See output parameter STATREAS (below) for details as to the exact error.

STATINVT (12) The request type in STATTYPE is not valid.

SSTREAS Contents: When SSOBRETN contains an 8 (STATLERR) indicating a logic error, the field SSSTREAS indicates the specific error detected. SSSTREAS will be set to one of the following decimal values:

Value (Decimal)	Meaning
STATRDST (4)	Destination in STATDEST is not valid.
STATRJBL (8)	Low job ID in STATJBIL is not valid.
STATRJBH (12)	High job ID in STATJBIH is not valid.
STATRJLM (16)	The high job ID in STATJBIH is less than the low job ID in STATJBIL.
STATRCLS (20)	Job class in STATCLSL is not valid.
STATRVOL (24)	The volume list in STATVOL is null or has characters that are not that are not allowed.
STATRJBH (28)	The phase specified in STATPHAZ is either not valid or not supported by this subsystem.
STATRQUE (32)	Unable to access job queue.
STATREYE (36)	The eyecatcher in STATEYE is not C'STAT'
STATRLEN (40)	The length of the IAZSSST specified in STATLEN is too short.
STATRJBN (44)	The job name in STATJOBN is not valid.
STATROWN (48)	The owning userid in STATOWNR is not valid.
STATRSYS (52)	The system name in STATSYS is not a valid system name.
STATRMEM (56)	The member name in STATMEMB is not valid.
STATRCST (60)	STATSEL2 specifies to select only non-batch jobs and batch job class selection was specified in STATSCLS.
STATROJB (64)	Original job ID in STATOJBI is not valid.
STATRSEC (68)	The SECLABEL in STATSECL is not valid.
STATRORG (72)	The origin node in STATORGN is not defined.
STATRXEQ (76)	The execution node in STATXEQN is not defined.
STATRPRI (80)	The priority in STATPRIO is not valid for this JES.
STATRSVC (84)	The service class in STATSRVC is not valid.
STATSSEN (88)	The scheduling environment in STATSSEN is not valid.
STATRSCT (92)	The SYSOUT token pointed to by STATCTKN is not valid.
STATRSCR (96)	The SYSOUT owner in STATSCRE is not valid.
STATRSSD (100)	The SYSOUT destination in STATSDES is not valid.

STATRSSC (104) The SYSOUT class in STATSCLA is not valid.

STATRSXW (108) The SYSOUT external writer in STATSWTR is not valid.

STATRECJ (112) STATSJBI and STATSCTK are mutually exclusive.

SSSTREA2 Contents: When SSOBRETN contains an 8 (STATLERR) indicating a logic error, the field SSSTREA2 can further describe the reason for the error. The content of this field is subsystem dependent. For more information contact IBM service.

IAZSSST Contents: The extended status service returns two types of data, fixed data in the IAZSSST and elements for each job that matched the filters specified. The following describes the fixed data fields returned in the IAZSSST:

Field Name	Description
STATVERO	Version level of the last subsystem to respond to the request. The first byte is the high-level version of the responder. The second byte is the service level of the responder. For a more detailed explanation of the version and service levels, refer to the IAZSSST mapping macro in SYS1.MACLIB.
STATJOBF	Pointer to a chained list of output elements that contains information about the jobs that match the input filters. There is one element per job. See “Job Information Elements” on page 135 for a description of each element. If SYSOUT information is requested, the SYSOUT output elements are chained out of the job level output element (of the owning job). See “SYSOUT Information Elements” on page 141 for a description of each SYSOUT level element.
STATNRJQ	The number of jobs that match the specified filter requirements.

Job Information Elements

For each job that matches specified filter requirements, an information element is added to the chain pointed to by STATJOBF. Each element is composed of the following:

- A variable-sized prefix (mapped by the STATJQ DSECT)
- A fixed-size job queue element header (mapped by the STATJQHD DSECT)
- One or more variable-sized data sections

Information Element Prefix: Each job information element starts with a prefix area. This area is mapped by the STATJQ DSECT in the IAZSSST macro. STATJOBF points to the start of the first prefix area. Subsequent areas are chained using the STJQNEXT field. Because the size of the prefix area can vary as a result of service being applied, do not use the equate STJQSIZE to access the data that follows the prefix. To obtain the address of subsequent fields, add the field STJQOHDR to the start of the prefix.

The fields in the STATJQ prefix are:

Field Name	Description
STJQEYE	Eycatcher C‘SJQE’.
STJQOHDR	Offset from the start of the STATJQ to the first job information data section.

STJQNEXT Address of the next STATJQ area on the STATJOBF chain.

STJQSE If SYSOUT data is requested, this is the head of the SYSOUT information elements (STATSE) for this job.

STJQOSS Name of the subsystem that created this entry.

Information Element Data Sections: The variable data sections, which contain information about the job, follow the STATJQ prefix. Each section starts with a 2-byte length, a 1-byte section type, and a 1-byte section modifier. The data length can be from 1 through 65535 bytes. The type and modifier are used to determine the mapping needed to access the data in the section. The first section after the STATJQ prefix is a special 4-byte section which describes the length and type of all sections that follow. The DSECTs that map each section are in the IAZSSST macro.

Job Queue Element 1st Section: This section is mapped by the STATJQHD DSECT and is identified by a type of STHD1HDR (0) and a modifier of STHD1MOD (0). This is the only fixed-size section with a length of STHDSIZE (4 bytes). The length in this section is the total length of all sections that follow.

The fields in the STATJQHD section are:

Field Name	Description
STHDLEN	Length of all sections which follow (including this section)
STHDTYPE	Section type identifier of STHD1HDR (0)
STHDMOD	Section type modifier of STHD1MOD (0)
STHDSIZE	Length of this section (4 bytes)

Job Queue Element Terse Section: This section is mapped by the STATJQTR DSECT and is identified by a type of STTRTERS (1) and a modifier of STTRTMOD (0). All job information elements have at least one section of this type. This section contains information common to all types of jobs.

The fields in the STATJQTR section are:

Field Name	Description
STTRLEN	Length of this section
STTRTYPE	Section type identifier of STTRTERS (1)
STTRMOD	Section type modifier of STTRTMOD (0)
STTRNAME	Job name
STTRJID	Job ID
STTROJID	Original job ID. This might be different from STTRJID if the job was sent using NJE.
STTRCLAS	Job execution class. In JES2, started tasks (STCs) have a job class of '$' and time sharing users (TSUs) have a job class of '@'.
STTRONOD	Job's origin node
STTRXNOD	Job's execution node

STTRPRND The default print node for the job

STTRPRRE The default print remote or userid for the job

STTRPUND The default punch node for the job

STTRPURE The default punch remote for the job

STTROUID The userid currently assigned as the owner of the job by the security product.

STTRSECL The SECLABEL currently assigned to the job by the security product.

STTRSYS MVS system name where the job is active (blank if the job is not active).

STTRMEM JES member name where the job is active (blank if the job is not active).

STTRDEVN JES device name on which the job is active (blank if the job is not active on a device).

STTRPHAZ Current job phase. See STATPHAZ for a list of possible values.

STTRHOLD Current hold state for the job.

Field Value	Description
STTRJNHL	Job is not held
STTRJHLD	Job is held
STTRJHLD	Job is held for duplicate job name

STTRJTYP Type of job

Field Value	Description
STTRSTC	Started task
STTRTSU	Time sharing user
STTRJOB	Batch job
STTRAPPC	APPC initiator

STTRPRIO Job's priority

STTRARMS Job's automatic restart manager status

Bit Value	Description
STTRARMR	Job is automatic restart manager registered
STTRARMW	Job is awaiting automatic restart manager restart

STTRMXRC The return code for the job if it has completed execution. The first byte is an indicator of how the job completed. The following three bytes contain a code associated with completion status. This information is not returned by JES3 subsystems.

Field Name	Description
STTRXIND	Indicator of how the job ended. The first two bits indicate if the field STTRMSCC contains a value. The remaining six bits contain the actual completion type.

Field Name	Description
STTRXAB	If this bit is on, STTRMXCC contains an ABEND code.
STTRXCDE	If this bit is on, STTRMXCC contains a completion code.
STTRXUNK	No completion information is available. This might be because the job has not completed or when the job did complete, the completion information was not saved.
STTRXNRM	Job ended normally
STTRXCC	Job ended by completion code
STTRXJCL	Job had a JCL error
STTRXCAN	Job was canceled
STTRXABN	Job ABENDed
STTRXCAB	Converter ABENDed while processing the job
STTRXSEC	Job failed security checks
STTRXEOM	Job failed in end-of-memory

STTRMXCC	Code associated with completion. If the job ABENDed (STTRXAB is on), then the first 12 bits is the SYSTEM ABEND code, and the next 12 bits is a USER ABEND code.

Job Queue Element JES2 Terse Section: This section is mapped by the STATJ2TR DSECT and is identified by a type of STJ2TERS (2) and a modifier of STJ2TMOD (0). This section is present if the job information came from a JES2 subsystem. This section contains JES2-specific information common to all types of jobs.

The fields in the STATJ2TR section are:

Field Name	Description
STJ2LEN	Length of this section
STJ2TYPE	Section type identifier of STJ2TERS (2)
STJ2MOD	Section type modifier of STJ2TMOD (0)

STJ2FLG1 General flag byte

Bit Value	Description
STJ21PRO	Job is protected
STJ21IND	Job is set to independent mode
STJ21SYS	Job represents a system data set
STJ21CNW	Job can only be processed by a converter that can wait for OS resources

STJ2JKEY The JES2 job key for the JOB

STJ2SPOL The SPOOL token associated with the job

STJ2SPAC Number of track groups of SPOOL space used by the job (a value of -1 indicates that the count is not available).

Job Queue Element Member Affinity Section:This section is mapped by the STATAFFS DSECT and is identified by a type of STAFFIN (3) and a modifier of STAFTMOD (0). This section is present if the job has affinities to a subset of members. This section is not present if the job can run on any member.

The fields in the STATAFFS section are:

Field Name	Description
STAFLEN	Length of this section
STAFTYPE	Section type identifier of STAFFIN (3)
STAFMOD	Section type modifier of STAFTMOD (0)
STAFNUM	Number of members for which the job has affinity
STAFMEMB	First member for which job has affinity. Other member names follow after this member name. The number of member names present is in field STAFNUM.

Job Queue Element Execution Scheduling Section:

This section is mapped by the STATSCHD DSECT and is identified by a type of STSCHED ('04'x) and a modifier of STSCTMOD ('00'x). This section is present if the job is scheduled for execution. This section is not returned by JES3 subsystems.

The fields in the STATSCHD section are:

Field Name	Description
STSCLEN	Length of this section.
STSCTYPE	Section type identifier of STSCHED ('04'x)
STSCMOD	Section type modifier of STSCTMOD ('00'x)

STSCAHLD Reasons why the job will not run

Bit Value	Description
STSCJCLS	Job class is held
STSCJCLM	Job class limit has been reached
STSCJSCH	Scheduling environment is not available
STSCJAFF	Systems for which the job has affinity are not available
STSCJSPL	Spool volumes needed by the job are not available
STSCJBSY	Job is busy on a device
STSCNOSY	No system(s) with the correct combination of resources is available

STSCFLG1 General flag byte

Bit Value	Description
STSC1JCM	Mode of the JOBCLASS. Off is JES mode, on is WLM mode.

STSCSRVC Service class associated with the job

STSCESTT Estimated time to execute (in seconds) for the job. This is only available if the job:

Is awaiting execution
Is scheduled to a WLM-managed job class
Is not held
Can currently run (STSCAHLD is zero)

If the estimated time is not available, this field is set to negative 1 (–1). The time is calculated on the average queue time for a job in this job class (STSCAVGQ) and the amount of time this job has been queued (STSCQTIM). If the job has been waiting longer than average, STSCESTT will be set to negative 1 (–1).

STSCSENV Scheduling environment required by the job.

STSCQPOS Position of this job on a WLM service class queue (if STATSPOS is on)

STSCQNUM Number of jobs on this WLM service class queue (if STATSPOS is on)

STSCQACT Number of active jobs on this WLM service class queue (if STATSPOS is on)

STSCAVGQ Average queue time for jobs in this WLM service class. STSCAVGQ is one component of STSCESTT. If STSCESTT is not available, this field is zero (0). If the job has already waited more than the average wait time, this field (and STSCQTIM) is set to negative 1 (–1).

STSCQTIM Actual queue time for this job. STSCQTIM is one component of STSCESTT. If STSCESTT is not available, this field is zero (0). If the job has already waited more than the average wait time, this field (and STSCAVGQ) is set to negative 1 (–1).

Job Queue Element Schedulable Systems Section:

This section is mapped by the STATSCHS DSECT and is identified by a type of STSCHED ('04'x) and a modifier of STSSTMOD ('01'x). This section is present if the job is scheduled for execution, requires a scheduling environment, and that environment is available on at least one system. This section lists the MVS system names where the scheduling environment is available. This section is not returned by JES3 subsystems.

The fields in the STATSCHS section are:

Field Name	Description
STSSLEN	Length of this section.
STSSTYPE	Section type identifier of STSCHED ('04'x)
STSSMOD	Section type modifier of STSSTMOD ('01'x)
STSSNUM	Number of systems that have the required scheduling environment.
STSSSYS	Name of first system that has the required scheduling environment. Other system names follow after this system name. The number of system names present is in field STSSNUM.

SYSOUT Information Elements

When SYSOUT information is requested, for each SYSOUT element that matches specified filter requirements, a SYSOUT information element is added to the corresponding job level information element (STATJQ) chain pointed to by STJQSE. Each element is composed of the following:

- A variable-sized prefix (mapped by the STATSE DSECT)
- A fixed-size SYSOUT element header (mapped by the STATSEHD DSECT)
- One or more variable-sized data sections

SYSOUT Information Element Prefix:

Each SYSOUT information element starts with a prefix area. This area is mapped by the STATSE DSECT in the IAZSSST macro. STJQSE of the corresponding job information element (STATJQ) points to the start of the first prefix area. Subsequent areas for the same job are chained using the STSEJNXT field. Because the size of the prefix area can vary as a result of service being applied, do not use the equate STSESIZE to access the data that follows the prefix. To obtain the address of subsequent fields, add the field STSEOHDR to the start of the prefix.

The fields in the STATSE prefix are:

Field Name	Description
STSEEYE	Eyecatcher C‘SOUT’

STSEOHDR	Offset from the start of the STATSE to the first SYSOUT information data section.
STSEJNXT	Address of the next STATSE area for this job.
STSEJOB	Address of the STATJQ for the job that owns this SYSOUT.

SYSOUT Information Element Data Sections:

The variable data sections which contain information about the SYSOUT follow the STATSE prefix. Each section starts with a 2-byte length, a 1-byte section type, and a 1-byte section modifier. The data length can be from 1 through 65535 bytes. The type and modifier are used to determine the mapping needed to access the data in the section. The first section after the STATSE prefix is a special 4-byte section which describes the length and type of all sections that follow. The DSECTs that map each section are in the IAZSSST macro.

SYSOUT Queue Element 1st Section:

This section is mapped by the STATSEHD DSECT and is identified by a type of STSH1HDR ('40'x) and a modifier of STSH1MOD ('00'x). This is the only fixed-size section with a length of STSHSIZE (4 bytes). The length in this section is the total length of all sections that follow.

The fields in the STATSEHD section are:

Field Name	Description
STSHLEN	Length of all sections which follow (including this section)
STSHTYPE	Section type identifier of STSH1HDR ('40'x)
STSHMOD	Section type modifier of STSH1MOD ('00'x)
STSHSIZE	Length of this section (4 bytes)

SYSOUT Element Terse Section:

This section is mapped by the STATSETR DSECT and is identified by a type of STSTTERS ('41'x) and a modifier of STSTTMOD ('00'x). All job information elements have at least one section of this type. This section contains information common to all types of jobs.

The fields in the STATSETR section are:

Field Name	Description
STSTLEN	Length of this section.
STSTTYPE	Section type identifier of STSTTERS ('41'x)
STSTMOD	Section type modifier of STSTTMOD ('00'x)
STSTOUID	Userid that owns the SYSOUT
STSTSECL	SECLABEL assigned to the SYSOUT
STSTDEST	SYSOUT's destination
STSTCLAS	Class assigned to the SYSOUT

STSTNREC Number of records in the SYSOUT element

STSTPAGE Number of pages in the SYSOUT element

STSTLNCT Number of lines in the SYSOUT element (JES3 only)

STSTBYCT Number of bytes in the SYSOUT element (JES3 only)

STSTFORM Form assigned to the SYSOUT

STSTFCB Forms control buffer (FCB)

STSTUCS Universal character set (UCS)

STSTXWTR External writer name

STSTPMDE Processing mode (PRMODE)

STSTFLSH Flash

STSTCHAR Character sets assigned to the SYSOUT (JES3 only)

STSTMODF MODIFY=(modname) value (JES3 only)

STSTMODC MODIFY=(,trc) value (JES3 only)

STSTSYS MVS system name where output currently being processed (blank if not currently active)

STSTMEM JES member name where output currently being processed (blank if not currently active).

STSTDEVN Device name on which output currently being processed (blank if not currently active)

STSTHSTA Current hold status of the SYSOUT

Bit Value	Description
STSTHOPR	An operator hold has been set using an operator command
STSTHUSR	A user hold has been set using JCL (such as HOLD=YES on the DD statement)
STSTHSYS	A system (error) hold has been set (see STSTHRSN for hold reason)
STSTHTSO	SYSOUT is held for TSO (JES3 only)
STSTHXWT	SYSOUT is held for an external writer (JES3 only)

STSTHRSN System hold reason (see fields OHLDJxxx in IAZOHLD for the definition of possible values)

STSTDISP Current OUTDISP value for the SYSOUT (JES2 only)

Field Value	Description
STSTDHLD	OUTDISP=HOLD
STSTDLVE	OUTDISP=LEAVE
STSTDWRT	OUTDISP=WRITE
STSTDKEP	OUTDISP=KEEP

STSTFLG1 General flag byte

Bit Value	Description
STST1BRT	BURST=YES requested
STST1DSI	3540 held SYSOUT element
STST1IPA	SYSOUT destination includes an IP address
STST1CPD	SYSOUT element includes page mode data
STST1SPN	SYSOUT element was spun

STSTPRIO Priority assigned to the SYSOUT

STSTSODI EBDCIC SYSOUT identifier which can be used in operator commands for this SYSOUT element. The contents of this field are subsystem dependent and can change from one release to another.

STSTCTKN SYSOUT token associated with the SYSOUT element. This token can be passed on subsequent extended status requests or on the SYSOUT API (SAPI). This token may be different that the SYSOUT token returned by dynamic allocation. This is currently only returned by JES2 subsystems.

SYSOUT Element JES2 Terse Section:

This section is mapped by the STATSJ2T DSECT and is identified by a type of STS2TERS ('42'x) and a modifier of STS2TMOD ('00'x). This section is present if the SYSOUT information came from a JES2 subsystem. This section contains JES2-specific information common to all SYSOUT.

The fields in the STATSJ2T section are:

Field Name	Description
STS2LEN	Length of this section.
STS2TYPE	Section type identifier of STS2TERS ('42'x)
STS2MOD	Section type modifier of STS2TMOD ('00'x)
STS2FLG1	General flag byte

Bit Value	Description
STS21DSH	JOE representing this SYSOUT data set has been cloned
STS21TSO	JOE is available for TSO OUTPUT processing

STS2OGNM JOE output group name

STS2CRTM JOE create time (STCK format system clock time)

Example

The following is a coded example of a program that generates an extended status function call (SSI function code 80).

This program is reentrant, and must run in an authorized library.

```
STATUS2  TITLE 'Sample expanded status SSI call'
STATUS2  CSECT ,
STATUS2  AMODE 31
STATUS2  RMODE ANY

         USING STATWORK,R10        Est work area addressability
         USING STATMAIN,R12        Est base addressability

STATMAIB STM   R14,R12,12(R13)     Save callers registers
         LR    R12,R15             Set base register
         LR    R8,R1               Save CPPL address

         STORAGE OBTAIN,LENGTH=STATWLEN,ADDR=(R10),LOC=ANY            C
                                   Obtain local work area

         LR    R0,R10              Zero the
         LA    R1,STATWLEN           work area
         SLR   R15,R15                 that was
         MVCL  R0,R14                    just obtained

         ST    R13,SAVEAREA+4      Chain
         LA    R15,SAVEAREA          in
         ST    R15,8(R13)              new
         LR    R13,R15                    save area

*********************************************************************
*        Determine the local userid                                 *
*********************************************************************

         IAZXJSAB READ,USERID=THISUSER  Get execution user ID

*********************************************************************
*        Set up basic extended status SSOB                          *
*********************************************************************

         USING SSOB,STSSOB         Est SSOB addressability

         LA    R0,STSSOB           Ensure that
         LA    R1,L'STSSOB           the SSOB
         SLR   R15,R15                 area is
         MVCL  R0,R14                    all zero

         MVC   SSOBID,=C'SSOB'     Set SSOB eyecatcher
         MVC   SSOBLEN,=Y(SSOBHSIZ)  Set length of SSOB header
         MVC   SSOBFUNC,=Y(SSOBESTA)  Set status 2 function code
         MVC   SSOBSSIB,=F'0'      Use LOJ SSIB
         LA    R0,SSOB+SSOBHSIZ    Point to STAT extension
         ST    R0,SSOBINDV         Point base to extension

         USING STAT,SSOB+SSOBHSIZ  Est STAT extension addr'blty

         MVC   STATEYE,=C'STAT'    Move in the eyecatcher
         MVC   STATLEN,=Y(STATSIZE)  Set length of extension
         MVC   STATVER,=AL1(STATCVRL,STATCVRM) Set current version
         MVI   STATTYPE,STATTERS   Set terse data request
```

```
*********************************************************************
*        Make only filter this userid                               *
*********************************************************************

         OI    STATSEL1,STATSOWN   Indicate OWNER is a filter
         LA    R0,STATOWNR         Get area in STAT
         LA    R1,L'STATOWNR         and length
         LA    R14,THISUSER        Get this userid
         LA    R15,L'THISUSER        and length
         ICM   R15,B'1000',=C' '   Pad with blanks
         MVCL  R0,R14              Copy parm to STAT

*********************************************************************
*        Call the subsystem                                         *
*********************************************************************

         MODESET MODE=SUP          Supervisor state for SSI function

         LA    R1,STSSOB           Get SSOB address
         O     R1,=X'80000000'     Indicate last SSOB
         ST    R1,PARMPTR          Set parm pointer
         LA    R1,PARMPTR          Get R1 for IEFSSREQ
         IEFSSREQ                  Issue extended status SSI call
         LTR   R15,R15             Any SSI errors?
         BNZ   SSREQERX              Yes, go process errors

         MODESET MODE=PROB         Return to problem program state

*********************************************************************
*        Process results for IEFSSREQ here                          *
*********************************************************************

         USING STATJQ,R4           Est STATJQ addressability

         LA    R4,STATJOBF-(STJQNEXT-STATJQ)  Get 0th STATJQ
LOOPSTJQ ICM   R4,B'1111',STJQNEXT Get next area
         BZ    DONESTJQ              No more, done with STATJQs
         LH    R3,STJQOHDR         Get length of STATJQ
         LA    R5,STATJQ(R3)       Point to 1st section
         SLR   R2,R2                           Get total
         ICM   R2,B'0011',STHDLEN-STATJQHD(R5)   Header length
         LA    R5,STHDSIZE(R5)     Point to 1st variable section
         SL    R2,=A(STHDSIZE)     Decriment for 1st header length

LOOPSECT CLC   2(2,R5),=AL1(STTRTERS,STTRTMOD)  Terse section?
         BNE   NOTTERSE              No, check next type

         USING STATJQTR,R5         Est Terse section addr'blty
*        Process terse section data
         DROP  R5                  Drop terse section
         B     NEXTSECT            Go process next section
```

```
NOTTERSE CLC   2(2,R5),=AL1(STJ2TERS,STJ2TMOD)  JES2 section?
         BNE   NOTJES2               No, check next type

         USING STATJ2TR,R5         Est JES2 section addr'blty
*        Process JES2 section data
         DROP  R5                  Drop JES2 section

         B     NEXTSECT            Go process next section

NOTJES2  CLC   2(2,R5),=AL1(STAFFIN,STAFTMOD)  Affinity section?
         BNE   NEXTSECT              Not known, get next section

         USING STATAFFS,R5         Est Affinity section addr'blty
*        Process JES2 section data
         DROP  R5                  Drop Affinity section

NEXTSECT SLR   R15,R15             Get length of
         ICM   R15,B'0011',0(R5)     current section
         SR    R2,R15              Decrement total count
         BNP   LOOPSTJQ              None left, loop
         ALR   R5,R15              Point to next section
         B     LOOPSECT            Loop for all sections

DONESTJQ DS    0H                  Done processing all elements

*************************************************************************
*        Return data area passed                                        *
*************************************************************************

         MODESET MODE=SUP          Supervisor state for SSI function

         MVI   STATTYPE,STATMEM    Set memory management call

         LA    R1,STSSOB           Get SSOB address
         O     R1,=X'80000000'     Indicate last SSOB
         ST    R1,PARMPTR          Set parm pointer
         LA    R1,PARMPTR          Get R1 for IEFSSREQ

         IEFSSREQ                  Issue extended status SSI call

         MODESET MODE=PROB         Return to problem program state
         B     EXIT                Go exit the command processor

SSREQERX LR    R2,R15              Save return code
         MODESET MODE=PROB         Return to problem program state
         LR    R15,R2              Restore return code
         B     SSREQERR            Go process error
```

```
*********************************************************************
*        Process IEFSSREQ error return codes                        *
*********************************************************************

         USING GFDSECTD,R1         Est general failure parm list

SSREQERR LA    R1,FAILPARM         Get address of fail parm area
         ST    R1,PARMPTR          Save in pointer word

         ST    R15,GFRCODE         Save IEFSSREQ return code
         MVC   GFCALLID,=Y(GFSSREQ)  Indicate IEFSSREQ error
         ST    R8,GFCPPLP          Save CPPL pointer addr
         MVC   ECBADS,=F'0'        Zero ECB address
         LA    R0,ECBADS           Set ECB address
         ST    R0,GFECBP             into the PPL

         LA    R1,PARMPTR          Get addr of parm pointer
         LINK  EP=IKJEFF19         Call TSO GNRLFAIL service

         B     EXIT                Return to caller

         DROP  R1                  Drop GFDSECTD

*********************************************************************
*        Return to the caller                                       *
*********************************************************************

EXIT     L     R13,SAVEAREA+4      Get callers save area

         STORAGE RELEASE,LENGTH=STATWLEN,ADDR=(R10)                  C
                                   Return local work area

         L     R14,12(R13)         Restore callers
         LM    R0,R12,20(R13)        registers
         SLR   R15,R15             Set a zero return code
         BR    R14                 Return to caller

         DROP  R10,R12             Drop STATWORK, Local

         LTORG ,
```

```
*********************************************************************
*        Work area DSECT                                            *
*********************************************************************

STATWORK DSECT ,
SAVEAREA DS    18F                 Save area

THISUSER DS    CL8                 This user ID

PARMPTR  DS    A                   Pointer for MVS calls

ECBADS   DS    F                   CMD processor ECB

FAILPARM DS    XL(GFLENGF)         Parm area for GNRLFAIL

STSSOB   DS    XL(SSSTLEN8)        Enhanced status SSOB

STATWLEN EQU   *-STATWORK          Length of local storage area

*********************************************************************
*        Equates                                                    *
*********************************************************************

R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*********************************************************************
*        TSO and MVS DSECTs                                         *
*********************************************************************

         IKJEFFGF GFDSECT=YES
         IEFJESCT ,
         IEFJSSOB ,
         IAZSSST DSECT=YES
         IAZJSAB ,
         IHAPSA ,
         IHAASCB ,
         IHAASSB ,
         IKJTCB ,
         IHASTCB ,
         CVT   DSECT=YES

STATUS2  CSECT ,
         END   ,
```

JES Client/Server Print Interface

In OS/390 Release 5, JES provides an interface for a job to function as a server and make SYSOUT requests on behalf of a client.

There are several ways in which a data set created by a server differs from a data set created by an ordinary SYSOUT DD or dynamic allocation.

1. Data sets created by a server use the DALRTCTK dynamic allocation text unit, which causes JES to create a unique **Client Token**(CTOKEN) associated with the data set from that point on.
2. The server can use the Extended Status or SYSOUT Application Programming Interface (SAPI) Subsystem Interface (SSI) calls to access a data set, specifying a CTOKEN in the selection criteria in order to request a particular data set, without needing to know any other information about the data set.
3. When a data set has a CTOKEN, JES informs the application, through the use of ENF signal 58, of events relating to the data set. Among these events are selection by a writer, deselection by a writer, and data set purge. JES also issues signals for important events related to any job that has created at least one data set with a CTOKEN, such as job purge.

Creating a CTOKEN

The server creates an 80-byte CTOKEN using the Dynamic Allocation text unit of DALRTCTK. The DALRTCTK text unit appears as follows:

+0	+2	+4	+6
DALRTCTK	00 01	00 50	Returned data

Upon return from SVC 99, the data starting at position 6 in the CTOKEN text unit contains the CTOKEN returned by JES, provided the allocation was successful. When a CTOKEN is returned to you, add it to your list of CTOKENs for later use.

CTOKENs contain internal information that JES uses to locate the data set when the server issues SSI requests. Once you have received a CTOKEN from JES, do not change its contents except in one special case that will be discussed later.

CTOKENs contain ordering information. This allows you to store CTOKENs in a data structure of your choice that can make use of the order and result in faster searches. CTOKENs are not ordered according to a creation timestamp; they are ordered internally by JES.

Refer to “SSI Function Codes Your Program Can Request” on page 13 and SSI 54 for a detailed description of the Subsystem Version Information Call.

Determining If You Can Request a CTOKEN

The DALRTCTK text unit is available only when JES2 or JES3 is at the OS/390 Release 5 level, or higher. If a DYNALLOC request is made with this text unit below the required JES level, the allocation will fail with a return code of 4 and an S99ERROR code of x'02CC'.

There are two ways you can determine whether the JES level you are running on can accept the DALRTCTK text unit. One is to try the allocation and intercept the return and error code. Another way is to make a subsystem interface request using the Subsystem Version Information SSI code (54). If JES is at the required level, the returned information string contains CLIENT_PRINT='YES'. If not, this string will not appear at all. The information string will never contain CLIENT_PRINT='NO'.

Comparing CTOKENs

At various times during your processing, you will need to compare CTOKENs. Typically, you will do this when JES signals an event for a CTOKEN and you need to find this token in your list so that you can take some kind of action based on the event.

To compare one CTOKEN to another, you must not simply compare the entire 80 byte values. This is because under certain JES processing, CTOKEN equality is based on a subset of the information in the CTOKENs matching while other information in the CTOKENs could be different. IBM provides a macro IAZXCTKN which you must use to compare CTOKENs. This macro determines which information in two CTOKENs is significant and compares just this information. The macro works in such a way that you never need to interpret any information inside the CTOKEN.

Depending on the return code from the IAZXCTKN macro, you can determine whether the two CTOKENs are the same, whether the first CTOKEN is less than the second one, or whether the second CTOKEN is less than the first one.

IAZXCTKN also provides a special comparison function. At certain times, JES signals events for an entire job. When this happens, the signal includes a **job level CTOKEN**. Using IAZXCTKN, you can determine whether a CTOKEN for a data set in which you are interested is covered by the job level CTOKEN that JES provides.

Job level CTOKENs contain no ordering information; therefore a job level CTOKEN can be considered by IAZXCTKN to be "equal" or "not equal" to another CTOKEN but never "greater" or "less" than another CTOKEN.

Refer to the book *OS/390 MVS Programming: Authorized Assembler Services Guide* for information about using the IAZXCTKN macro.

Obtaining Status for a Data Set

You can obtain status for a data set using the Extended Status subsystem interface (SSI 80) code. To do this, you supply as STATCTKN the address of the CTOKEN for the data set you are interested in and set the selection flag STATSCTK. When you use the STATSCTK selection flag, you cannot use the STATSJBI selection flag, and vice versa.

Refer to “SSI Function Codes Your Program Can Request” on page 13 and SSI 80 for a detailed description of the Extended Status call.

Accessing a Data Set

You can access a data set using the SYSOUT Application Programming Interface, SSI 79. You would do this in order to:

- Show the contents of the data set to the requesting client.
- Allow the client to delete a data set.
- Allow the client to release a data set from hold to print.

To request JES to perform a SAPI operation on a client data set, you supply as SSS2CTKN the address of the CTOKEN for the data set you are interested in and set the selection flag SSS2SCTK. When you use the SSS2SCTK selection flag, you cannot use the SSS2SJBI selection flag, and vice versa.

When a data set is processed by a program written using SAPI, there is a distinction between a data set that is selected for processing and a data set that is selected for browsing. In the former case, the intention is to select the data set in much the same way as it would be selected for a writer (such as an external writer), which may or may not cause its state to be changed. In the latter case, the intention is to not change its state at all. The main purpose for this distinction is to prevent "noise" caused by unnecessary ENF signals.

You can set the flag SSS2SBRO when you know that the intention of a SAPI access is to browse a data set. When this flag is on, JES will not issue any signals for the SAPI access to this data set. When this flag is off JES will issue signals whenever a data set with a CTOKEN is selected or deselected by a SAPI Put/Get operation. Do not set this flag if you need to be informed of selects and deselects.

This flag controls signals for selects and deselects only. If a data set is purged by a SAPI Put operation (for example, by turning off flag SSS2DKPE), a signal will be issued even if SSS2SBRO is set.

You must use SAPI in order to suppress signals when accessing a data set for browse. When a Process Sysout (PSO) application selects or deselects a data set with a CTOKEN, a signal is always issued.

The SSS2SBRO flag is valid only for Put/Get requests.

Refer to “SSI Function Codes Your Program Can Request” on page 13 and SSI 79 for a detailed description of SAPI.

Security

Since all SYSOUT allocations and SAPI calls are being done by you as the server, preventing a client from having unauthorized access to another client's data set is your responsibility.

One way you can do this is by performing the dynamic allocation to create the SYSOUT file under a security environment with the client's identity. This is accomplished by using the RACROUTE macro with REQUEST=VERIFY. Then, when a client makes a request requiring you to make a SAPI SSI call, you would use RACROUTE REQUEST=VERIFY with the requesting client's user id to

establish a security environment for the requestor. As part of the SAPI processing, JES makes authorization checks using the JESSPOOL security class.

Refer to the book *OS/390 SecureWay Security Server External Security Interface (RACROUTE) Macro Reference* for information on using the RACROUTE macro.

This method requires your clients to be defined as users in your security product, even if they never directly log on to your system. If this is not possible, you must design your own security protocol.

Identifying a Requestor on a Header Page

JES typically has printers defined to print with a header page identifying the job creating a SYSOUT data set.

However, the job information that prints on the header page is associated with the job that created the data set. This ordinarily identifies the job that runs your server, not the client that requested the printout. You would probably prefer that the client's identification print rather than the server's in order to be able to tell one client's output apart from another's.

In order to do this, you can use the IAZXJSAB macro. You could do something like the following:

```
IAZXJSAB CREATE,JOBNAME=client_jobname,
          USERID=client_userid,TYPE=SUBTASK
```

To make sure that the job identification does not persist beyond the requested data set, you can delete the JSAB by making the following call:

```
IAZXJSAB DELETE,TYPE=SUBTASK
```

or you can update it with different user identification by making the following call:

```
IAZXJSAB UPDATE,JOBNAME=new_client_jobname,
          USERID=new_client_userid,TYPE=SUBTASK
```

In all of these cases, you must use the parameter TYPE=SUBTASK, otherwise JES will not recognize the requesting user identification correctly.

Refer to the book *OS/390 MVS Programming: Authorized Assembler Services Guide* for information on using the IAZXJSAB macro.

Listening for Events

During the course of JES operations, data sets and jobs are subject to changes for various reasons. When such events occur for a client data set (for example, a data set that was allocated with the DALRTCTK text unit) or a job containing at least one client data set, JES issues an Event Notification (ENF) signal. The ENF number of the signal is 58. The signal is issued only for data sets that have been allocated using the DALRTCTK text unit.

To listen for this signal, you could do something like the following:

```
ENFREQ ACTION=LISTEN,CODE=58,EXIT=exit_address,XSYS=YES,
     PARM=parameter_address,DTOKEN=end_token_address
```

Note: The XSYS=YES parameter is used because JES could be issuing signals on a different processor from the one where your server runs.

To stop listening for this signal, you could do something like the following:

```
ENFREQ ACTION=DELETE,CODE=58,DTOKEN=end_token_address
```

The data area received by your listen exit from ENF is mapped by the IAZENF58 macro. You must include this macro in your program in order to use the data supplied by the ENF signal. This data area contains the following information:

ENF58_LENGTH Length of parameter list

ENF58_QUALIFIER Qualifier code — defined below

ENF58_Q_PURGE Data set was purged

ENF58_Q_SELECT Data set was selected

ENF58_Q_DESELECT_PROCESSED Data set was processed

ENF58_Q_DESELECT_NOT_PROCESSED Data set is no longer selected , disposition was not changed

ENF58_Q_DESELECT_NOT_PROCESSED_HELD Data set is no longer selected, disposition was not changed and data set is held

ENF58_Q_DESELECT_ERROR An error resulting in a system level hold occurred

ENF58_Q_EOD_OK End of data set notification occurred — successful

ENF58_Q_EOD_ERROR End of data set notification occurred — unsuccessful

ENF58_Q_JOB_CHANGE Job-status change occurred

ENF_Q_TOKEN_CHANGE Client token has changed

ENF58_SYS_HOLD System hold reason — refer to IAZOHLD for possible values

ENF58_JES_NAME JES2 Member Name / JES3 MAIN name

ENF58_REASON Reason text

ENF58_CTOKEN Data Set Client Token

ENF58_NEW_CTOKEN New client token that should replace the CTOKEN for a TOKEN_CHANGE ENF type

You should determine what action you need to take based on this event. For example, if you receive a signal with ENF58_Q_PURGE it usually means that you should delete from your list all information pertaining to the dataset with the CTOKEN of ENF58_CTOKEN.

To take action on the CTOKEN, you must first go through your CTOKEN list and issue IAZXCTKN macros, comparing ENF58_CTOKEN to CTOKENs from your list until you find the CTOKEN specified in the signal in your list. If ENF58_QUALIFIER is ENF58_Q_JOB_CHANGE, it means that ENF58_CTOKEN is a job level CTOKEN and you must go through your entire list of CTOKENs until you have identified, and taken action on, all data set level tokens covered by the job level CTOKEN.

Notes:

1. When ENF58_QUALIFIER is ENF58_Q_JOB_CHANGE, the CTOKEN in ENF58_CTOKEN is a job level CTOKEN. At all other times it is a data set level CTOKEN.
2. When ENF58_QUALIFIER is ENF58_Q_TOKEN_CHANGE, the ENF58 parameter list contains a new CTOKEN and ENF58_LENGTH reflects the existence of this new CTOKEN.
3. When an event with ENF58_Q_TOKEN_CHANGE is received, the CTOKEN in your list should be replaced with the contents of ENF58_NEW_CTOKEN. This is the only time that you should change the contents of a CTOKEN. Replacing this CTOKEN does not change the ordering of the CTOKEN you previously had in your list for this data set.
4. ENF58_NEW_CTOKEN is present only when ENF58 QUALIFIER is ENF58_Q_TOKEN_CHANGE. ENF58_LENGTH is larger for this qualifier type than it is for other types.

Refer to the books *OS/390 MVS Programming: Authorized Assembler Services Guide* and *OS/390 MVS Programming: Authorized Assembler Services Reference ENF-IXG* for information about using the ENFREQ macro and coding the listen exit.

Setting Up Your Subsystem

This chapter describes planning considerations for setting up and writing your own subsystem. When a directed request is made for a specific subsystem, the SSI searches for the subsystem requested. If the SSI finds that the named subsystem handles the requested function, the SSI passes control to the function routine. When a broadcast request is made, the SSI checks every subsystem to see if the subsystem handles the requested function. This search is done in the same order that the subsystems are defined to MVS, with the exception that the primary job entry subsystem (JES) is first. If the SSI finds that a subsystem handles the requested broadcast function, the SSI passes control to the function routine. This process is repeated for each subsystem that handles the requested function.

When you want to write your own subsystem, you must:

- Provide the routines to support the request for a function. These function routines get control from the SSI. They may actually perform the function or may pass control to other routines that you provide.
- Provide a subsystem address space (if required).
- Let MVS know that the subsystem exists (define the subsystem).
- Provide the information to the SSI that it will need to find your function routines (initialize the subsystem).
- Provide accounting information parameters to your subsystem (if required).

Note: When writing your own subsystem you must also provide any control blocks or resources that the subsystem requires for its own operation, which MVS does not provide.

Function Routines/Function Codes

Based on what you want your subsystem to do, you must supply one or more function routines. The same function routine can handle multiple function codes. You must decide how many separate functions you need and identify each function by a unique function code in the subsystem vector table (SSVT). The **SSVT** identifies:

- The SSI function codes to which the subsystem responds
- The subsystem routines that process the supported functions.

The MVS-defined function codes your subsystem can support are described in “SSI Function Codes Your Subsystem Can Support” on page 185. If your subsystem handles installation-defined directed requests, you must identify each function using a function code from 236 to 255. These codes are not broadcast functions. You can also subdivide installation-defined function codes by using subtypes you identify by passing parameters in your SSOB function dependent area.

If you plan to have your subsystem support the MVS-defined function codes, see the specific function code descriptions for requirements on your function routine. The sections that follow describe general considerations for all function routines you write.

Environment

On entry to a function routine, the function routine must save registers using standard save area conventions.

The register contents on entry to a function routine are:

Register	Contents
Reg 0	Address of the SSCVT (mapped by the IEFJSCVT macro)
Reg 1	Address of the SSOB control block passed by the requestor. This is explained in “Subsystem Options Block (SSOB)” on page 7.
Reg 13	Standard 18-word save area
Reg 14	Return address
Reg 15	Entry point address

On exit from a function routine, a function routine must restore registers 0 — 14 to the contents on entry using standard exit linkage.

As you write your function routines, be aware of what state and key the function routine must be in to do its work. Your function routine gets control in the key and state of the requestor. If your routine requires that it be in a different key or state, your routine must handle mode and state switching. However, you must reverse the mode switch before returning control to the SSI because the SSI gets control back in your routine's key and state.

Address mode (AMODE) considerations are handled by the SSI system routines. Other addressability considerations must be handled by the function routine. Any addresses passed to an AMODE 24 function routine (including the save area) must be below 16 megabytes. If the subsystem runs in a separate address space, the function routine must establish cross memory space communication either by SRB scheduling or cross memory instructions. For an explanation of using multiple address spaces, see *OS/390 MVS Programming: Extended Addressability Guide*.

The function routine can pass back some information when processing for the request is complete. The information is put in fields in the control blocks that the user passed to the SSI when the request was made. The control blocks (SSOB, SSIB and SSOB function dependent area) are explained more fully in “Making a Request of a Subsystem” on page 7. The function routine must:

- Set the return code in the SSOBRETN field of the SSOB
- Put information (if required) in the SSOB function dependent area.

See Appendix A, Examples — Subsystem Interface Routines for coding examples of function routines.

Recovery and Integrity

When you write a function routine, IBM recommends that you provide recovery in case your function routine fails. Your recovery routine should indicate unsuccessful processing, clean up any resources used, and return control to the SSI. You might also want to disable one or more of your supported function codes. See “Disabling Previously Supported Functions” on page 173 for more information.

Attention: Because there is no serialization used for updating the function codes in the SSVT, other requests for supported functions might be coming in asynchronously. The **SSVT** identifies:

- The SSI function codes to which the subsystem responds
- The subsystem routines that process the supported functions.

Therefore, do not delete a function routine from storage (because a task may be using it) and do not delete the SSVT.

Placement of Function Routines

Your subsystem function routines must be addressable from any address space, as the SSI gives control to the subsystem in the caller's environment. To meet this requirement, the following are the choices for placement of your function routines:

- Place your function routines in one of the data sets from which LPA (PLPA, MLPA, or FLPA) is built. That is, those specified in the LPALSTxx, IEALPAxx, or IEAFIXxx members of SYS1.PARMLIB.
- Place your function routines in one of the data sets specified in the LNKLSTxx member of SYS1.PARMLIB. Note that if SYS1.PARMLIB member IEASYSxx specifies LNKAUTH=APFTAB, this data set must also be defined in IEAAPFxx, or in the APF section of SYS1.PARMLIB member, PROGxx.

The placement of your function routines influences the setting of the load-to-global option that is used when building your SSVT or enabling functions with the IEFSSVT macro. If you decide to place your function routines in LPALSTxx, IEALPAxx, or IEAFIXxx, the load-to-global option has no effect. If you decide to place your function routines in LNKLSTxx, you must specify the load-to-global option. When set, this option causes the system to load the function routines into pageable CSA. A subsystem can choose to place all of its function routines in LPA, or in pageable CSA, or a combination of the two. See “Building the SSVT” on page 170 or “Enabling Your Subsystem for New Functions” on page 172 for more information.

Note: If you request load-to-global, the SSI, running under your task, issues a LOAD macro with the end of memory (EOM) keyword set to YES. Function routines that are loaded this way are deleted from storage if the home address space of the requesting task ends. To protect the system, you must deactivate your subsystem or disable all its function codes if the address space ends. To do this, write a function routine that gets control for broadcast function code 8 (end-of-address space). If the address space that owns the function routine ends, invoke IEFSSVT to disable your subsystem's function codes or invoke IEFSSI to deactivate your subsystem. See “Disabling Previously Supported Functions” on page 173 for information on IEFSSVT and see “Deactivating Your Subsystem” on page 175 for information on IEFSSI.

Do You Need a Subsystem Address Space?

When people think of a subsystem, they often think of JES2 or JES3. They usually do not differentiate between the JES subsystem and the JES address space. The subsystem and the address space, however, are not the same. It is just that the JES subsystem was implemented with a requirement for an address space with the same name as the subsystem.

A subsystem is not required to have its own address space, although many subsystems do have a separate address space. Remember that the subsystem routine is entered in the address space of the caller. Therefore, a major decision you need to make is where you want the subsystem to reside: in common storage or in its own address space.

As mentioned earlier, the code that gets control directly from the SSI must be addressable from any address space. That function routine, however, can pass control to your subsystem code that might reside in a separate address space.

If your subsystem requires minimal space, and your installation is not suffering from present (nor anticipating potential) storage constraints for common storage, you can keep all the routines in common storage. On the other hand, having a separate address space is useful if the subsystem needs its own data areas. You can create a separate address space by having your initialization routine use the ASCRE macro, or by having your subsystem run as a started task. See *OS/390 MVS Programming: Authorized Assembler Services Reference ALE-DYN* for information on the ASCRE macro.

Defining Your Subsystem

If you want to use dynamic SSI services, your subsystem must be defined to MVS in one of the following ways:

- IEFSSNxx parmlib member (keyword format) processing during IPL
- IEFSSI macro invocation
- SETSSI system command invocation.

The maximum number of subsystems you can define is 32,767.

If you do not want to be able to use dynamic SSI services, your subsystem must be defined to MVS at IPL time in the positional format of the IEFSSNxx parmlib member.

See *OS/390 MVS Initialization and Tuning Reference* for detailed information on the syntax and rules for coding IEFSSNxx. See *OS/390 MVS Programming: Authorized Assembler Services Reference ENF-IXG* for information on the syntax and rules for coding the IEFSSI macro. See *OS/390 MVS System Commands* for information on the syntax and rules for issuing the SETSSI system command.

There are some special things to think about when defining your subsystem, including:

- Naming your subsystem
- Passing parameters
- The primary subsystem

Naming your subsystem

The name you use for your subsystems depends on how your subsystem is defined to MVS. Use one of the following naming conventions:

- If your subsystem is defined to MVS through the IEFSSNxx parmlib member processing at IPL, the subsystem name can be no more than four characters long, beginning with an alphabetic character or #, @ or $. The remaining characters can be alphabetic, numeric, or #, @, or $.

- If your subsystem is defined to MVS through the IEFSSI macro, the subsystem name can be no more than four characters long, containing any character other than blanks or nulls.
- If you subsystem is defined to MVS through the SETSSI command, the subsystem name can contain any character other than blanks or nulls that is valid for system commands. See *OS/390 MVS System Commands* for more information on the valid characters.

You cannot use the following names for your subsystems:

- APPC
- ASCH
- MSTR
- OMVS
- STC
- SYS
- TSO

It is a good idea to use a meaningful name for your subsystem. When debugging a problem, it is much easier to recognize a meaningful name. Also, check for the subsystem names that are currently in use by IBM-supplied and vendor-supplied products.

Note: Since subsystems can be added after IPL, it is difficult to determine which unique name to use for a subsystem. You can use the query request of the IEFSSI macro to find the names of existing subsystems to ensure that your subsystem name is unique.

Passing parameters

If you want to pass parameters to the initialization routine, you can list them in one of the following:

- IEFSSNxx parmlib member during IPL
- IEFSSI macro
- SETSSI system command.

See “Initializing Your Subsystem” on page 168 for more information.

The primary subsystem

For work to be done, MVS requires that at least one subsystem be defined as a job entry subsystem (JES) to bring jobs into the system. The JES in fact is called the primary subsystem. You can select either JES2 or JES3. If you do not specify an IEFSSNxx member in SYS1.PARMLIB, MVS attempts to use the system default member, IEFSSN00. IEFSSN00, as supplied by IBM, contains the definition for the default primary job entry subsystem, JES2.

If you attempt to IPL without specifying an IEFSSNxx member and IEFSSN00 is not present or does not identify the primary subsystem, the system issues message IEFJ005I (see “Handling Initialization Errors” on page 255) and prompts the operator for the primary subsystem.

For an IPL, do not define a subsystem more than once in a combination of IEFSSNxx members that can be used together or within a single member. (The same subsystem can appear in two different IEFSSNxx members when the

members will not be used together.) In general, if MVS detects a duplicate name, both of the following are true:

- MVS does not define the duplicate subsystem
- MVS does not give control to the initialization routine.

The system issues the following message:

```
IEFJ003I:  DUPLICATE SUBSYSTEM subname NOT INITIALIZED
```

Providing a Routine to Initialize Your Subsystem

When writing your own subsystem you need to provide a routine to initialize your subsystem. You need to decide what your subsystem initialization routine will do and how you will initialize your subsystem.

What Your Subsystem Initialization Routine Can Do

One of the things that you must do to initialize your subsystem is to tell the SSI what function codes and function routines your subsystem supports. This is done by building an SSVT. The SSI provides the IEFSSVT macro to build your subsystem's SSVT. See “Building the SSVT” on page 170 for more information.

After building your subsystem's SSVT, your subsystem initialization routine must let MVS know that your subsystem is active and ready to accept SSI requests.

The following are examples of other things your subsystem initialization routine can do:

- It can tell MVS that your subsystem requires the services of a JES.
- It can define command prefix characters for your subsystem.
- It can create and anchor subsystem specific control blocks for use by its function routines.
- It can specify whether the subsystem is to respond to the SETSSI command.

For more information, see “Initializing Your Subsystem” on page 168.

How to Initialize Your Subsystem

There are two ways to initialize your subsystem:

- Specifying an initialization routine
- Using the START command

You can also combine these methods, doing part of the setup through an initialization routine, then completing initialization through a START command.

Specifying an Initialization Routine

You can optionally specify the name of your subsystem initialization routine when you define your subsystem. See “Defining Your Subsystem” on page 160 for the list of ways that subsystems are defined to MVS. If the functions the subsystem supplies might be needed during the IPL process, define your initialization routine in IEFSSNxx. In this case, the initialization routine handles all the preparation to ensure the subsystem is active.

Using the START Command

If the subsystem functions are not needed until a later time, you can use the START command to initialize your subsystem. See *OS/390 MVS System Commands* and *OS/390 MVS JCL Reference* for more information on the START command.

Figure 17 shows how you can initialize your subsystem either by specifying an initialization routine or by using the START command.

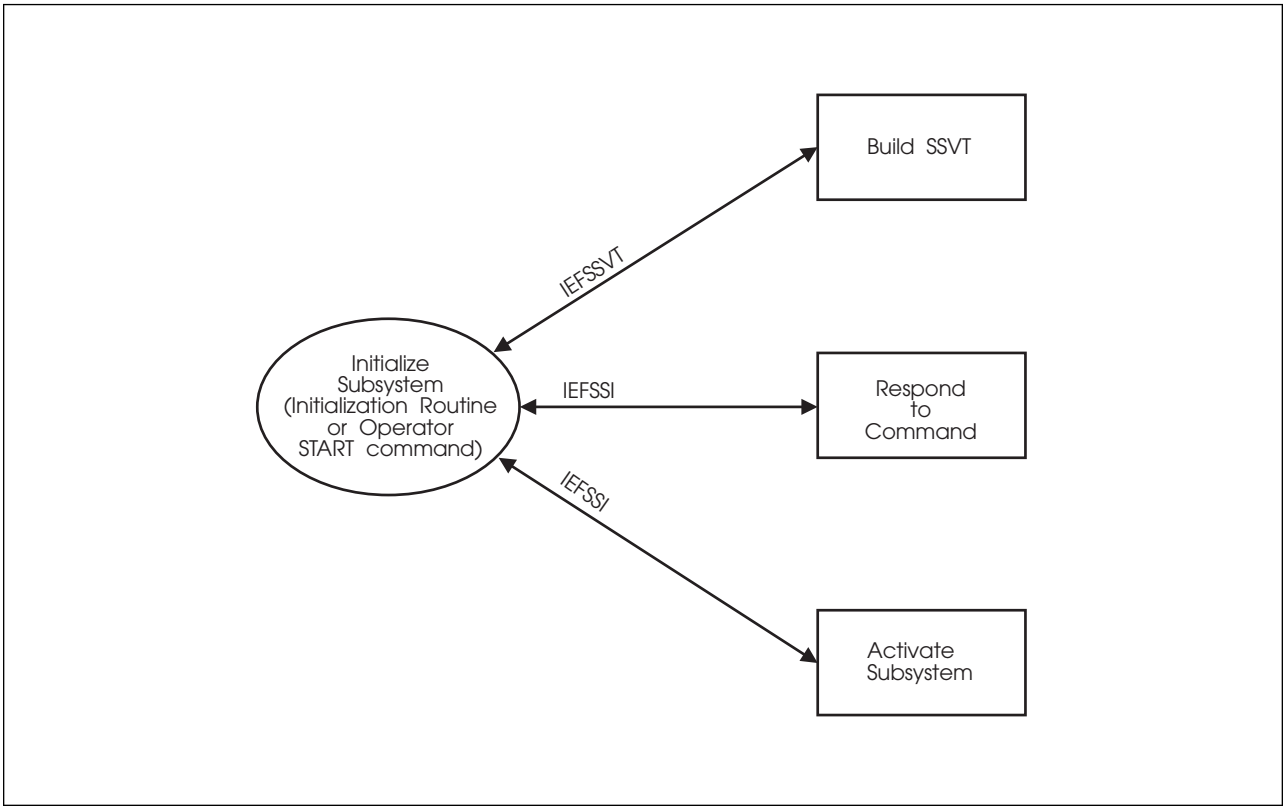

Figure 17. Initializing Your Subsystem.

Starting Your Subsystem With the START Command: You can initialize your subsystem with the START command and run under either a job entry subsystem (JES) or the MSTR subsystem.

See “Subsystem Identification Block (SSIB)” on page 8 for more information on started tasks.

MVS uses one of the following naming conventions to identify the name of the subsystem being started:

- START CAW — MVS interprets CAW as the subsystem name
- START CAW.CAW1 — MVS interprets CAW1 as the subsystem name
- START CAW,JOBNAME=CAW2 — MVS interprets CAW2 as the subsystem name.

In each case, MVS looks for the matching subsystem name that was previously defined to MVS.

If you want to start multiple instances of a specific subsystem using different names, you can, for example, define the following subsystems:

- CAW — the first instance of the CAW subsystem
- CAW1 — the second instance of the CAW subsystem
- CAW2 — the third instance of the CAW subsystem

and then specify the following with the START command:

- START CAW,JOBNAME=CAW
- START CAW,JOBNAME=CAW1
- START CAW,JOBNAME=CAW2

For more information about started tasks, see *OS/390 MVS JCL Reference*.

Passing Accounting Parameters to Your Subsystem

SMF allows your subsystem to receive a set of accounting parameters through the use of the SUBPARM option in the SMF parmlib member (SMFPRMxx). Some examples of parameters you can receive are:

- Record type number for SMF records
- Recording interval time
- Level of SMF recording (high, medium, low, or none).

The syntax of the option allows the installation to specify a subsystem name and a set of parameter values (up to 60 characters in length) that are associated with that subsystem.

See *OS/390 MVS Initialization and Tuning Reference* for more information on the SMF parmlib member and the SUBPARM option.

Processing the SUBPARM Option

The processing of SUBPARM involves the following:

- Initializing the SMF parameters
- Initializing the subsystem
- Modifying the SUBPARM value.

Initializing the SMF Parameters

During SMF initialization, the SMF parameter that the installation specified are processed and the requested actions are taken. For example, your installation can specify as parmlib options any of the following:

- Perform SMF recording
- Activate specific SMF exits.

SMF parameter initialization includes processing the SUBPARM option. That is, the value the installation specified must be stored in an SMF storage area for the subsystem's use.

Initializing the Subsystem

During subsystem initialization, the subsystem must request the SMF accounting parameter values from SMF. The subsystem uses the SMFSUBP macro to retrieve the parameter value that the installation requested. If the macro request is successful, the system returns a pointer to the specific parameter value. The system returns a non-zero return code if errors are encountered during the macro's processing. See *OS/390 MVS System Management Facilities (SMF)* for more information on the SMFSUBP macro.

Modifying the SUBPARM Value

After subsystem initialization is complete, the installation can modify the SUBPARM option value for a specified subsystem by using:

- An SMF console command
- An SMF macro.

Using an SMF Console Command

To change the SUBPARM option value with an SMF console command, use either:

- The SETSMF command
- The SET SMF=xx command.

When either of these commands is issued and causes a change to the value of the SUBPARM option for a selected subsystem, the SMF SUBPARM Option Change call (SSI function code 58) is issued to notify the specified subsystem of the change. See “SMF SUBPARM Option Change Call — SSI Function Code 58” on page 234 for a description of this function code. The SSI function code 58 parameter list does not include the changed parameter value. The subsystem can issue the SMFSUBP macro to retrieve the updated parameter values and modify its processing.

Using an SMF Macro

To change the SUBPARM option value with an SMF macro, the subsystem uses the SMFCHSUB macro. See *OS/390 MVS System Management Facilities (SMF)* for more information on the SMFCHSUB macro.

Note: Changes made by the SMFCHSUB macro do not cause SSI function code 58 to be invoked.

Example

The following steps show how an installation can pass accounting parameters to the subsystem.

- The SMF parmlib member used at SMF initialization contains:

  ```
  SUBPARM(ABCD(ONESETOFPARMS))
  ```

- During the initialization of the ABCD subsystem, ABCD issues the SMFSUBP macro to retrieve the initial parameter information.
  - During this point in the processing, the subsystem does whatever it is specified to do by checking the contents in the parameter area.
  - It then continues with its initialization.
- If the installation changes the value of the parameter, either by using the SET SMF=xx command to change parmlib members, or by using the SETSMF command as follows:

```
SUBPARM(ABCD(ANOTHERSETOFPARMS))
```

to change the value for the SUBPARM, the result is that SMF issues the SMF SUBPARM Option Change call (SSI function code 58) to the ABCD subsystem to signal the change.

- Subsystem ABCD could be any of the following:
  - Undefined, which causes an SSI error
  - Not enabled for the function code, which means no action
  - Enabled for the function code, which invokes the subsystem's routine for the function code.
- The function routine uses the SMFSUBP macro to retrieve the updated parameter information.
- At this point in the processing, the subsystem processing depends on the contents of the parameter area, which will probably update controls for the subsystem.

Services for Building and Using Your Subsystem

This chapter describes MVS services that are provided to help you build and use your subsystems when performing the following tasks:

- Adding your subsystem
- Initializing your subsystem
- Defining what your subsystem can do
- Changing what your subsystem can do
- Activating your subsystem
- Deactivating your subsystem
- Swapping subsystem functions
- Storing and retrieving subsystem-specific information
- Defining subsystem options
- Querying subsystem information
- Maintaining information about your subsystem

Adding Your Subsystem

To dynamically add your subsystem, you can use:

- The keyword format IEFSSNxx parmlib member
- The IEFSSI macro
- The SETSSI command

When you add and define a subsystem, you make the subsystem's name known to the system. Previously, the only way to add a subsystem was to add and define it in the positional format IEFSSNxx parmlib member, which meant that an addition of a new subsystem required you to re-IPL the system.

You can still add a subsystem with the positional format IEFSSNxx parmlib member; however, you cannot use the dynamic SSI services if you add a subsystem this way.

Using the IEFSSNxx Parmlib Member

Both the positional and the keyword format IEFSSNxx parmlib member allow the installation to specify the following information about a subsystem:

- The subsystem name
- The subsystem initialization routine
- The parameters to be passed to the initialization routine
- For the primary subsystem, whether it should be automatically started during master scheduler initialization

Use the keyword format IEFSSNxx parmlib member to dynamically add a subsystem, which allows you to specify the following additional information about a subsystem during subsystem definition processing:

- The console to which messages issued by the SSI will be directed.
- The console to which messages issued by the subsystem initialization routine will be directed.

The installation or subsystem can use the CONSNAME parameter of an IEFSSNxx parmlib entry to specify a console name. The SSI does not verify that the named

console is defined or active. If you specify a console name that is not valid, the standard write-to-operator processing occurs. If you do not specify a console name, messages are directed to the master console.

The console name is passed to the subsystem initialization routine in the parameter list mapped by IEFJSIPL. The initialization routine can use the console name when issuing messages.

Specifying a console name is important only during subsystem initialization. After subsystem initialization, SSI messages are issued in response only to dynamic SSI commands; such as, SETSSI and DISPLAY SSI. These messages are issued to the console from which the command was issued, or in the case of the DISPLAY SSI command, to the specified console, if any.

See *OS/390 MVS Initialization and Tuning Reference* for the syntax of the keyword format IEFSSNxx parmlib member.

Using the IEFSSI macro

Use the add request of the IEFSSI macro to dynamically add a subsystem and allow you to use dynamic SSI services. As with using the IEFSSNxx parmlib member, the installation or subsystem can use the CONSNAME parameter of the IEFSSI macro to specify a console name.

Using the SETSSI command

Use the SETSSI ADD command to dynamically add a subsystem and allow you to use dynamic SSI services. As with using the IEFSSNxx parmlib member and the add request of the IEFSSI macro, the installation or subsystem can use the CONSNAME keyword of the SETSSI command to specify a console name.

Initializing Your Subsystem

If you are defining your own subsystem, you can code an initialization routine and have control pass to that routine by specifying the name of the initialization routine when you define your subsystem. You can define parameters to be passed to your initialization routine.

The initialization routine is linked to in supervisor state and key zero. On entry to the routine, there are no locks held and register 1 points to a two-word parameter list:

Word	Contents
One	Address of the SSCVT (mapped by the IEFJSCVT macro).
Two	Address of the subsystem initialization parameter list (JSIPL, mapped by IEFJSIPL). See *OS/390 MVS Data Areas, Vol 3 (IVT-RCWK)* for the format of JSIPL.

Figure 18 on page 169 shows the input to the initialization routine when your initialization routine gets control from the system.

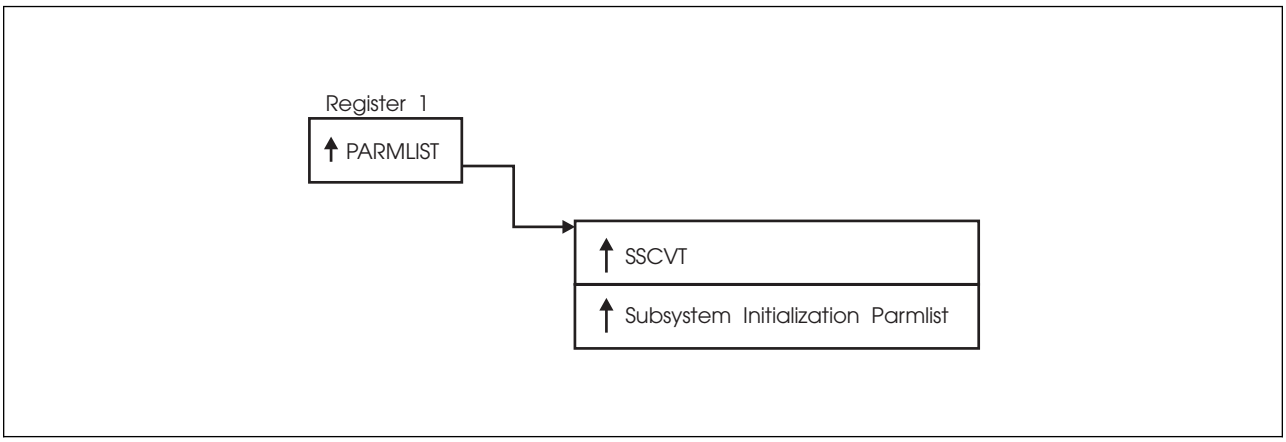

Figure 18. Input to the Initialization Routine

Coding the Initialization Routine

Before coding your initialization routine, consider:

- You can set up a control block structure for your subsystem by building a control block to hold any necessary information and anchoring that control block with the put/get function of the IEFSSI macro. See “Storing Subsystem-specific Information” on page 177 and “Retrieving Subsystem-specific Information” on page 177 for more information on the put/get function of the IEFSSI macro. If, for example, you are planning to use cross memory, your subsystem control block can point to your PC table.
- If you have chosen to have your subsystem run in a separate address space, do not activate the subsystem until the address space is started unless you have made some other provisions for handling requests.
- When you initialize your subsystem with the START command, you must consider whether you want to start your subsystem:
  - Under the job entry subsystem (JES)
  - Under the master subsystem.

  If the operator specifies the SUB=keyword on the START command, the system uses the subsystem that the operator specifies.

  If the operator does not specify the SUB=keyword on the START command, the system defaults to the subsystem that is specified on the REQDSUB parameter of the options function of the IEFSSI macro, or to the MSTR subsystem, if the operator does not specify the REQDSUB parameter of the options function or does not use the options function at all. See “Defining Subsystem Options” on page 177 for more information on the options function of the IEFSSI macro.
- Your initialization routine determines whether the subsystem can respond to the SETSSI command by using the options function of the IEFSSI macro. See *OS/390 MVS System Commands* for more information on the SETSSI command.
- Your initialization routine must be reentrant if it is used by multiple instances of your subsystem, and must reside in a library specified by LNKLST or LPA.
- Your initialization routine must be APF-authorized.

- Your initialization routine is entered in key 0 and supervisor state.
- Your initialization routine can have any addressing mode (AMODE) and any residency mode (RMODE).
- Your initialization routine should issue messages to explain unsuccessful processing using the console information passed in the JSIPL parameter list.
- Your initialization routine should use standard linkage conventions.
- Your initialization routine can define command prefix characters for your subsystem.

  IBM recommends that you use the command prefix facility (CPF) to register your valid command prefix characters. CPF is described in *OS/390 MVS Programming: Authorized Assembler Services Guide*.
- The environment your initialization routine runs in depends upon the way your subsystem is defined. If your subsystem is defined by:
  - The keyword format of the IEFSSNxx parmlib member, your initialization routine runs in the master scheduler address space, under a permanent task.
  - The SETSSI command, your initialization routine runs in the master scheduler address space, under a transient task.
  - The IEFSSI macro, your initialization routine runs in the address space and under the task of the issuer of the IEFSSI macro.

"Example 1 — Subsystem Initialization Routine (TSYSINIT)" on page 261 shows a coding example of a sample initialization routine.

Defining What Your Subsystem Can Do

To define what your subsystem can do, you can use the REQUEST=CREATE parameter of the IEFSSVT macro to build an SSVT for your subsystem.

Note: IEFSSVT macro services are available only to dynamic subsystems. However, other subsystems can use the IEFJSVEC service. See Appendix B, "Using IEFJSVEC with Your Subsystem" on page 279 for more information on IEFJSVEC.

Building the SSVT

The REQUEST=CREATE parameter of the IEFSSVT macro allows you to build an SSVT for your subsystem. The IEFSSVT macro allows users to specify function routines by address rather than requiring the SSI to load the routines. This is useful if the subsystem wants to load its function routines into global storage, but does not want the routines to be deleted if the address space ends. In this case, the subsystem can perform a load-to-address, rather than a standard load, and pass the addresses to the IEFSSVT macro. See *OS/390 MVS Programming: Authorized Assembler Services Reference LLA-SDU* for more information on the LOAD macro.

When preparing to build your subsystem's SSVT, consider:

- When you want to invoke the IEFSSVT macro. You can invoke the IEFSSVT macro either through a subsystem initialization routine or through a subsystem routine invoked during START command processing, as described under "Providing a Routine to Initialize Your Subsystem" on page 162.

- Which common storage subpool your subsystem's SSVT is to be built in. Note that the system uses the mode and key of the caller to access the SSVT and invoke the function routines. Therefore, the storage subpool specified for the SSVT must be a common subpool. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on selecting a common storage subpool.
- What are the maximum number of function routines you expect the subsystem to need. The maximum number of function routines you specify applies to the function routines you define on this build request, and also to any function routines that you define when enabling or disabling functions with the IEFSSVT macro.
- What are the actual number of function routines you want to specify on the current request.
- What is the name or address of each function routine and the function code(s) it supports.
- Where the subsystem function routines are to reside. See "Placement of Function Routines" on page 159 for more information.

Inputs

Before invoking the IEFSSVT macro, the subsystem must use the IEFSSVTI macro to create a table that relates function routines and the function codes they support.

The IEFSSVTI macro can do any one of the following:

- Create a static function routine input table
- Reserve dynamic storage for a function routine input table
- Copy a static table to dynamic storage
- Modify a function routine input table in dynamic storage

A static function routine input table is used when all the information required to build the SSVT is known at compile time.

IEFSSVTI does not attempt to verify that its caller is a dynamic subsystem. IEFSSVTI can be used only in conjunction with IEFSSVT.

Outputs

When control returns to the caller of the IEFSSVT macro create request, the OUTTOKEN parameter contains a token that identifies the SSVT that was created. Use this token when activating or deactivating the subsystem with the IEFSSI macro, or when modifying the SSVT with the enable, disable, or exchange request of the IEFSSVT macro.

A subsystem can have a maximum of two SSVTs created with the create request of the IEFSSVT macro. A create request fails if the maximum number of vector tables already exists.

Changing What Your Subsystem Can Do

To change what your subsystem can do, you can use the IEFSSVT macro to:

- Enable your subsystem for new functions - enable request
- Disable a previously supported function - disable request
- Associate a new function routine with a supported function code - exchange request

The caller of either the enable, disable or exchange request can use the INTOKEN parameter of the IEFSSVT macro to specify a token to identify the subsystem vector table that is to be modified. You can get the INTOKEN parameter by issuing the create request of the IEFSSVT macro. If you do not specify a token, the request applies to the active subsystem vector table (the subsystem vector table currently in use). In this case, the request fails if there is not an active subsystem vector table. You can specify the function routines in the subsystem vector table by name or by address.

Another way to change what your subsystem can do is to use the swap request of the IEFSSI macro. See “Swapping Subsystem Functions” on page 176 for more information.

Enabling Your Subsystem for New Functions

You can use the enable request of the IEFSSVT macro to:

- Dynamically add one or more new function routines, and, for each function routine, one or more function codes that the function routine is to support.

  When preparing to enable additional function routines and function codes, consider:

  – When you will be invoking IEFSSVT.

  – What are the actual number of function routines your subsystem currently supports.

    To dynamically add more function routines to your subsystem, the actual number of function routines your subsystem currently supports must be less than the maximum number of function routines that was specified when your subsystem's SSVT was built.

  – What is the name or entry point address of each additional function routine and the function codes it is to support.

  – Where your subsystem function routines are to reside. See “Setting Up Your Subsystem” on page 157 for more information on where your function routines can reside.

- Dynamically associate one or more function codes with an existing function routine. This function routine might have been specified on the original build SSVT request or might have been added by a previous enable request.

  When preparing to enable additional function codes, consider:

  – When you will invoke IEFSSVT.

  – Which existing function routines will support which additional function codes.

Note: IEFSSVT macro services are available only to dynamic subsystems. However, other subsystems can use the IEFJSVEC service. See Appendix B, “Using IEFJSVEC with Your Subsystem” on page 279 for more information on IEFJSVEC.

Inputs

Before invoking the IEFSSVT macro, the subsystem must use the IEFSSVTI macro to create a table that relates function routines and the function codes they support.

Disabling Previously Supported Functions

You can use the disable request of the IEFSSVT macro to dynamically disable a function code so that your subsystem no longer gets control for that function. Disabling a function is in effect a "logical delete".

Attention: Because there is no serialization on updating the table in the SSVT, other requests for the supported functions might be coming in asynchronously. Therefore, it is important to not remove the function routines from storage.

When preparing to disable one or more function codes, consider:

- When you will be invoking IEFSSVT.
- Which of the existing function codes are no longer supported.

Inputs

Before invoking the IEFSSVT macro, the subsystem must use the IEFSSVTI macro to create a table that relates function routines and the function codes they support.

Unlike the enable request, the disable request does not use the name or address of the function routines in the subsystem vector table when disabling function codes. It uses only the function code itself.

If possible, the SSI reclaims the space in the subsystem vector table occupied by the function routines associated with the disabled function codes. If a function routine does not support any remaining function codes, the SSI makes its subsystem vector table space available for reuse in subsequent enable requests.

Associating a New Function Routine with a Supported Function Code

You can use the exchange request of the IEFSSVT macro to associate the function routine with a supported function code so that the new function routine gets control for that function.

Inputs

Before invoking the IEFSSVT macro, the subsystem must use the IEFSSVTI macro to create a table that relates function routines and the function codes they support.

If possible, the SSI reclaims the space in the subsystem vector table occupied by the function routines associated with the disabled function codes. If a function routine does not support any remaining function codes, the SSI makes its subsystem vector table space available for reuse in subsequent enable requests.

Activating Your Subsystem

To activate your subsystem, you can use:

- The IEFSSVT macro to create an SSVT to define the subsystem's response to the function requests.
- The IEFSSI macro to inform the system that the subsystem is ready to accept function requests.

Using the IEFSSVT macro

Use the create request of the IEFSSVT macro to build the SSVT. See “Building the SSVT” on page 170 for information on building the SSVT.

Using the IEFSSI macro

Use the activate request of the IEFSSI macro to activate your subsystem.

Note: You can use the activate request to activate SSVTs that were built with the create request of the IEFSSVT macro.

The subsystem usually issues the activate request at initialization to activate the subsystem, since the subsystem handles building the vector table. However, the system operator can use also use the SETSSI ACTIVATE command, if the subsystem enabled the SETSSI ACTIVATE command. See *OS/390 MVS System Commands* for more information on the SETSSI ACTIVATE command and “Defining Subsystem Options” on page 177 for more information on using the IEFSSI options service to determine the subsystem's response to the SETSSI command.

Inputs

The activate request provides for the specification of an input token that represents the SSVT to be used to activate the subsystem. This is the token returned to the caller of the create request when the SSVT is built.

The SETSSI ACTIVATE command does not accept a corresponding input, because the system operator cannot manipulate vector tables and does not have access to the tokens.

Considerations

When activating your subsystem, consider:

- The activate request fails if a valid SSVT has not been defined for the subsystem. A valid SSVT is one that has been built as described in “Building the SSVT” on page 170.
- A subsystem can have a maximum of two SSVTs defined to the SSI at any time. Only one of the SSVTs can be active or both SSVTs can be inactive (not currently in use to process requests). An activate request fails if the subsystem is already active.

If more than one vector table exists, the SSI determines which vector table it uses to activate the subsystem as follows:

- If activating the subsystem through the IEFSSI macro, the SSI uses the vector table identified by the vector table token specified with the INTOKEN parameter.
- If activating the subsystem through the SETSSI command or if a vector table token is not specified with the IEFSSI macro, the SSI uses the most recently active vector table.
- If none of the vector tables have ever been active, the SSI uses the last vector table created.
- If the SSI does not manage the vector table, the request fails.

Reactivating a Subsystem after Deactivation

Use the activate request or the SETSSI ACTIVATE command to reactivate a deactivated subsystem. A subsystem can be activated, deactivated and reactivated as many times as is necessary.

Deactivating Your Subsystem

To deactivate your subsystem, you can use either:

- The IEFSSI macro
- The SETSSI DEACTIVATE command.

Use the deactivate request of the IEFSSI macro or the SETSSI DEACTIVATE command to deactivate your subsystem so that your subsystem can suspend operations or stop responding to SSI function requests. The SSI stops routing requests, including broadcast requests, to the subsystem when it receives the deactivation request or command. However, there may be outstanding function requests that have not completed. Since it is not possible to determine when the outstanding requests complete, subsystems must not attempt to delete function routines or other resources that might still be in use after either the deactivate request or SETSSI DEACTIVATE command has been issued.

Note: If a job requires the use of paired subsystem function requests, such as, allocate/unallocate or open/close, the job may not end as expected if the subsystem processing these requests is deactivated when the first request of the pair has been processed but the second has not. The SSI cannot determine if this situation exists. It is both the installation's and the subsystem's responsibility to control the job sequence and subsystem deactivation requests to avoid potential problems.

Outputs

The deactivate request returns a vector table token to its caller in the location identified by the optional OUTTOKEN parameter. The token represents the SSVT that has been deactivated. You can use the token in subsequent activate requests, if the same set of functions is supported when it is reactivated. The vector table token is output only. A deactivate request always applies to the active subsystem vector table.

A deactivate request or command is processed only if the target subsystem is dynamic, even if the active vector table is not managed by the SSI. In this case, the output token contains a zero and the request receives the IEFSSI_WARNING (4) return code.

Note: If the subsystem does not have vector tables managed by the SSI, the subsystem cannot be reactivated dynamically.

Swapping Subsystem Functions

A subsystem can maintain two subsystem vector tables. The two tables can describe different sets of functions to which the subsystem responds or identify different function routines to be invoked for the same function codes.

A subsystem would find it useful to maintain two subsystem vector tables if, for example, a subsystem must quiesce operations. This way, a subsystem can keep one full-function vector table and a second limited-function vector table, and swap so that it can continue to support some minimum set of function while shutting down.

The swap request of the IEFSSI macro allows the subsystem to deactivate the active vector table and activate the inactive table in a single operation. The swap request eliminates the need for separate deactivate and activate requests, which would result in a period of time when the subsystem cannot respond to requests.

Inputs

The swap request allows the user to specify a subsystem vector table token on input. The input token, which is named with the INTOKEN parameter, identifies the vector table that is to be activated (with the activate request or command). If INTOKEN is not specified, the inactive (previously created) vector table is activated.

Outputs

The swap request allows the user to specify a subsystem vector table token on output. On completion of the swap, the output token, which is named with the OUTTOKEN parameter, identifies the outgoing (previously active) vector table.

If the subsystem is initially inactive, the swap request receives the IEFSSI_WARNING (4) return code and is treated as an activate request. The output token identified with the OUTTOKEN parameter contains a zero. If the outgoing (initially active) vector table is not managed by the SSI, the output token contains a zero and the request receives the IEFSSI_WARNING return code.

Storing and Retrieving Subsystem-specific Information

To store and retrieve subsystem-specific information, you can use the IEFSSI macro. A subsystem or a subsystem initialization routine needs to be able to pass information to the subsystem's function routines. If the subsystem code and its function routines are in separate load modules or run in separate address spaces, there may be no direct way for the subsystem to communicate with its function routines. The store and retrieve services provide a way for subsystems to store and retrieve subsystem-specific information and pass that information between subsystem components.

Storing Subsystem-specific Information

Use the put request of the IEFSSI macro to store subsystem-specific information. The put service allows a subsystem to store a total of 8-bytes of subsystem-specific information in two non-contiguous 4-byte fields, which are identified by the SUBDATA1 and SUBDATA2 parameters. The user can store the data in either or both of the two fields on a single invocation of the put service.

A typical use of the put service is to store a pointer to a subsystem-specific control block, which the subsystem initialization routine created and made available for use by the subsystem function routines.

IBM recommends that your subsystem create and anchor control blocks to store subsystem data, even if the stored data is small enough to fit within the two fields provided. This lets your subsystem store more information at a later time. In addition, the information stored using this service does not reside in fetch-protected storage. However, the subsystem can create its control block in a fetch-protected subpool.

Retrieving Subsystem-specific Information

Use the get request of the IEFSSI macro to retrieve subsystem-specific information. The get service allows a subsystem to retrieve subsystem-specific information that was stored using the put request. The retrieved information, which is identified by the SUBDATA1 and SUBDATA2 parameters, is the information that was originally identified by the corresponding put service parameter.

Defining Subsystem Options

To define subsystem options, you can use the IEFSSI macro. The options request allows a subsystem to specify:

- Whether it responds to the SETSSI command
- The subsystem (MSTR or primary) under which the subsystem is to be started.

Use

You can invoke the options request more than once for a single subsystem. The most recent invocation of the service determines the characteristics of the subsystem. The first time the service is invoked, the defaults described in the IEFSSI macro are effective for parameters that are not specified. See *OS/390 MVS Programming: Authorized Assembler Services Reference ENF-IXG* for more information on the IEFSSI macro. For subsequent invocations, characteristics corresponding to omitted parameters retain their most recent value. For example, if the first invocation does not specify the COMMAND parameter, the default of COMMAND=NO is used. However, if the first invocation specifies COMMAND=YES and a second invocation does not specify the COMMAND parameter, the subsystem continues to respond to the SETSSI command as specified by the first invocation.

Responding to the SETSSI Command

The system does not process the SETSSI command directed to subsystems that have not explicitly authorized the commands, because existing subsystem were not designed for the possibility of dynamic manipulation by commands. The system may be disrupted if these subsystems are manipulated unexpectedly by commands.

Starting Your Subsystem Under the Primary Subsystem

A subsystem may require the services of the primary subsystem when being started. For example, it may require the primary subsystem to provide the use of subsystem data sets or an internal reader. The options service specifies whether the subsystem being added requires the primary subsystem, and is intended for use in a subsystem initialization routine.

If the START command does not specify the subsystem under which the target subsystem should start, the system uses the information specified with the REQDSUB parameter of the options request.

Querying Subsystem Information

To query subsystem information, an application can use the IEFSSI macro or an operator can use the DISPLAY SSI command. The query request allows either an application or the operator to query the following information for all subsystems defined to the SSI:

- The subsystem name
- If the subsystem is dynamic or not dynamic
- If the subsystem is the primary subsystem
- If the subsystem is active or inactive
- If the subsystem is dynamic, whether it accepts or rejects dynamic SSI commands
- If the subsystem is active, which function codes it supports.

An application can also query the following additional information:

- The number of vector tables associated with the subsystem, with a maximum of two vector tables.
- The following information for each associated vector table:
  - If the vector table is managed by the SSI. A vector table managed by the SSI is a vector table created with the IEFSSVT REQUEST=CREATE macro.
  - A locator. This locator is a token if the vector table is managed by the SSI and is an address if the vector table is not managed by the SSI.
  - If the vector table is active.
  - The function codes supported by the vector table.

This information represents a snapshot of the subsystems defined to the SSI when you process the query request.

To obtain information about the primary subsystem without knowing its name, use the query request and specify a subsystem name of !PRI.

Using the Subsystem Query Request of the IEFSSI Macro

The query request of the IEFSSI macro is the only service provided by this macro that does not require the caller to be authorized.

Inputs

The SSI obtains the storage necessary to return the query request information, because the issuer of the query request cannot determine in advance how much information will be returned. The issuer of the query request can use the WORKASP parameter to specify the subpool in which the SSI can obtain the storage. The query request fails if the SSI is unable to obtain enough storage. Unauthorized callers are limited to unauthorized subpools.

The query request returns information either for a single subsystem or for all subsystems matching the pattern specified with the SUBNAME parameter. The pattern can contain the following wildcard characters:

- An asterisk ('*') — matches zero or more characters
- A question mark ('?') — matches one character.

Outputs

The mapping macro IEFJSQRY maps the output returned by the query request.

If the SSI obtains the storage it needs to use the query request, the SSI returns the address of the output work area in the variable that the WORKAREA parameter identifies. The JQRYLEN field mapped by the IEFJSQRY macro contains the length of the returned storage. Upon completion, the issuer of the IEFJSQRY macro must free the returned storage. You should have established a recovery routine to free the returned storage in case your program ends abnormally. IBM recommends you use task-oriented or job-oriented storage to ensure that the storage is released upon task or job completion.

If you request information about multiple subsystems, the output lists the information in broadcast order. That is, the subsystems are listed in the same order in which SSI broadcast processing invokes them. For each subsystem, the IEFSSI query request returns information about all associated vector tables managed by the SSI, active or not. For vector tables that are not managed by the SSI, the system locates only the active vector table and returns information about that vector table only.

A query request may fail to return information about some subsystems. If a subsystem is defined after IPL by directly manipulating the SSI control blocks and the definition either occurs during the processing of the query request or is not correctly completed, some subsystems may not be represented in the response to the query request.

Using the Display SSI Command

The DISPLAY SSI command displays status information about all subsystems defined to the SSI. You can request information for all subsystems at once or for those subsystems which meet the criteria specified by the filters used when issuing the DISPLAY SSI command. You can use filters to limit the information displayed to:

- One particular subsystem or those subsystems whose names match a specified pattern
- Subsystems that are either dynamic or not dynamic
- Subsystems that are either active or not active
- Subsystems that respond to a given list of function codes.

In addition, the issuer of the command can use the LIST or ALL keywords to specify whether to display subsystem function codes. Subsystem information is displayed in broadcast order.

Maintaining Information About the Callers of Your Subsystem

A common requirement for a subsystem is to maintain information specific to each of its callers. To accomplish this, a subsystem needs both:

- A method of uniquely identifying each caller.
- A work area to store information about each caller (or a place to store the address of a work area).

The subsystem affinity service solves both of these requirements. It allows a subsystem to store and retrieve data at the task control block (TCB) level, thus removing its dependence on information passed by callers.

Consider the following example: A subsystem provides service to many callers, and must also maintain use counts by caller. Each caller can be identified by the TCB that is associated with it.

The subsystem uses the subsystem affinity service to maintain a separate use count for each of its callers. For each caller, the subsystem affinity service provides the subsystem with a unique fullword entry, called a ***subsystem affinity entry***.

Figure 19 shows how the subsystem uses a subsystem affinity entry for a particular caller, to hold a pointer to a work area. The subsystem records use counts in the work area. Because the subsystem affinity service allows each caller to be uniquely identified by the TCB that it runs under, the subsystem can track the use count for each of its callers.

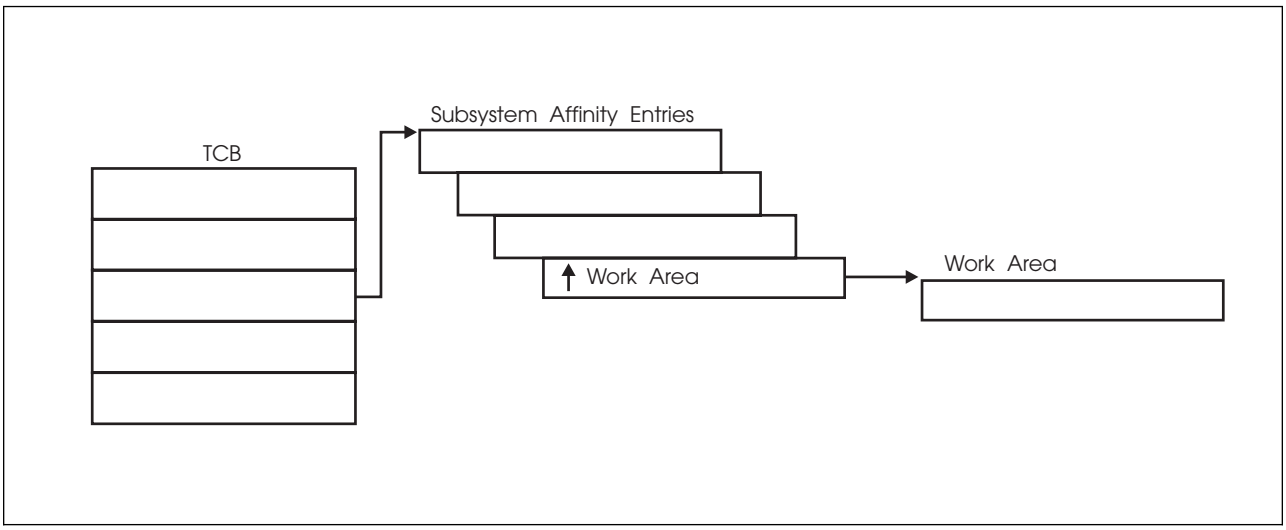

Figure 19. Subsystem Affinity Service

Accessing the Subsystem Affinity Entry: To access the subsystem affinity entry for each of its callers, a subsystem needs to:

- Invoke the verify subsystem function (SSI function code 15) through the IEFSSREQ macro to acquire its ***subsystem affinity index***. See "Verify Subsystem Function Call — SSI Function Code 15" on page 44 for information on SSI function code 15.
- Issue the SSAFF SET request to store data in the entry.

On subsequent invocations, the subsystem can issue the SSAFF OBTAIN request to retrieve the address of a work area from the entry.

SSAFF: Set/Obtain Subsystem Affinity

Use the SSAFF macro to SET or OBTAIN a subsystem affinity entry.

An SSAFF SET request places one fullword of subsystem passed data in the subsystem affinity entry, which is identified by the TCB parameter and the subsystem affinity index. This allows the subsystem to put its entry in the subsystem affinity entry of the current, active TCB.

An SSAFF OBTAIN request extracts and returns to the subsystem the fullword of data from the subsystem affinity entry identified by the current TCB and the subsystem's index value. The OBTAIN request works only for the subsystem affinity entry pointed to by the current TCB.

Note: A subsystem that uses the TCB subsystem affinity service cannot rely on information stored in a subsystem affinity entry on a checkpoint/restart: the subsystem affinity index value could change from one system initialization to another. For additional information about the restrictions and use of the checkpoint/restart facility, see *OS/390 DFSMS Checkpoint/Restart*.

Before you issue the SSAFF macro, register 13 must point to an 18-word save area.

The syntax of the SSAFF macro is:

[symbol]	SSAFF	{SET [,TCB=tcb-address]} {OBTAIN } ,DATA=data-address ,ENTRY=index-value

One blank is required before and after “SSAFF”.

SET requests have the following requirements:

- The caller must be enabled, unlocked, and in supervisor state, key 0.
- The caller must not be in cross-memory mode.
- The TCB must be in the caller's home address space and must be either the current TCB or a subtask of the current TCB. If any of these conditions are not satisfied, the calling routine abends.

OBTAIN requests have the following requirements:

- The caller must be in task mode. If this condition is not met, the calling routine abends.
- The caller must have current addressability to the home address space.

The SSAFF macro parameters have the following meanings:

symbol
any valid assembler language symbol.

SET
indicates that MVS is to place the value specified by the DATA parameter into the subsystem's associated subsystem affinity entry. The SET request destroys the contents of registers 14, 15, 0, 1, and 2.

OBTAIN
indicates that MVS is to place the contents of the specified subsystem affinity entry of the issuing task in the register or data area specified by the DATA parameter. The OBTAIN request destroys the contents of registers 14, 15, 0, and 1.

,TCB=tcb-address — RX-Type Address, or Register (2)-(12)
this parameter, valid only for SET requests, specifies the register or storage location that contains the address of the TCB whose subsystem affinity entry MVS is to use when processing the SET request.

Note: If you omit the TCB parameter, MVS uses the current task's TCB. If you allow this default, the calling program must include the IHAPSA mapping macro to identify the current TCB.

,DATA=data-address — RX-Type Address, or Register (1) or (3)-(12)
For SET, this parameter specifies the register or fullword storage location that contains the subsystem's data. MVS stores the data in the subsystem affinity entry for a SET request.

For OBTAIN, this parameter specifies the register or fullword storage location that is to contain the value extracted from the subsystem affinity entry.

MVS returns a value of zero if any one of the following is true during an OBTAIN request:

- The subsystem affinity entry associated with the specified index-value contains a zero.
- A null subsystem affinity entry exists for the caller. (A SET request was not performed prior to the OBTAIN request.)
- The specified index value exceeds the size of the caller's subsystem affinity entry.

,ENTRY=index-value — RX-Type Address, or Register (0) or (3)-(12)
this parameter specifies the register or fullword storage location that contains the subsystems affinity index value. If you specify an index value greater than the number of subsystems currently defined to MVS, the request fails.

For SSAFF SET requests, the subsystem affinity service uses:

- The TCB address to locate the required subsystem affinity table. When the subsystem does not supply the TCB address, MVS uses the currently-executing TCB (PSATOLD).
- The subsystem affinity index value to locate the specific subsystem affinity entry that is to be set.

For SSAFF OBTAIN requests, the subsystem affinity service uses:

- The currently-executing TCB to locate the required subsystem affinity table.
- The subsystem affinity index value to locate the specific subsystem affinity entry to be returned.

SSI Function Codes Your Subsystem Can Support

This chapter contains detailed information on function codes your subsystem can support. The following is a list of SSI function codes, along with their purpose and the type of subsystem request.

Function Code	Requested Function	Type of Request
4	End-of-task	Broadcast
8	End-of-address space (End-of-memory)	Broadcast
9	WTO/WTOR	Broadcast
10	Command processing	Broadcast
14	Delete operator message	Broadcast
50	Early notification of end-of-task	Broadcast
54	Request subsystem version information	Directed
58	SMF SUBPARM option change	Directed
78	Tape device selection	Broadcast

Your subsystem can define and use its own function codes, using the range 236 to 255.

SSI Function Code Descriptions

Your subsystem can support several SSI function codes when coding for an MVS/SP-JES2/JES3 environment. This section contains detailed descriptions of the SSI function codes listed at the beginning of this chapter.

See Appendix A, Examples — Subsystem Interface Routines for coding examples of function routines.

Product-sensitive programming interface

End-of-Task Call — SSI Function Code 4

The End-of-Task call (SSI function code 4) provides the ability to do task-related resource clean up. Whenever a task ends, all active subsystems that are enabled to receive SSI function code 4 are given control from the SSI after resource managers are given control, including resource managers which were dynamically defined. Each subsystem function routine will get control for every task that ends.

Note: This broadcast request is issued after all dynamic resource managers have been given control, but not all system resource managers. For instance, the following resource managers receive control after this End-of-Task call:

- PC Auth
- RSM

Type of Request

Broadcast SSI call.

Use Information

Your subsystem can use the SSI function code 4 to clean up any resources for a task that is associated with a particular subsystem, and free any resources not normally handled by a resource manager.

Because your function routine gets control for every End-of-Task call, using your own subsystem may not be the most efficient way to do your own clean up for ending tasks. IBM recommends that you define your own resource manager through the use of the RESMGR macro. RESMGR can be used to monitor specific ending tasks, rather than having to check each ending task or address space to see if it used the subsystem. For a general description of resource managers and how they can be defined at both IPL time and dynamically, see *OS/390 MVS Programming: Authorized Assembler Services Guide*.

Issued to

- All active subsystems that indicate they support the End-of-Task function when the system (MVS) issues the End-of-Task call.

Related SSI Codes

SSI function code 4 is similar to SSI function code 50 (Early End-of-Task call). The only difference is that, for SSI function code 4, your routine is given control after most resource managers are given control. For SSI function code 50, your routine is given control before most resource managers are given control. If you want to obtain control before most resource managers have been invoked, see SSI function code 50 (Early End-of Task).

Related Concepts

None.

Environment

Review “Function Routines/Function Codes” on page 157, which describes both the general environment on entry to your function routine and other programming considerations that your function routine should take into account.

If you decide to set up your subsystem to handle End-of-Task calls, make sure that your function routine is in place before you enable the subsystem to receive SSI function code 4. IBM recommends that you use the IEFSSVT macro to notify MVS that your subsystem should be given control whenever End-of-Task calls are made. IEFSSVT macro services are available only to dynamic subsystems. Subsystems that are not dynamic can still use the IEFJSVEC service; see “Building the SSVT” on page 279 and “Enabling Your Subsystem for New Functions” on page 284 for more information.

The subsystem function routine runs in the address space of the ending task. Because each subsystem function routine is called for every ending task, the subsystem function routine should not be a long running program. That is, the function routine should quickly determine if the subsystem was ever associated with the ending task and, if not, return to the system. Also, do not code a function routine that enters an explicit WAIT or uses a system service that enters a WAIT. Entering a WAIT can cause degraded system performance.

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your function routine:

- IEFSSOBH
- IEFJSSIB
- IEFSSET

The function routine receives control in the following environment:

Minimum authorization	Supervisor state with PSW key 0
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control parameters	The SSIB, SSOB, and SSET control blocks reside in storage below 16 megabytes.
Recovery	The function routine should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on how to set up an ESTAE-type recovery environment.

Figure 20 on page 188 shows the environment on entry to the function routine for SSI function code 4.

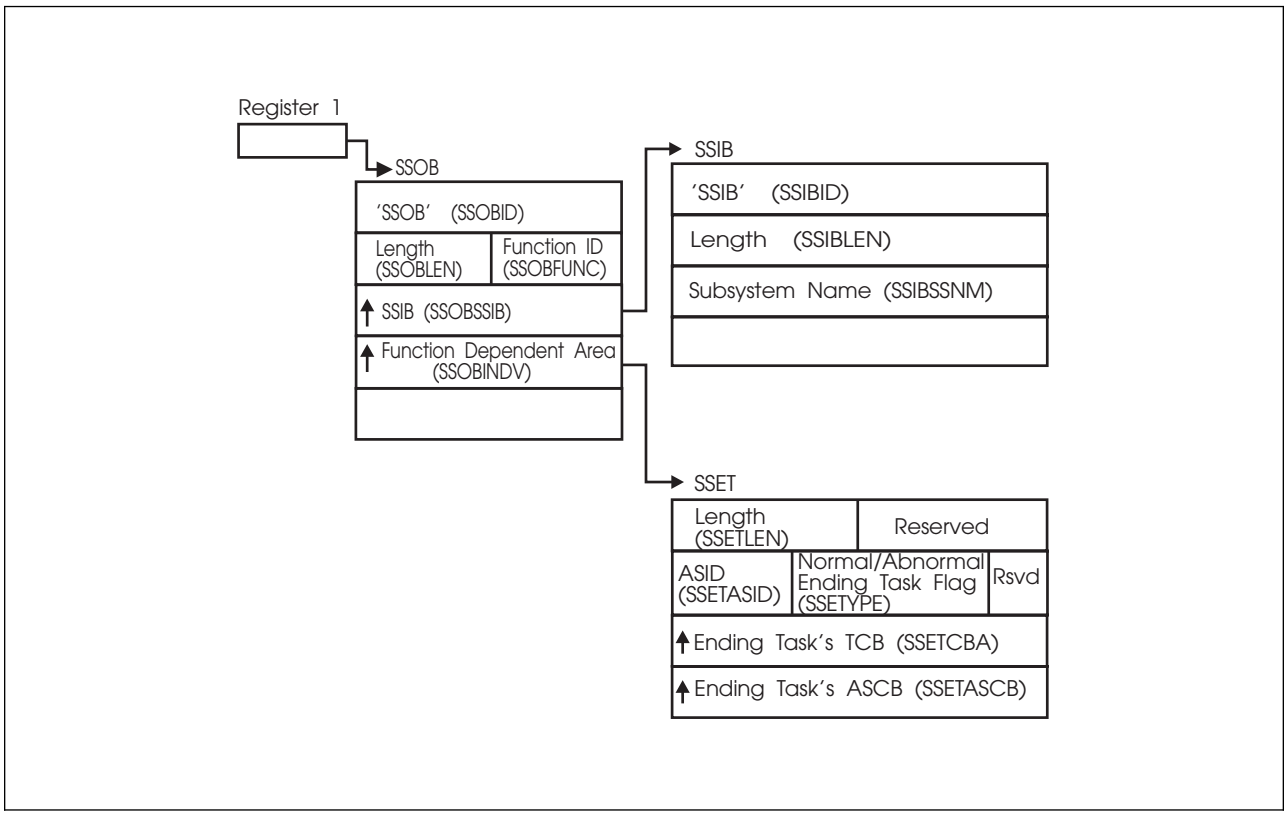

Figure 20. Environment on Entry to the Function Routine for SSI Function Code 4

Input Register Information

On entry to the function routine the general purpose registers contain:

Register	Contents
0	Address of the subsystem's SSCVT
1	Address of the SSOB control block
13	Address of a standard 18-word save area
14	Return address
15	Entry point address

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSET

SSOB Contents: MVS sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function code 4 (SSOBEOT)

SSOBSSIB	Address of the SSIB control block
SSOBRETN	Return code from previous subsystem function routine or zero. Because broadcast requests are routed to all active subsystems, the SSOBRETN field contains the return code value set by some previously invoked subsystem or zero. See “Output Register Information” for a list of possible SSOBRETN return codes.
SSOBINDV	Address of the function dependent area (SSET control block)

SSIB Contents: MVS sets the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of the subsystem enabled to receive this function code.

SSET Contents: MVS sets the following fields in the SSET control block on input:

Field Name	Description
SSETLEN	Length of the SSET (SSETSIZE) control block
SSETASID	ASID of the address space in which the task was active
SSETFLAG	Flag indicators • **SSETYPE ON** — indicates an abnormal ending task • **SSETYPE OFF** — indicates a normal ending task
SSETCBA	Address of ending task's TCB
SSETASCB	Address of ending task's ASCB

Output Register Information

Upon exit from the function routine, the general purpose registers must contain:

Register	Contents
0 — 12	Restored to contents on entry
14	Return address
15	Return code

Return Code Information

For MVS to process broadcast functions properly, you must use the following return code conventions for function routines that handle broadcast calls. When a routine returns control to the SSI:

- Set register 15 to 0.
- Set the SSOBRETN field in the SSOB control block to one of the following:

  Return Code

(Decimal)	Meaning
0	The function routine recognized the request but did not process it.
4	The function routine recognized the request and processed it.

End of Product-sensitive programming interface

Product-sensitive programming interface

End-of-Address Space (End-of-Memory) Call — SSI Function Code 8

The End-of-Address Space Function (End of Memory) call (SSI function code 8) provides the ability to free up any system-level resources, such as CSA, obtained by a subsystem on behalf of an address space. Whenever an address space ends, all active subsystems that are enabled to receive SSI function code 8 are given control from the SSI. The function routine gets control for every address space that ends.

Type of Request

Broadcast SSI call.

Use Information

Your subsystem can use SSI function code 8 to clean up any system-level resources which your subsystem obtained for one or more address spaces. Because private storage for the address space has already been deleted, your function routine must not reference any storage in the ending address space.

Because your function routine gets control for every address space that ends, using your own subsystem may not be the most efficient way to do your own clean up for ending address spaces. IBM recommends that you define your own resource manager through the use of the RESMGR macro. You can use RESMGR to receive control for specific ending address spaces, rather than having to check each ending task or address space to see if it used the subsystem. For a general description of resource managers and how they can be defined at both IPL time and dynamically, see *OS/390 MVS Programming: Authorized Assembler Services Guide*.

Issued to

- All active subsystems that indicate they support the End-of-Address space function when the system (MVS) issues the End-of-Address space call.

Related SSI Codes

None.

Related Concepts

None.

Environment

Review “Function Routines/Function Codes” on page 157, which describes both the general environment on entry to your function routine and other programming considerations that your function routine can take into account.

If you decide to set up your subsystem to handle End-of-Address space calls, make sure that your function routine is in place before you enable the subsystem to receive SSI function code 8. IBM recommends that you use the IEFSSVT macro to notify MVS that your subsystem should be given control whenever End-of-Address space calls are made. IEFSSVT macro services are available only to dynamic subsystems. Subsystems that are not dynamic can still use the IEFJSVEC service;

see “Building the SSVT” on page 279 and “Enabling Your Subsystem for New Functions” on page 284 for more information.

The subsystem function routine runs in the master scheduler address space. Because each subsystem function routine is called for every ending address space, the subsystem function routine should not be a long running program. That is, the function routine should quickly determine if the subsystem was ever associated with the ending address space and, if not, return to the system. Also, do not code a function routine that enters an explicit WAIT or uses a system service that enters a WAIT. Entering a WAIT can cause degraded system performance.

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your function routine:

- IEFSSOBH
- IEFJSSIB
- IEFSSEN

The subsystem function routine receives control in the following environment:

Minimum authorization	Supervisor state with PSW key 0
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control parameters	The SSOB, SSIB, and SSEN control blocks reside in storage below 16 megabytes.
Recovery	The function routine should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on an ESTAE-type recovery environment.

Figure 21 on page 193 shows the environment on entry to the function routine for SSI function code 8.

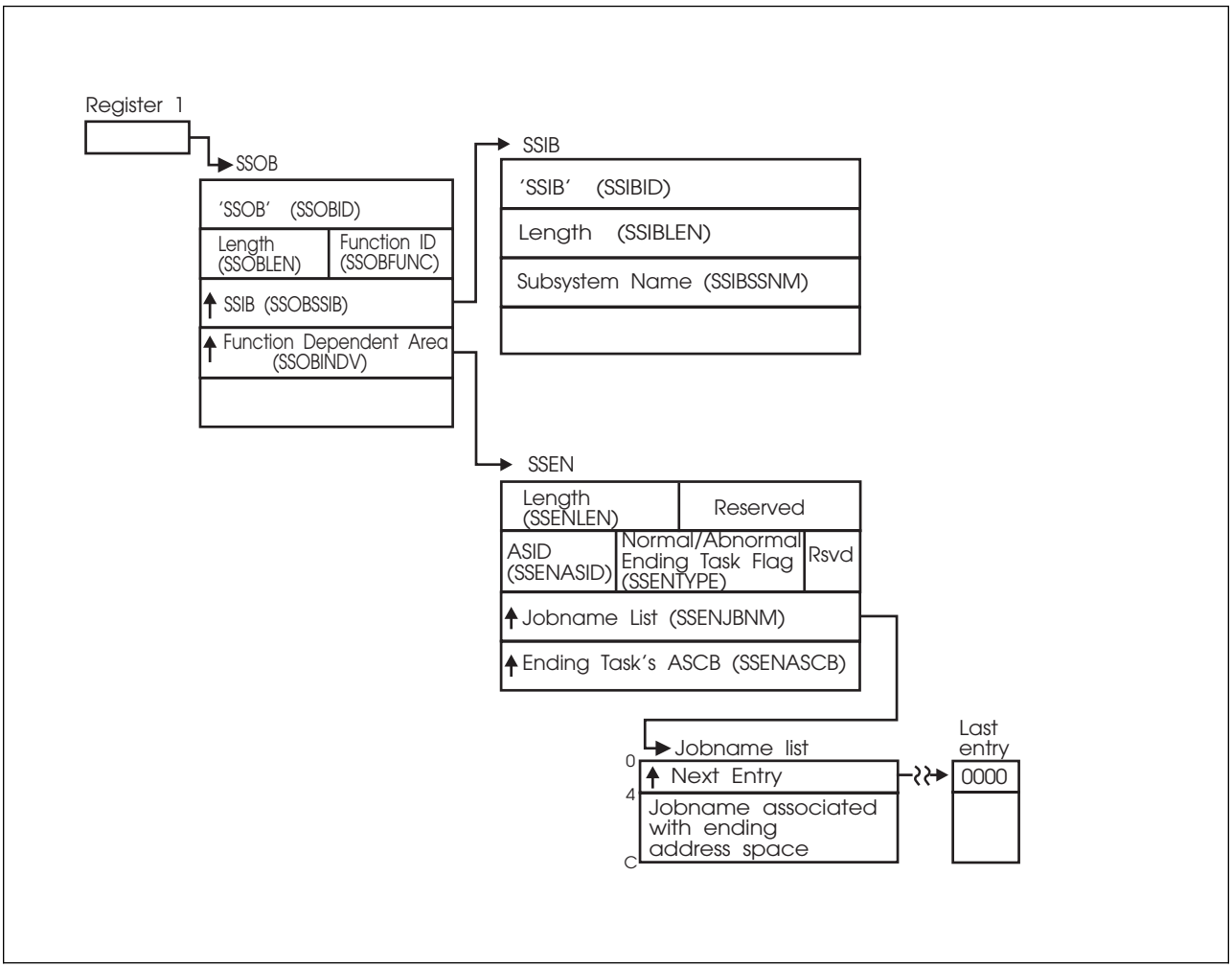

Figure 21. Environment on Entry to the Function Routine for SSI Function Code 8

Input Register Information

On entry to the function routine the general purpose registers contain:

Register	Contents
0	Address of the subsystem's SSCVT
1	Address of the SSOB control block
13	Address of a standard 18-word save area
14	Return address
15	Entry point address

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSEN

SSOB Contents: MVS sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function code 8 (SSOBEOM)
SSOBSSIB	Address of SSIB control block
SSOBINDV	Address of function dependent area (SSET control block)

SSIB Contents: MVS sets the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of subsystem that is enabled to receive this function code.

SSEN Contents: MVS sets the following fields in the SSEN control block on input:

Field Name	Description
SSENLEN	Length of SSEN (SSENSIZE) control block
SSENASID	ASID of ending address space
SSENFLAG	Flag indicators • **SSENTYPE ON** — indicates an abnormal ending address space • **SSENTYPE OFF** — indicates a normal ending address space
SSENJBNM	Job name list pointer. For both normal and abnormal endings, contains the list of job names that represents work associated with the address space that is ending. Each entry in the list consists of 12 bytes (first 4 bytes contains pointer to next job name block or zero if last; remaining 8 bytes contains the job name).
SSENASCB	Address of ending address space's ASCB

Output Register Information

Upon exit from the function routine, the general purpose registers must contain:

Register	Contents
0 — 12	Restored to contents on entry
14	Return address
15	Return code

Return Code Information

For MVS to process broadcast functions properly, you must use the following return code conventions for function routines that handle broadcast calls. When a routine returns control to the SSI:

- Set register 15 to 0.
- Set the SSOBRETN field in the SSOB control block to one of the following:

Return Code (Decimal)	Meaning
0	The function routine recognized the request but did not process it.
4	The function routine recognized the request and processed it.

End of Product-sensitive programming interface

Product-sensitive programming interface

WTO/WTOR Call — SSI Function Code 9

All applications running on MVS, MVS subsystems, and MVS itself, generate messages. Each time a message is generated (with a write-to-operator (WTO) or a write-to-operator-with-reply (WTOR) macro), the WTO/WTOR call (SSI function code 9) is issued.

Note that WTOs and WTORs are issued in one of the following forms:

- Single-line message (WTO)
- Multi-line message (WTO) — first line of message
- Multi-line message (WTO) — subsequent lines of message
- Single-line message with reply (WTOR).

Type of Request

Broadcast SSI call.

Use Information

To have your function routine receive control for SSI function code 9, you must use the IEAVG700 interface. You only need to issue IEAVG700 once for each IPL. Use the following coding fragment to call module IEAVG700:

```
⋮
*     Register declarations
R1       EQU  1                  Declaration for register 1
R13      EQU  13                 Declaration for register 13
R15      EQU  15                 Declaration for register 15

⋮
DOBRDCST EQU  *                  Request broadcast of WTO/WTORs
         LA   R1,SCSRPLST        Get addressability to SCSR
         ST   R1,SCSRPTR         Save pointer for standard linkage
         XC   SCSRPLST(SCSPLEN),SCSRPLST  Zero out parameter list
         MVC  SCSACRO,SCSRACRN   Set acronym value
         MVI  SCSVER,SCSVERSN    Set version level
         OI   SCSFUNC1,SCSBRDON  Indicate to broadcast WTO/WTORs
         LA   R1,SCSRPTR         Set up standard entry linkage
         LA   R13,SAVEAREA       Set up standard save area
         LINK EP=IEAVG700        Call subsystem console routine
         LTR  R15,R15            See if request was successful
         BNZ  BRDFAIL            Branch to process unsuccessful call
* Processing continues here for successful call
⋮
*        Module static storage area
SCSRACRN DC   CL4'SCSR'

⋮
*        Module dynamic storage area
SCSRPTR  DS   A                   Pointer to SCSR
SAVEAREA DS   18F                 Standard save area

⋮
*        Include mapping for Subsystem Console Service Routine
         IEZVG100                 Include SCSR mapping macro
```

The SCSR (subsystem console service routine) parameter list is mapped by mapping macro IEZVG100. Module IEAGV700 must be invoked in key 0, supervisor state, running enabled in task mode with no locks held.

Upon ending, your subsystem should request that broadcasting be discontinued. Use the same type coding fragment as above, except that the **SCSBRDOF** bit (Broadcast off) is set, instead of the **SCSBRDON** bit (Broadcast on).

Your installation might also use the WTO/WTOR call (SSI function code 9) to take any of the following actions against a message:

- Alteration — including text and routing information
- Deletion
- Generation of a reply (in the case of WTOR)
- Suppression.

Your installation can use the following methods to affect WTO/WTOR message processing:

- Message processing facility (MPF) — see *OS/390 MVS Planning: Operations*.
- Installation-written exit routines — see *OS/390 MVS Installation Exits*.
- Automation — see *OS/390 MVS Planning: Operations*.

In choosing which method to use to affect WTO/WTOR message processing, take the following into consideration:

- The WTO general exit (IEAVMXIT) or message processing facility (MPF) exits are the recommended ways to take actions against MVS messages prior to their distribution to consoles and the system log, because they get control before the SSI gets control, and they can be changed easily through the SYS1.PARMLIB member. See *OS/390 MVS Installation Exits* for information about IEAVMXIT and MPF exits.
- The primary subsystem (JES) is usually the first subsystem to get control from the SSI.
- Automation subsystems (such as NetView) are common users of SSI function code 9. Automation subsystems also get control from the SSI so that, depending on what you want your program to do, placing your subsystem before or after an automation product may be of concern. For example, subsystems may alter messages. If you are using an automation product that gets its messages from the SSI, it may not receive the final version of a message if there are other subsystems that subsequently change the message. If so, make sure you code the subsystems in SYS1.PARMLIB member IEFSSNxx in the order in which you want the subsystems to get control.

IBM recommends that you affect message processing with MPF or through one of the automation subsystems.

MCSOPER/MCSOPMSG Macro Services: While SSI function code 9 is useful for an application that needs to trap messages from the MVS message stream, it is no longer the recommended interface for that purpose. The MCSOPER/MCSOPMSG macro services (also known as Extended Operator) are the recommended programming interface for receiving MVS messages. See *OS/390 MVS Programming: Authorized Assembler Services Reference LLA-SDU* for further information about these services.

Issued to

- All active subsystems that indicate they support the WTO/WTOR function when the system (MVS) issues the WTO/WTOR call.

Related SSI Codes

None.

Related Concepts

You need to know how to use WTO and WTOR macros and the IEAVG700 interface. You also need to understand the role that routing information (routing codes) plays in determining the destinations of a message. See *OS/390 MVS Programming: Authorized Assembler Services Reference SET-WTO* and *OS/390 MVS Routing and Descriptor Codes* for more information.

Environment

Review “Function Routines/Function Codes” on page 157, which describes both the general environment on entry to your function routine and other programming considerations that your function routine should take into account.

If you decide to set up your subsystem to handle WTO/WTOR calls, make sure that your function routine is in place before you enable the subsystem to handle SSI

function code 9. IBM recommends that you use the IEFSSVT macro to notify MVS that your subsystem should be given control whenever WTO/WTOR calls are made. IEFSSVT macro services are available only to dynamic subsystems. Subsystems that are not dynamic can still use the IEFJSVEC service; see “Building the SSVT” on page 279 and “Enabling Your Subsystem for New Functions” on page 284 for more information.

WTOs occur frequently on MVS. Function routines should therefore be as efficient as possible. Function routines should never enter a WAIT and should never use system services that have implied WAITs (such as I/O). Entering a WAIT can cause degraded system performance.

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your function routine:

- IEFSSOBH
- IEFJSSIB
- IEFSSWT
- IHAWQE
- IHAORE

The write-to-operator WTO queue element (WQE), mapped by IHAWQE, represents a message.

The operator reply element (ORE), mapped by IHAORE, represents a WTOR.

The function routine receives control in the following environment:

Minimum authorization	Supervisor state with PSW key 0
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control parameters	The SSOB, SSIB, SSWT, and WQE control blocks reside in storage below 16 megabytes. The ORE control block resides above 16 megabytes.
Recovery	The function routine should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Reference ENF-IXG* for more information on these macros. Failure to establish a recovery environment causes the current message to be deleted from the system, if the function routine ends abnormally while processing the message. The function routine's recovery should specify a retry point (address) and return 4 on the SETRP macro before returning to system. The retry point should be used to complete a normal return to the function routine's caller. When the function routine returns to its caller under these circumstances, it should indicate to the system to take no action against the message by setting both register 15 and the SSOBRETN to zero. See “Input Register Information” on page 200 for more information about specifying to the system the action that should be taken by your function routine.

Input Register Information

On entry to the function routine the general purpose registers contain:

Register	Contents
0	Address of the SSCVT
1	Address of the SSOB control block
13	Address of a standard 18-word save area
14	Return address
15	Entry point address

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSWT
- WQE
- ORE

SSOB Contents: MVS sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB (SSIBHSIZ) control block
SSOBFUNC	SSI function code 9 (SSOBWTO)
SSOBSSIB	Address of the SSIB control block
SSOBRETN	Return code from previous function routine (when SSI function code 9 is operating in broadcast mode).
SSOBINDV	Address of the function dependent area (SSWT control block)

SSIB Contents: MVS sets the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of the subsystem which is enabled to receive this function code.

SSWT Contents: MVS sets the following fields on input when either a single-line WTO, multi-line WTO, or WTOR is being passed on the SSI call.

SSWT Contents for a Single-line WTO: MVS sets the following fields in the SSWT control block on input for a single-line WTO:

Field Name	Description
SSWTLEN	Length of the SSWT (SSWTSIZE) control block
SSWTWQE	Address of the WQE control block
SSWTNMOD	Value of SUBSMOD keyword on the WTO macro

SSWTPRSP	Indicates whether the SSWTPRTY field is valid
SSWTPRTY	Value of the PRTY keyword on the WTO macro
SSWTSNSP	Indicates whether the JOBNAME keyword was specified on the WTO
SSWTSISP	Indicates whether the JOBID keyword was specified on the WTO

Figure 22 shows the environment for a single-line WTO in the SSWT control block.

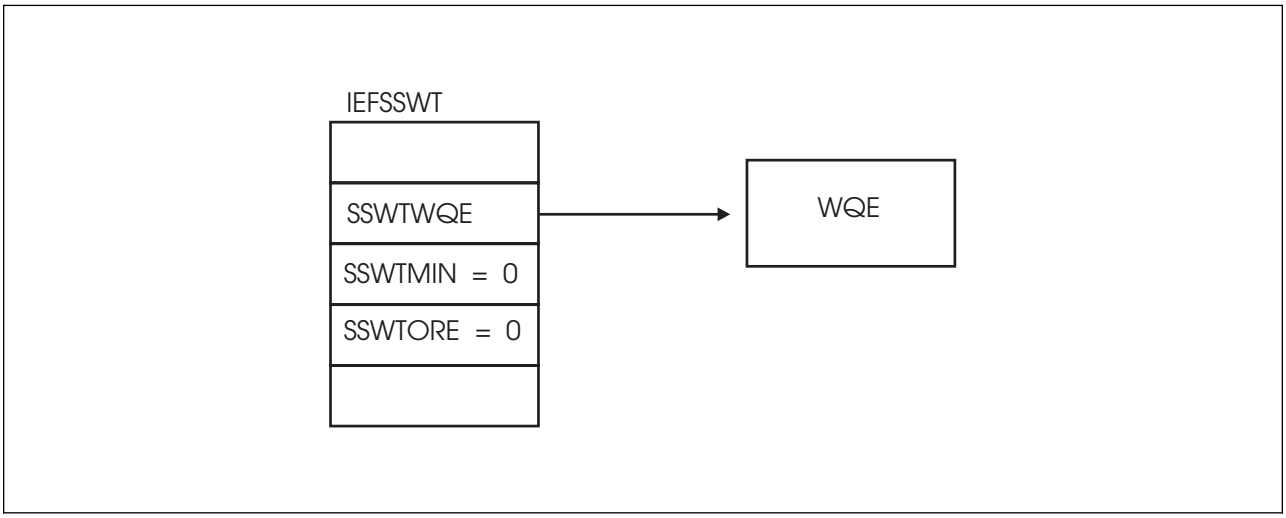

Figure 22. Environment for a Single-line WTO in the SSWT Control Block

WQE Contents for a Single-line WTO: MVS sets the following fields in the WQE control block on input for a single-line WTO:

Field Name	Description
WQETXTLN	Length of the message text
WQETS	EBCDIC time stamp
WQEJOBNM	Jobname (inserted by the primary subsystem)
WQETXT	Message text
WQEXA	Indicators • **WQEWTOR** — indicates the message is a WTOR • **WQEAUTH** — indicates the message is issued by an authorized program.
WQEASID	ASID of the message issuer
WQETCB	TCB address of the message issuer
WQESEQ#	Message DOM id
WQEMCSF1	Indicators • **WQEMCSA** — indicates the WQEROUT and WQEDESCD fields are valid • **WQEMCSB** — indicates the WQECNID and WQECNNME fields are valid • **WQEMCSC** — indicates the message is a command response • **WQEMCSD** — indicates the WQEMSGTP field is valid

- **WQEMCSE** — indicates the message is reply to WTOR
- **WQEMCSFF** — indicates BRDCAST was specified on the WTO
- **WQEMCSG** — indicates HCONLY was specified on the WTO.

WQEMCSF2 Indicators

- **WQEMCSM** — indicates the message is a hardcopy image of the operator command
- **WQEMCSN** — indicates that NOCPY was specified on the WTO.

WQEMSGTP Message type

WQEROUT Routing codes

WQEFLG1 Indicators

- **WQERETAN** — indicates that the message is retained by AMRF
- **WQENMOD** — indicates the subsystem cannot modify the message
- **WQEPPNA** — indicates the message issued by the problem program
- **WQERISS** — indicates the message is an SVC reissue of a message that has already been processed by SVC WTO MPF and the SSI have already processed the message. Note that MPF processing occurs only during the original SVC WTO. Examples of using this indicator include messages that originate on one system (MVS sysplex), but are transported for display to another system.

WQEDESCD Descriptor codes

WQEJSTCB Address of the job step TCB

WQEVRSN Version level — contains the WQEVRID

WQEMCSE1 Indicator

- **WQEEBUSY** — indicates that BUSYEXIT was specified on the WTO.

WQESYSNM System name

WQEXMOD Copy of the MPF/IEAVMXIT user exit request flags

WQEMLVL Message level

WQEERC Extended routing codes

WQELENG Length of WQE — contains the WQESIZE

WQEKEY Value of the KEY keyword on the WTO

WQETOKN Value of the TOKEN keyword on the WTO

WQECNID Console ID

WQEOJBID Originating job ID

WQEOJBNM Originating job name

WQEPRTY Value of the PRTY keyword on the WTO

WQEAUTOT Value of the AUTOTOKEN from MPF

WQEERFS Extended request flags from the MPF/IEAVMXIT user exit

WQECNNME Console name

WQECART Value specified on the CART keyword on the WTO

WQEBENIP Indicators

- **WQEDOMD** — indicates the message has been deleted by the DOM macro.
- **WQENBEW** — indicates the message created by the branch-entered WTO. Branch-entered WTOs are WTOs that MVS has called for subsequent SVCs. Note that the ASCB/TCB for SSI function code 9 is not the same as the ASCB/TCB of the issuer of the branch-entered WTO.
- **WQENHABD** — indicates the message has been displayed on the IPL or system console. This is a result of issuing a WTO with SYNCH=YES specified.

WQECASEL Message color

WQEHASEL Message highlighting

WQEIASEL Message intensity

WQEMISC Indicator

- **WQEAUTO** — indicates AUTO(Y) specified in the MPF for this message.

SSWT Contents for a Multi-line WTO: MVS sets the following fields in the SSWT control block on input for a multi-line WTO:

See "SSWT Contents for a Single-line WTO" on page 200 for the fields that MVS sets as they are the same except for the SSWTMIN field which contains the following:

Field Name	Description
SSWTMIN	Address of the minor WQE

Figure 23 on page 204 shows the environment for a multi-line WTO in the SSWT control block.

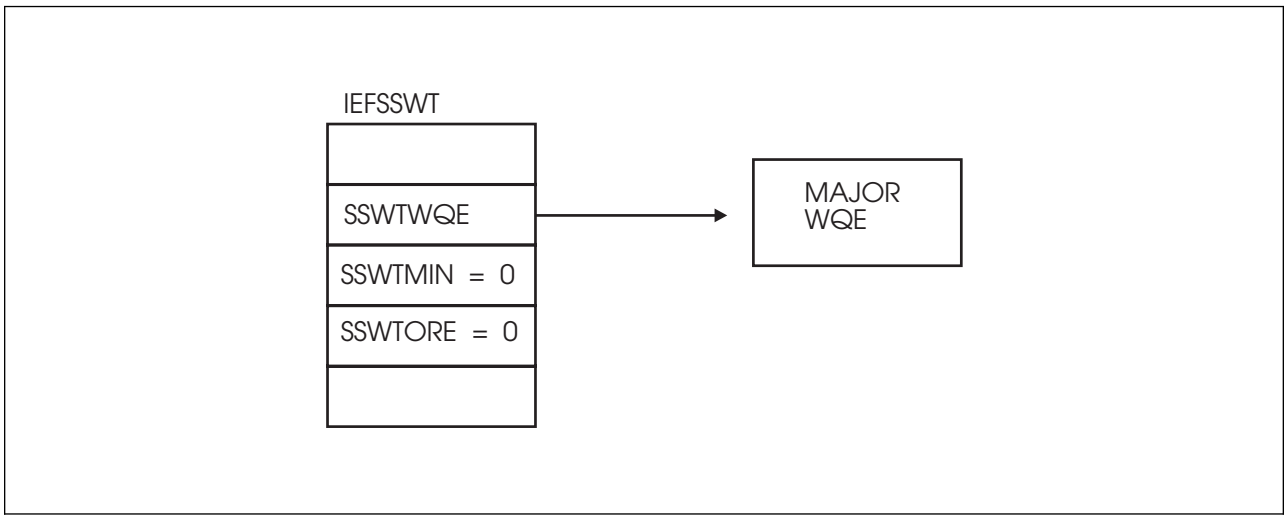

Figure 23. Environment for a Multi-line WTO in the SSWT Control Block

WQE (major WQE) Contents for the First Line of a Multi-line WTO: MVS sets the following fields in the WQE control block for the first line of a multi-line WTO:

Field Name	Description
WMJMMLW	Multi-line indicator • **WMJMMLWB** — indicates the WQE is multi-line
WMJMAREA	Value specified on the WTO AREA keyword
WMJMTXTL	Length of message text
WMJMTS	EBCDIC time stamp
WMJMJBNM	Jobname (inserted by the primary subsystem)
WMJMTXT	Message text
WMJMDSP	Indicator • **WMJMDSPH** — indicates the message issued by the authorized program
WMJMASID	ASID of the message issuer
WMJMTCB	TCB address of the message issuer
WMJMSEQ#	Message DOM id
WMJMMCS1	Indicators • **WMJMCS1A** — indicates that the WMJMRTC and WMJMDEC fields are valid • **WMJMCS1B** — indicates that the WMJMCNID and WMJMCNME fields are valid • **WMJMCS1C** — indicates the message is a command response • **WMJMCS1D** — indicates the WMJMMT field is valid • **WMJMCS1E** — indicates the message is a reply to the WTOR • **WMJMCS1F** — indicates the BRDCAST keyword is specified on the WTO

- **WMJMCS1G** — indicates the HCONLY is specified on the WTO

WMJMMCS2 Indicators

- **WMJMCS2E** — indicates the message is the hardcopy image of the operator command
- **WMJMCS2F** — indicates the NOCPY is specified on the WTO

WMJMT Message type

WMJMRTC Routing codes

WMJMFLG1 Indicator

- **WMJMRETN** — indicates the message will be retained by AMRF
- **WMJMNMOD** — indicates the subsystem cannot modify the message
- **WMJMPPNA** — indicates the message is issued by the problem program
- **WMJMRISS** — indicates the message is an SVC reissue of a message that has already been processed by SVC WTO. MPF and the SSI have already processed the message. Note that MPF processing occurs only during the original SVC WTO. Examples of using this indicator include messages that originate on one system (MVS sysplex), but are transported for display to another system (JES3 complex).

WMJMDEC Descriptor codes

WMJMJTCB Address of job step TCB

WMJMVRSN Version level — contains the WQEVRID

WMJMCE1 Indicator

- **WMJEBUSY** — indicates the BUSYEXIT is specified on the WTO

WMJMSNM System name

WMJMXMOD Copy of the MPF/IEAVMXIT user exit request flags

WMJMMLVL Message level

WMJMERC Extended routing codes

WMJMLENG Length of WQE — contains the WMJMSIZE

WMJMKEY Value of the KEY keyword on the WTO

WMJMTOKN Value TOKEN keyword on the WTO

WMJMCNID Console ID

WMJMOJBI Originating job ID

WMJMOJBN Originating job name

WMJMPRTY Value of the PRTY keyword on the WTO

WMJAUTOT Value of the AUTOTOKEN from the MPF

WMJERFS Extended request flags from the MPF/IEAVMXIT user exit

WMJMCNME Console name

WMJMCART Value is specified on the CART keyword on the WTO

WMJBENIP Indicators

- **WMJMDOMD** — indicates the message has been deleted by the DOM macro.
- **WMJMNBEW** — indicates the message created by the branch-entered WTO. Branch-entered WTOs are WTOs that MVS has called for subsequent SVCs. Note that the ASCB/TCB for SSI function code 9 is not the same as the ASCB/TCB of the issuer of the branch-entered WTO.
- **WMJMNHABD** — indicates the message has been displayed on the IPL or system console. This is a result of issuing a WTO with SYNCH=YES specified.

WMJCASEL Message color

WMJHASEL Message highlighting

WMJIASEL Message intensity

WMJMMISC Indicator

- **WMJMAUTO** — indicates the AUTO(Y) is specified in MPF for this message

WQE (minor WQE) Contents for Subsequent Lines of a Multi-line WTO: MVS sets the following fields in the WQE on input for subsequent lines of a multi-line WTO:

Field Name	Description
WMNMLT1	Line type indicators • **WMNMLT1B** — label line • **WMNMLT1C** — data line • **WMNMLT1D** — end line **Note:** The WMNMLT1C and WMNMLT1D fields can be on at same time.
WMNMLTH1	Length of the minor WQE
WMNMTL1	Length of the minor text
WMNMTXT1	Minor line text
WMN1XMOD	Copy of the request flags from the MPF/IEAVMXIT user exit

Figure 24 on page 207 shows the environment for minor lines of a multi-line WTO in the SSWT control block.

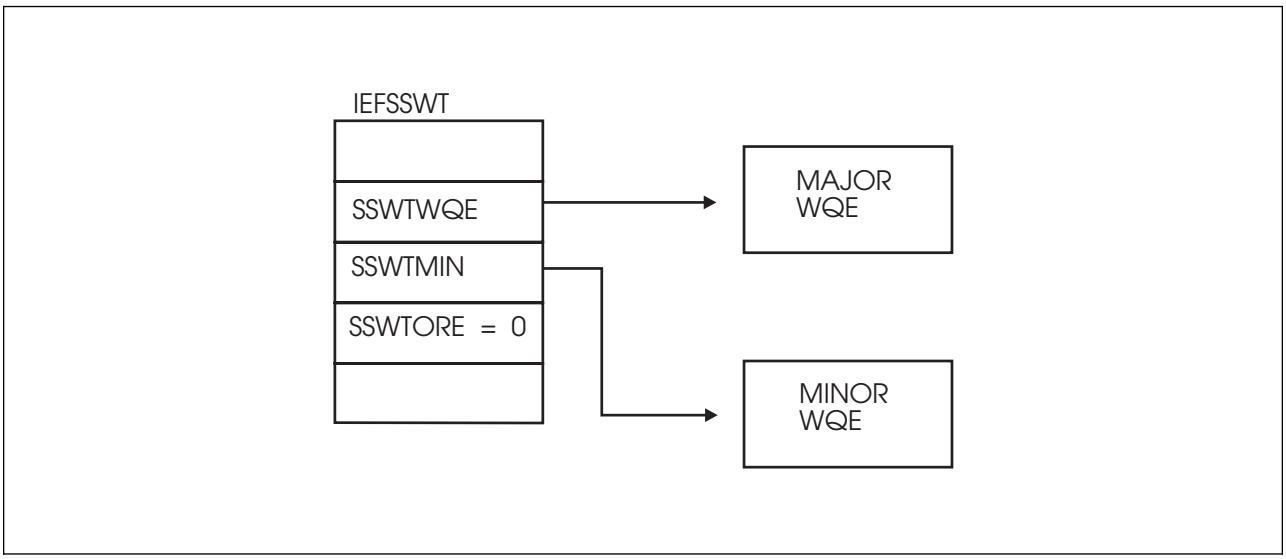

Figure 24. Environment for Minor Lines of a Multi-line WTO in the SSWT Control Block

Multi-line Use Information

For multi-line messages, the major WQE is only presented upon the first call of the function routine. If the function routine wants to process the minor WQEs (minor lines) associated with the message, it must set the WMJMRPML indicator in the WMJMXMOD field of the Major WQE. MVS calls the function routine for each minor WQE until all minor lines have been processed. Note that when processing minor WQEs, the major WQE is intended for read access only. Changes to the major WQE can only be made during the first SSI call for this message. The first SSI call has no minor WQEs present.

SSWT Contents for a WTOR (always single-line): MVS sets the following fields in the SSWT control block on input for a WTOR (always single-line):

See "SSWT Contents for a Single-line WTO" on page 200 for the fields that MVS sets as they are the same except for the SSWTORE field which contains the following:

Field Name	Description
SSWTORE	Address of the ORE control block

WQE Contents for a WTOR (always single-line): The fields in the WQE control block for a WTOR (always single-line) that MVS sets on input contain the same information as the WQE control block for a single-line WTO. See "WQE Contents for a Single-line WTO and WTOR" on page 209 for this information.

ORE Contents for a WTOR (always single-line): MVS sets the following fields in the ORE control block on input for a WTOR (always single-line):

Field Name	Description
ORERPYA	Address of the WTOR issuer's reply buffer
OREECBA	Address of the WTOR issuers ECB
ORECBID	Acronym — 'ORE'
OREVRSN	Version level — OREVRID

ORELNTH Maximum length of the requested reply (specified by the WTOR issuer)

ORERPIDB Binary reply ID

Figure 25 shows the environment for a WTOR in the SSWT control block.

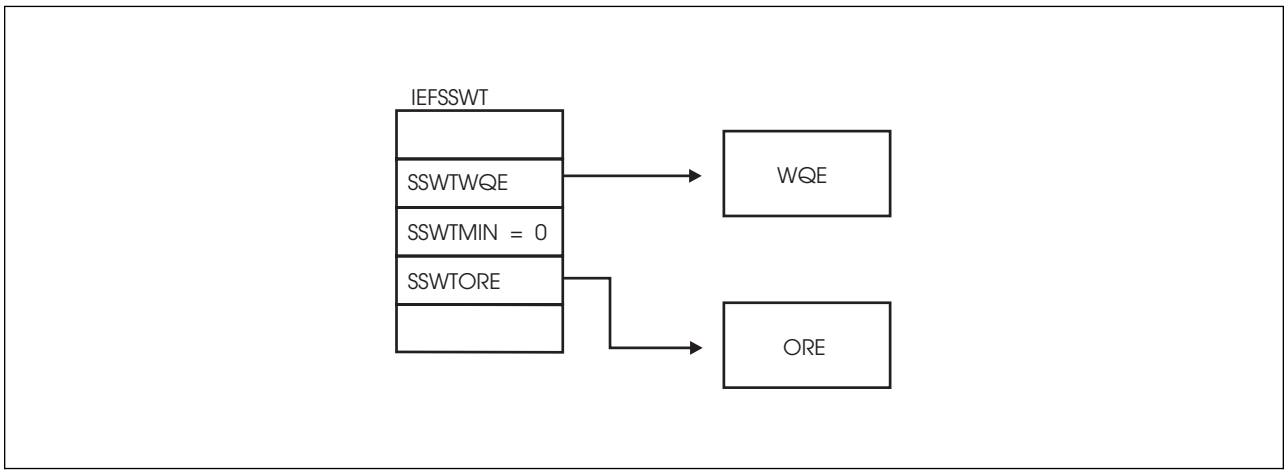

Figure 25. Environment for a WTOR in the SSWT Control Block

Output Register Information

Upon exit from the function routine, the general purpose registers must contain:

Register	Contents
0 — 12	Restored to contents on entry
14	Return address
15	Return code

Return Code Information

For MVS to process broadcast functions properly, you must use the following return code conventions. When a routine returns control to the SSI:

- Set register 15 to 0.
- Set the SSOBRETN field in the SSOB control block to one of the following:

Return Code (Decimal)	Meaning
SSWTRTOK (0)	The function routine recognized the request but did not process it.
SSWTNDSP (4)	Do not display message; hardcopy message
SSWTOKNH (8)	Display message; do not hardcopy message
SSWTNDNH (12)	Do not display message; do not hardcopy message

Output Parameters

Output parameters for the function routine are:

- WQE

WQE Contents for a Single-line WTO and WTOR: The contents of the following fields in the WQE control block for a single-line WTO and WTOR on output are:

Field Name	Description
WQETXTLN	New length of the message text (if text was altered)
WQETXT	New/changed message text
WQEMSGTP	New/changed message type. The WQEMCSD field must be set appropriately.
WQEROUT	New/changed routing codes. The WQEMCSA field must be set appropriately.
WQEDESCD	New/changed descriptor codes. The WQEMCSA field must be set appropriately. The WQEROUT field must be non-zero.
WQEERC	New/changed extended routing codes
WQECASEL	New/changed message color
WQEHASEL	New/changed message highlighting
WQEIASEL	New/changed message intensity

WQE Contents for a Multi-line WTO (major line): The contents of the following fields in the WQE control block for a multi-line WTO (major line) on output are:

Field Name	Description
WMJMTXTL	New length of the message text (if text was altered)
WMJMTXT	New/changed message text
WMJMXMOD	Copy of the MPF/IEAVMXIT user exit request flags
WMJMRPML	Set this field if the function routine should process minor lines
WMJMMT	New/changed message type. The WMJMCS1D field must be set.
WMJMRTC	New/changed routing codes. The WMJMCS1A field must be set.
WMJMDEC	New/changed descriptor codes. The WMJMCS1A field must be set. The WMJMRTC field must be non-zero.
WMJMERC	New/changed extended routing codes
WMJCASEL	New/changed message color
WMJHASEL	New/changed message highlighting
WMJIASEL	New/changed message intensity

WQE Contents for a Multi-line WTO (minor line): The contents of the following fields in the WQE control block for a multi-line WTO (minor line) on output are:

Field Name	Description
WMNMTL1	New length of the minor text (if the WMNMTXT1 field is modified)
WMNMTXT1	New/changed minor line text

End of Product-sensitive programming interface

Product-sensitive programming interface

Command Processing Call — SSI Function Code 10

The Command Processing call (SSI function code 10) is issued every time a system command is generated. SSI function code 10 allows the SSI to find system commands intended for your installation-written subsystem.

Type of Request

Broadcast SSI call.

Use Information

Your installation can use the Command Processing call (SSI function code 10) to:

- Receive a command for processing
- Alter the text of a command (add additional parameters)
- Monitor command traffic
- Prevent commands from being used on the system.

Issued to

- All active subsystems that indicate they support the Command Processing function when the system (MVS) issues the Command Processing call.

Related SSI Codes

None.

Related Concepts

You should know how to use command authorization and routing command responses through a WTO. See *OS/390 MVS System Commands Summary* and *OS/390 MVS Planning: Operations* for command authority concepts. See *OS/390 MVS Programming: Authorized Assembler Services Reference ALE-DYN* for information on routing command responses to operator consoles using the CONSID keyword.

Environment

Review “Function Routines/Function Codes” on page 157, which describes both the general environment on entry to your function routine and other programming considerations that your function routine should take into account.

If you decide to set up your subsystem to handle command processing calls, make sure that your function routine is in place before you enable the subsystem to handle SSI function code 10. IBM recommends that you use the IEFSSVT macro to notify MVS that your subsystem should be given control whenever Command Processing calls are made. IEFSSVT macro services are available only to dynamic subsystems. Subsystems that are not dynamic can still use the IEFJSVEC service; see “Building the SSVT” on page 279 and “Enabling Your Subsystem for New Functions” on page 284 for more information.

Do not code a function routine that enters an explicit WAIT or uses a system service that enters a WAIT. Entering a wait can cause degraded system performance.

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your function routine:

- IEFSSOBH
- IEFJSSIB
- IEFSSCM
- IEZMGCR

The function routine receives control in the following environment:

Minimum authorization	Supervisor state with PSW key 0
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control parameters	The SSOB, SSIB, and SSCM control blocks reside in storage below 16 megabytes.
Recovery	The function routine should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Reference ENF-IXG* for more information on these macros. Failure to establish a recovery environment ends the processing of the current operator command if an abend occurs. The function routine's recovery should retry. The retry point should take one of the following actions: • Ignore the command • Indicate to the system the command could not be processed • Indicate to the system the command was processed. The system issues error message IEE707E indicating the command failed. **Note:** Refer to "Output Register Information" on page 215 for instructions on what your function routine should specify to the system.

Figure 26 on page 213 shows the environment on entry to the function routine for SSI function code 10.

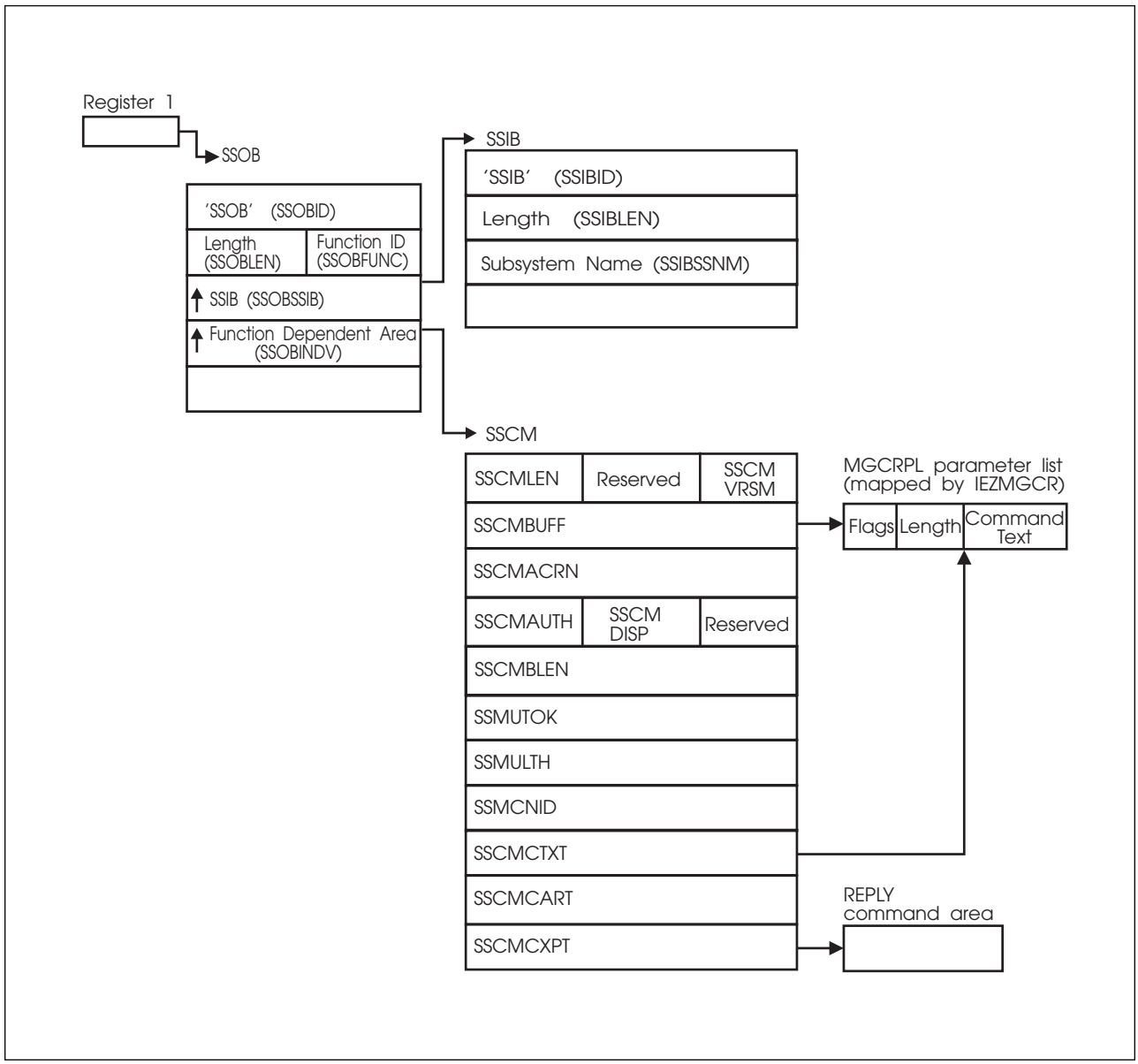

Figure 26. Environment on Entry to the Function Routine for SSI Function Code 10

Input Register Information

On entry to the function routine the general purpose registers contain:

Register	Contents
0	Address of the subsystem's SSCVT
1	Address of the SSOB control block
13	Address of a standard 18-word save area
14	Return address
15	Entry point address

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSCM
- Command Sensitive Area
- MGCRPL

SSOB Contents: MVS sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function code 10 (SSOBCMND)
SSOBSSIB	Address of the SSIB control block
SSOBINDV	Address of the function dependent area (SSCM control block)

SSIB Contents: MVS sets the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name 'MSTR'

SSCM Contents: MVS sets the following fields in the SSCM control block on input:

Field Name	Description
SSCMLEN	Length of the SSCM (SSCMSIZE) control block
SSCMVRSN	Version level of the SSCM (SSCMVRID) control block
SSCMBUFF	Address of the command buffer in the MGCRPL control block
SSCMACRN	Identifier 'SSCM'
SSCMAUTH	Command authority of command issuer — see the SSCM control block information for the definition of flags within this byte.
SSCMDISP	Disposition flags — see the SSCM control block information for the definition of flags within this byte.
SSCMBLEN	Length of the command buffer pointed to by the SSCMBUFF field.
SSCMOLIB	If the command text was changed by symbolic substitution (indicated by an ON value in the SSCMSYMS field), this field contains a DSECT that maps the original command text (the text that existed before symbolic substitution occurred).
SSCMOLIP	If the command text was changed by symbolic substitution (indicated by an ON value in the SSCMSYMS field), this field contains the address of the SSCMOLIB structure.
SSCMSYMS	The command text was changed by symbolic substitution.

SSCMUTOK	Address of the UTOKEN The UTOKEN identifies the issuer of the command. The RACROUTE macro accepts the UTOKEN to perform command authorization checking using a security product (RACF).
SSCMULTH	Length of the UTOKEN
SSCMCNID	4-byte console ID — identifies the console that the command was issued from.
SSCMSCNM	Console name of the console whose ID is in the SSCMCNID
SSCMCTXT	Address of a 126-byte buffer containing the command text
SSCMCLEN	Length of command text
SSCMCART	Command and response token To identify the source of the command, all command responses issued by a function routine through a WTO or WTOR should specify either SSCMCNID or SSSCNM and SSCMCART.
SSCMCXPT	Address of command sensitive area or zero

Command Sensitive Area Contents: MVS sets the following fields in the command sensitive area for a REPLY command on input. The address of this area (when present) is available in the SSCMCXPT field. If not present, SSCMCXPT=0.

Field Name	Description
SSCMCVRB	Command identifier — REPLY
SSCMRTCB	TCB address of the WTOR issuer
SSCMRASI	ASID of the WTOR issuer
SSCMRTXT	Offset to the reply text in the area pointed to by either the SSCMCTXT field or SSCMBUFF+4.
SSCMRFLG	Reply flag • **SSCMRSEC** — indicates whether the REPLY is to a security WTOR (route code of 9).

MGCRPL Contents: The address of the MGCRPL control block is available in the SSCMBUFF. MVS sets the following fields in the MGCRPL control block on input:

Field Name	Description
MGCRLGTH	Length of the command text + 4
MGCRFLG2	Command processing flags — see the MGCRPL control block information for a definition of these flags.
MGCRTEXT	Command text

Output Register Information

Upon exit from the function routine, the general purpose registers must contain:

Register	Contents
0 — 12	Restored to contents on entry
14	Return address
15	Return code

Return Code Information

For MVS to process broadcast functions properly, you must use the following return code conventions for function routines that handle broadcast calls. When a routine returns control to the SSI:

- Set register 15 to 0.
- Set the SSOBRETN field in the SSOB control block to one of the following:

Return Code

(Decimal)	Meaning
SSCMSCMD (0)	The command does not belong to the function routine.
SSCMSUBC (4)	The command belongs to the function routine and was processed.
SSCMIMSG (8)	The command belongs to the function routine, but could not be processed. Message IEE707I is issued.

Output Parameters

Output parameters for the function routine are:

- MGCRPL

MGCRPL Contents: The address of the MGCRPL control block is available in SSCMBUFF. Your function routine can modify the contents of the following fields in the MGCRPL control block on output:

Field Name	Description
MGCRLGTH	Length of the command text plus 4. A new length can be specified. The new length must be greater than or equal to 5, but cannot exceed 130.
MGCRTEXT	Command text — the command text can be altered and replaced. If the length is changed, MGCRLGTH (above) must also be updated.

Note: A function routine that alters the text of the command for processing by either another subsystem or MVS must specify SSOBRETN=0 upon return to the caller.

Restrictions

Only one subsystem can claim ownership of a command and assume responsibility for its processing by assigning a unique command prefix to the subsystem; any command prefixed by that command prefix is owned by that subsystem.

Notes:

1. A command prefix is a character string of one or more alphanumeric and/or national characters. Command prefixes often have a length of one character, although a maximum of eight characters is permitted.
2. See the CPF macro in *OS/390 MVS Programming: Authorized Assembler Services Reference ALE-DYN* for information on registering command prefixes.

General Considerations

Command processors that receive their input from SSI function code 10 should consider:

- Using the 4-byte console ID. This is found in the SSCMCNID field of the SSCM control block. An application that uses the MCSOPER interface (see MCSOPER in *OS/390 MVS Programming: Authorized Assembler Services Reference LLA-SDU* can only be assured of receiving the command response by using this field on a WTO. If a 1-byte console ID must be used, use the value in the SSCMSCID field. Please note, however, that the 1-byte console ID found in the SSCMSCID field cannot guarantee the command response message will reach the MCSOPER user who issued the command.
- Using the SSCMAUTH field. Use the flag settings of the SSCMAUTH field to test the command authority of the caller. This field is mapped by the UCMAUTH field in the UCME (IEECUCM).
- Using the SSCMCART field. All command response messages issued through a WTO should use the values passed in the SSCMCNID field (above) and in the SSCMCART field. The use of these values ensures proper delivery of the message to the command issuer.

Considerations for Command Processing Calls in a Sysplex:

In a sysplex, command processing SSI calls are made to subsystems:

- On the originating console's system only, when the command is not routed to any other system in the sysplex.
- On the originating console's system only, when the command is routed to another system in the sysplex as the result of the location (L=) operand on the command or the specification of a console by name.
- On the receiving system only, when it is a prefix command that is routed through the MCS command prefix facility.
- On both the originating system and the receiving system, when the ROUTE command is issued, as follows:
  - On the originating system for the ROUTE command.
  - On the receiving system for the command that is routed.

Considerations for Commands That Specify System Symbols:

When a command contains system symbols, MVS provides the command text to the SSI *after* it substitutes text for the system symbols. For example, if the following command is entered to display a console group on system SYS1:

```
DISPLAY CNGRP,G=(CN1GRP&SYSCLONE.)
```

The SSI receives the following text (assuming that the default for &SYSCLONE., the last two characters of the system name, is taken):

```
DISPLAY CNGRP,G=(CN1GRPS1)
```

If the function routine requires the original command text (the one that existed *before* symbolic substitution), it can access the SSCMOLIB field in the SSCM (see *OS/390 MVS Data Areas, Vol 1 (ABEP-DALT)* for a description of the IEFSSCM mapping macro, which maps the SSCM).

Do not use the function routine to add or change system symbols in command text. The system cannot substitute text for system symbols that are added or changed through the SSI.

End of Product-sensitive programming interface

Product-sensitive programming interface

Delete Operator Message — SSI Function Code 14

The Delete Operator Message call (SSI function code 14) is issued for every DOM that is created. SSI function code 14 allows the SSI to find DOMs intended for your installation-written subsystem.

Type of Request

Broadcast SSI call.

Use Information

Your installation can use the DOM Processing call (SSI function code 14) to:

- Receive a DOM for processing
- Monitor DOM traffic
- Verify that a WTO or WTOR message has been deleted
- Alter the type of DOM. (IBM does not recommend this use.)

Issued to

- All active subsystems that indicate they support the DOM processing function when the system (MVS) issues the DOM Processing call.

Related SSI Codes

None.

Related Concepts

You should know how to recognize and use DOMs. See *OS/390 MVS Programming: Authorized Assembler Services Reference ALE-DYN* and *OS/390 MVS Programming: Authorized Assembler Services Guide*.

Environment

Review “Function Routines/Function Codes” on page 157, which describes both the general environment on entry to your function routine and other programming considerations that your function routine should take into account.

If you decide to set up your subsystem to handle DOM processing calls, make sure that your function routine is in place before you enable the subsystem to handle SSI function code 14. IBM recommends that you use the IEFSSVT macro to notify MVS that your subsystem should be given control whenever DOM Processing calls are made. IEFSSVT macro services are available only to dynamic subsystems. Subsystems that are not dynamic can use the IEFJSVEC service; see “Building the SSVT” on page 279 and “Enabling Your Subsystem for New Functions” on page 284 for more information.

DOMs occur frequently with MVS. Function routines should therefore be as efficient as possible. Do not code a function routine that enters an explicit WAIT or uses a system service that enters a WAIT because entering a wait can cause degraded system performance.

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your function routine:

- IEFSSOBH
- IEFJSSIB
- IEFJSSOB
- IEFSSDM
- IHADOMC

The delete operator message mapped by IHADOMC represents a DOM.

The function routine receives control in the following environment:

Minimum authorization	Supervisor state with PSW key 0
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control parameters	The SSOB, SSIB, and SSCM control blocks reside in storage below 16 megabytes.
Recovery	The function routine should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Reference ENF-IXG* for more information on these macros. Failure to establish a recovery environment ends the processing of the current DOM if an abend occurs. The function routine's recovery should specify a retry point (address) and return 4 on the SETRP macro before returning to the system. Use the retry point to complete a normal return to the function routine's caller. When the function routine returns to its caller under these circumstances, it should indicate to the system, by setting both register 15 and the SSOBRETN to zero, to take no action against the message. See the next topic, Input Register Information, for how to specify to the system the action you want your function routine to take.

Input Register Information

On entry to the function routine the general purpose registers contain:

Register	Contents
0	Address of the subsystem's SSCVT
1	Address of the SSOB control block
13	Address of a standard 18-word save area
14	Return address
15	Entry point address

Input Parameters

Input parameters for the function routine are SSOB, SSIB, SSDM, and DOMC.

SSOB Contents: MVS sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'

SSOBLEN	Length of the SSOB (SSIBHSIZ) control block
SSOBFUNC	SSI function code 14 (SSOBDOM)
SSOBSSIB	Address of the SSIB control block
SSOBRETN	Return code from the previous function routine (when SSI function code 14 is operating in broadcast mode)
SSOBINDV	Address of the function-dependent area (SSDM control block)

SSDM Contents: MVS sets these fields in the SSDM control block on input:

Field Name	Description
SSDMLEN	Length of the SSDM (SSDMSIZE) control block
SSDMVRSN	Version level of the SSDM (SSDMVRID) control block
SSDMACRN	Identifier 'SSDM'
SSDMSEND	Indicator that the DOM request should be communicated to other systems
SSDMDMCB	The address of that part of the DOMC that is passed to the subsystem
SSDMDMC2	The address of the entire DOMC that is passed to the subsystem

DOMC Contents: The address of the DOMC control block is in SSDMDMC2. See *OS/390 MVS Data Areas, Vol 2 (DCCB-ITZYRETC)* for the DOMC information.

Output Register Information: Upon exit from the function routine, the general purpose registers must contain:

Register	Contents
0 — 12	Restored to contents on entry
14	Return address
15	Return code

Return Code Information: For MVS to process broadcast functions properly, you must use the following return code convention for function routines that handle broadcast calls: when a routine returns control to the SSI, set register 15 to 0.

The DOM Processing call does not have any return codes.

Output Parameters: The output parameter for the function routine is DOMC.

IBM recommends that the function routine does not alter the DOMC.

Restrictions: None.

General Considerations: IBM recommends that the function routine does not alter the DOMC.

End of Product-sensitive programming interface

Product-sensitive programming interface

Early Notification of End-of-Task Call — SSI Function Code 50

The Early Notification of End-of-Task call (SSI function code 50) provides the ability to do task-related resource clean up. Whenever a task ends, all active subsystems that are enabled to receive SSI function code 50 are given control from the SSI before resource managers are given control. Each subsystem function routine will get control for every task that ends.

Note: This broadcast request is issued before many resource managers have been given control, but not all resource managers. For instance, the following resource managers receive control before this Early Notification of End-of-Task call:

- Availability Manager (AVM)
- SVC Dump

Type of Request

Broadcast SSI call.

Use Information

Your subsystem can use SSI function code 50 to clean up any resources for a task associated with a particular subsystem, and free any resources not normally handled by a resource manager.

Because your function routine will get control for every Early Notification of End-of-Task call, using your own subsystem might not be the most efficient way to do your own clean up for ending tasks. The preferred way to define your own resource manager is through the use of the RESMGR macro. The RESMGR service can be used to receive control for specific ending tasks, rather than having to check each ending task or address space to see if it used the subsystem. For a general description of resource managers and how they can be defined at both IPL time and dynamically, see *OS/390 MVS Programming: Authorized Assembler Services Guide*.

Issued to

- All active subsystems that indicate they support the Early Notification of End-of-Task function when the system (MVS) issues the Early Notification of End-of-Task call.

Related SSI Codes

SSI function code 50 is almost identical to SSI function code 4 (End-of-Task call). The only difference is that, for SSI function code 50, your function routine is given control before most resource managers are given control, whereas, for SSI function code 4, your function routine is given control after most resource managers are given control. If you are interested in obtaining control after most resource managers have been invoked, see SSI function code 4 (End-of Task).

Related Concepts

None.

Environment

Review “Function Routines/Function Codes” on page 157, which describes both the general environment on entry to your function routine and other programming considerations that your function routine should take into account.

If you decide to set up your subsystem to handle Early Notification of End-of-Task calls, make sure that your function routine is in place before you enable the subsystem for SSI function code 50. IBM recommends that you use the IEFSSVT macro to notify MVS that your subsystem should be given control whenever Early Notification of End-of-Task calls are made. IEFSSVT macro services are available only to dynamic subsystems. Subsystems that are not dynamic can still use the IEFJSVEC service; see “Building the SSVT” on page 279 and “Enabling Your Subsystem for New Functions” on page 284 for more information.

The subsystem function routine runs in the address space of the ending task. Because each subsystem function routine is called for every ending task, the subsystem function routine should not be a long running program. That is, the function routine should quickly determine if the subsystem was ever associated with the ending task and, if not, return to the system. Also, do not code a function routine that enters an explicit WAIT or uses a system service that enters a WAIT. Entering a WAIT can cause degraded system performance.

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your function routine:

- IEFSSOBH
- IEFJSSIB
- IEFSSET

The function routine receives control in the following environment:

Minimum authorization	Supervisor state with PSW key 0
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control parameters	The SSIB, SSOB, and SSET control blocks reside in storage below 16 megabytes.
Recovery	The function routine should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on an ESTAE-type recovery environment.

Figure 27 on page 224 shows the environment on entry to the function routine for SSI function code 50.

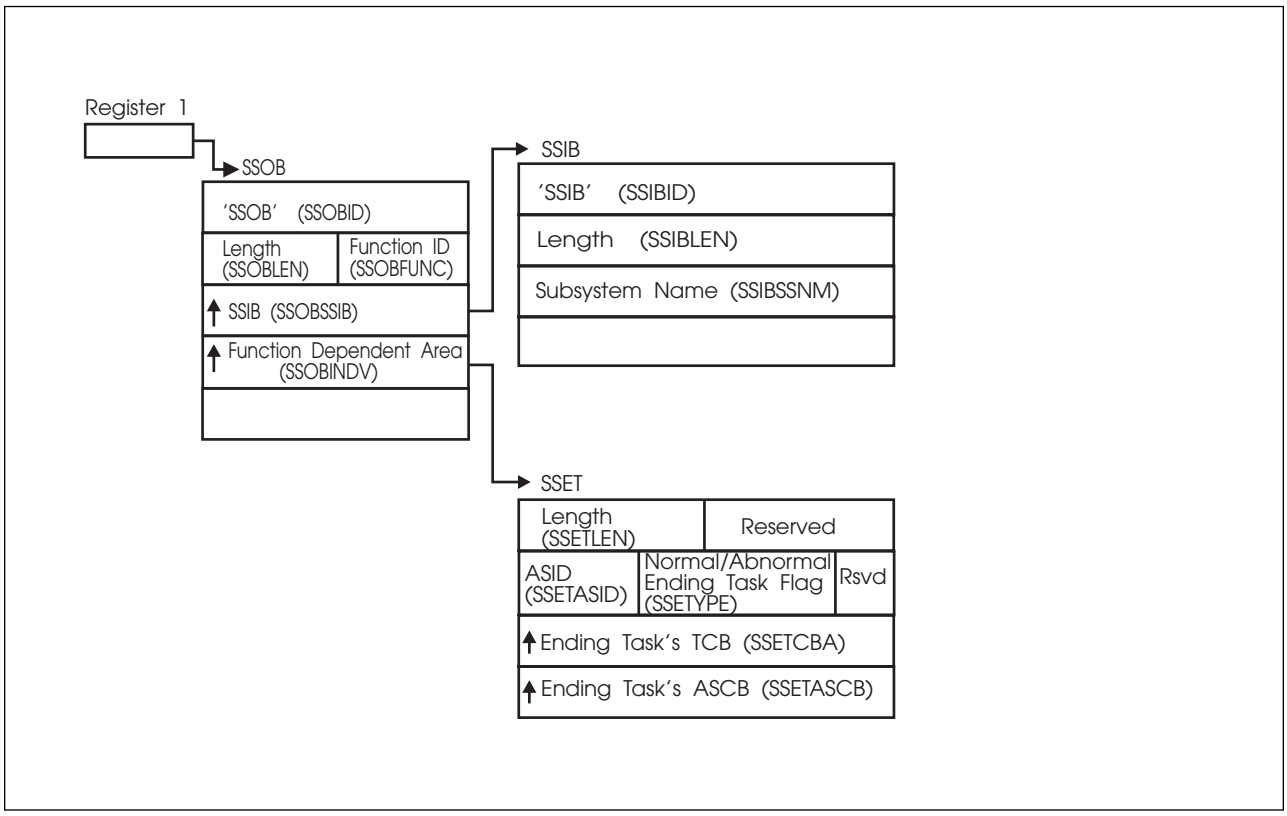

Figure 27. Environment on Entry to the Function Routine for SSI Function Code 50

Input Register Information

On entry to the function routine the general purpose registers contain:

Register	Contents
0	Address of the subsystem's SSCVT
1	Address of the SSOB control block
13	Address of a standard 18-word save area
14	Return address
15	Entry point address

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSET

SSOB Contents: The following fields in the SSOB control block are set on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function code 50 (SSOBFEOT)

SSOBSSIB	Address of the SSIB control block
SSOBRETN	Return code from previous subsystem function routine or zero. Since broadcast requests are routed to all active subsystems, upon entry to the function routine SSOBRETN contains the return code value set by the previously invoked subsystem function code(s) or zero. See “Output Register Information” for a list of possible SSOBRETN return codes.
SSOBINDV	Address of the function dependent area (SSET control block)

SSIB Contents: The following fields in the SSIB control block are set on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE) control block
SSIBSSNM	Subsystem name — name of subsystem which is enabled to receive this function code.

SSET Contents: The following fields in the SSET control block are set on input:

Field Name	Description
SSETLEN	Length of the SSET (SSETSIZE) control block
SSETASID	ASID of address space in which task was active
SSETFLAG	Flag indicators • **SSETYPE ON** — indicates an abnormal ending task • **SSETYPE OFF** — indicates a normal ending task
SSETCBA	Address of ending task's TCB
SSETASCB	Address of ending task's ASCB

Output Register Information

Upon exit from the function routine, the general purpose registers must contain:

Register	Contents
0 — 12	Restored to contents on entry
14	Return address
15	Return code

Return Code Information

For MVS to process broadcast functions properly, you must use the following return code conventions for function routines that handle broadcast calls. When a routine returns control to the SSI:

- Set register 15 to 0.
- Set the SSOBRETN field in the SSOB control block to one of the following:

  Return Code

(Decimal)	Meaning
0	The function routine recognized the request but did not process it.
4	The function routine recognized the request and processed it.

End of Product-sensitive programming interface

Request Subsystem Version Information Call — SSI Function Code 54

The Request Subsystem Version Information Call (SSI function code 54) provides a requesting program the ability to obtain version-specific information about a user-supplied subsystem. The information in “Request Subsystem Version Information Call — SSI Function Code 54” on page 60 describes what happens when SSI function code 54 is issued to the IBM-supplied subsystems (master or JES) by user-provided calling programs or routines.

The information that follows describes what a user-supplied subsystem needs to provide so that it can process incoming SSI function code 54 requests from callers that request information like the information provided by the two IBM-supplied subsystems. The user-supplied subsystem must then provide both the function routine to handle this request, as well as the information concerning the specific returned information. The user-supplied subsystem must provide information to the callers, because all version information returned to the caller is defined by, and has meaning only to, the user-supplied subsystem.

Type of Request

Directed SSI call.

Use Information

A subsystem may want to allow users to obtain the following information about itself:

- Product function modification identifier (FMID)
- Product version number
- Subsystem common name (such as 'XYZ1')
- Any other information that the subsystem wishes to present to the caller.

Issued to

- A user-supplied subsystem

Related SSI Codes

None.

Related Concepts

You need to understand:

- What the caller of the SSI function code 54 must code and what the caller expects to receive. See “Request Subsystem Version Information Call — SSI Function Code 54” on page 60 for a description of this Request Subsystem Version Information call from a calling program's point of view.
- What the format of the IEFSSVI functional extension is as defined in “Request Subsystem Version Information Call — SSI Function Code 54” on page 60.

Environment

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in you function routine:

- IEFSSOBH
- IEFJSSIB
- IEFSSVI

The function routine receives control in the following environment:

Minimum authorization	Any state, any key, depending on the implementation of the function routine. However, IBM suggests that you process this function in problem state, any key.
Dispatchable unit mode	Task
AMODE	24-bit or 31-bit, depending on the implementation of the function routine. If 24-bit AMODE, the callers of the routine must obtain all their control parameters below 16 megabyte storage so that the serving routine can address them. IBM recommends this program runs in AMODE 31.
Cross memory mode	PASN=HASN=SASN
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control parameters	Above or below 16 megabytes depending on the implementation of the routine. However, if the routine runs in AMODE 24, the caller must obtain the control parameters and pass to the serving routine below the line so that it can address them.
Recovery	The function routine should provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on how to set up an ESTAE-type recovery environment.

Figure 28 shows the environment on entry to the function routine for SSI function code 54.

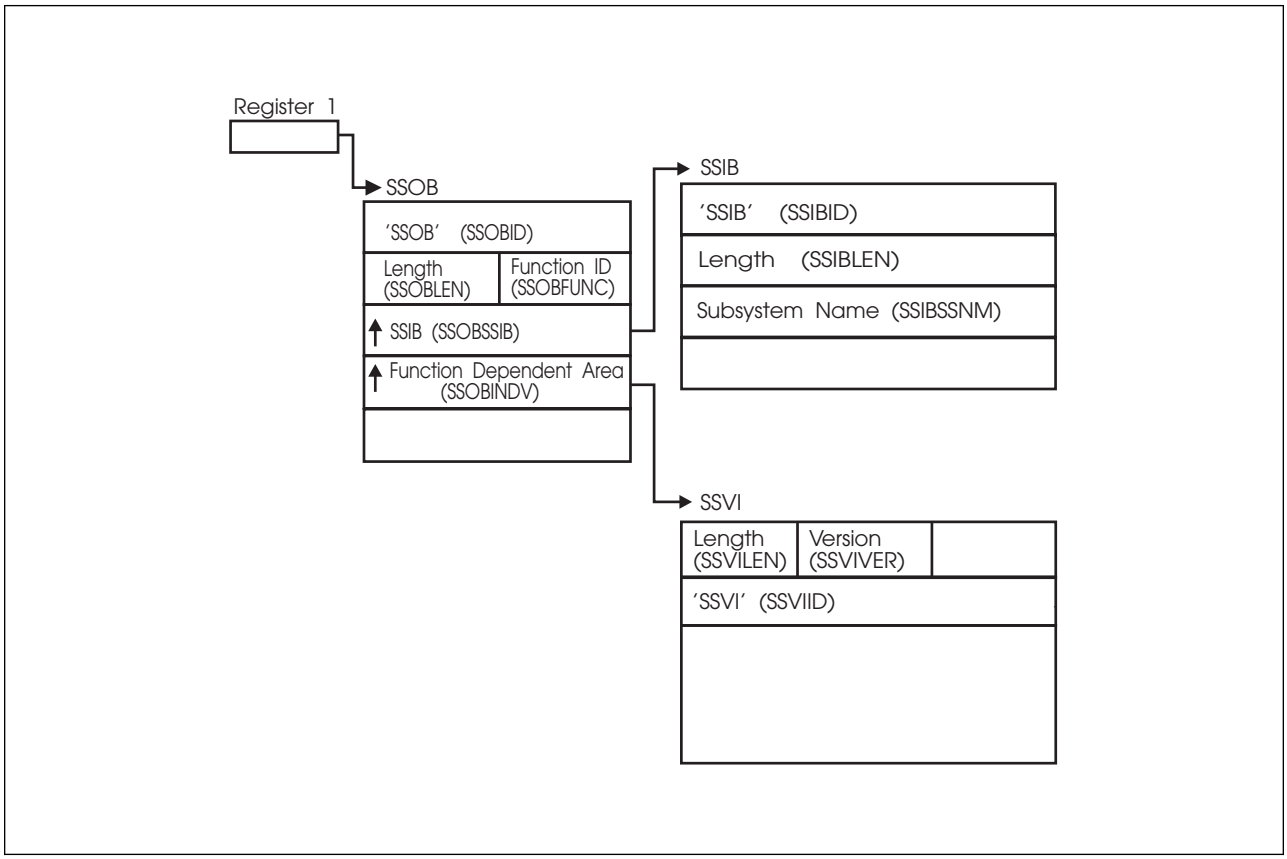

Figure 28. Environment on Entry to the Function Routine for SSI Function Code 54

Input Register Information

On entry to the function routine the general purpose registers contain:

Register	Contents
0	Address of the SSCVT
1	Address of the SSOB control block
13	Address of a standard 18-word save area
14	Return address of the requestor of the service
15	Entry point address

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSVI

SSOB Contents: The caller sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of SSOB control block
SSOBFUNC	SSI function code 54 (SSOBSSVI)
SSOBSSIB	Address of SSIB control block
SSOBINDV	Address of function dependent area (SSVI control block)

SSIB Contents: The caller sets the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of SSIB control block
SSIBSSNM	Subsystem name — name of the user provided subsystem invoked

SSVI Contents: See “Request Subsystem Version Information Call — SSI Function Code 54” on page 60 for the format of the input SSVI that your function routine expects to process.

Output Register Information

Upon exit from the function routine, the general purpose registers must contain:

Register	Contents
0 — 12	Restored to contents on entry
14	Return Address
15	Return code

Return Code Information

Set register 15 to zero.

Output Parameters

Output parameters for the function routine are:

- SSVI

The function routine performs processing to return the subsystem version information, and returns this information to the caller through settings, field updates, and pointers to information contained in the SSVI control block.

In addition, the information fields (For example, SSVIFMID, SSVIVERS, and SSVICNAM) are defined by, and have meaning only to, the function routine.

SSVI Contents: If the function routine returned successfully to the caller, the function routine may return the following information in the SSVI control block:

Field Name	Description
SSOBRETN	The function routine sets this field to SSVIOK (decimal 0).
SSVIRLEN	The function routine sets this field to the number of bytes that is used to return the requested information. This value includes the fixed section as well as the system variable section.
SSVIRVER	The function routine sets this field to the version of the SSVI macro used (SSVICVER).
SSVIFLEN	The function routine sets this field to the length of the fixed header output section (SSVIFSIZ).
SSVIASID	A 2-byte field that contains the ASID of the subsystem if the subsystem has an address space and supports the use of this field.
SSVIFMID	The function routine sets this field (left-justified, and padded to the right with blank (X'40') characters) to the function routine's FMID, if available. See *MVS Packaging Rules* for additional information about FMIDs and IBM's suggestions for choosing one (optional) for user-supplied or vendor-supplied programs that are installed using SMP/E facilities.
SSVIVERS	The function routine sets this field (left-justified, and padded to the right with blank (X'40') characters) to the version of the subsystem installed. The function routine defines and uses the version naming conventions and meanings.
SSVICNAM	The function routine sets this field (left-justified, and padded to the right with blank (X'40') characters) to the processing subsystem's common name. For example, a subsystem defined as 'JOHN' might choose to return the SSVICNAM value of 'JOHNNY'. The subsystem defines the common name.
SSVIPLVL	This 1-byte field contains the product level of the subsystem. The content of the field is defined by the subsystem. This field should only be used if the caller-supplied version in field SSVIVER is greater than or equal to 2.

SSVISLVL This 1-byte field contains the service level of the subsystem. The content of the field is defined by the subsystem.

This field should only be used if the caller-supplied version in field SSVIVER is greater than or equal to 2.

SSVIUDOF The function routine sets this field to zero (there is no installation variable output section).

SSVISDOF The function routine sets this field to the offset of the start of the system variable section (same value as SSVIFLEN), if the function routine wants to supply system variable information.

The DSECT SSVIVDAT mapping begins at this offset, within the SSVI control block that the caller provided to the function routine. The caller must provide an SSVI control block large enough to contain the fixed section and system variable section beginning at this offset (SSVISDOF) past the start of the fixed section (SSVIHEAD).

The function routine may provide a system variable output section that contains additional information returned to the caller and mapped using SSVIVDAT. If it doesn't provide this, the SSVISDOF field must be set to zero.

The function routine sets the first halfword of this system variable information section to the length of the system variable section (not including itself) in the SSVIVLEN field, so that the first byte of the character string starts past the SSVIVLEN field.

For example, the function routine may choose to return the following character string to the caller:

```
,EXAMPLE_SWITCH='NO'
```

The function routine places the length of the character string, 20 bytes (decimal) in the SSVIVLEN field, followed by the character string, beginning at the SSVIDAT field. The first byte at the SSVIDAT field contains an EBCDIC value for the comma in front of the word 'EXAMPLE'.

Note that the comma is the first character of the character string even if only a single keyword value is being returned. See the “Request Subsystem Version Information Call — SSI Function Code 54” on page 227 for more information on the syntax of the returned system variable sections. IBM recommends that your function routine also use the same syntax conventions.

If the function routine returned unsuccessfully to the caller, the system function may provide any of the following processing depending on the reasons for the unsuccessful return:

- Insufficient Storage

  The function routine has determined that the requestor has not supplied a storage area large enough to contain the requested information. That is, the caller has not provided a value in the SSVILEN field that is large enough to contain both the fixed section, as well as any possible system variable section (length plus actual data). The function routine therefore sets the following fields:

Field Name	Description
SSOBRETN	The function routine sets SSOBRETN to the value of SSVINSTR (decimal 8).
SSVIRLEN	The function routine sets SSVIRLEN to the amount of storage needed to satisfy the request. The function routine determines the SSVIRLEN value by adding the length of the fixed header section (SSVIFSIZ) to the length of the system variable output section, plus two bytes (for the length value) of the returned string. Suppose the caller in the example above only provided 30 decimal bytes for the returned information. Our function routine would return decimal 70 in the SSVIRLEN field as follows: 1. 48 — decimal value of the defined symbol SSVIFSIZ 2. 2 — length of the SSVIVLEN field (2 bytes long) 3. 20 — length of the character string (,EXAMPLE_SWITCH='NO').

All other fields in the SSVI control block are not set by the function routine.

- Requestor does not provide a valid SSVI

  The SSVI control block that is supplied by the caller should be validity-checked by the function routine. The following validations are suggested:

  – The SSOBINDV value in the SSOB control block should be non-zero.

  – The SSVILEN field supplied by the caller should be equal to or greater than SSVIMSIZ (an equated value within the SSVI).

  – The SSVIID field supplied by the caller should contain the EBCDIC characters 'SSVI'.

  – The SSVIVER field supplied by the caller should be non-zero.

    The current version of the SSVICVER field is equated to SSVIVONE (decimal 1).

    Future versions of the SSVI control block must have their version number increased, so both the caller and the function routine are able to determine what information is expected and provided.

  If any of the above conditions are not true, the function routine must set the SSOBRETN field as follows:

Field Name	Description
SSOBRETN	The function routine sets SSOBRETN to the value of SSVIPARM (decimal 16).

  All other fields in the SSVI control block are not set by the function routine.

- An abend or logical error within the function routine occurs

  It is possible that an abend or logical error occurs in your routine. IBM supplies an equate symbol for this return code. If your routine chooses to use it, the function routine must set the following field:

Field Name	Description
SSOBRETN	The function routine sets SSOBRETN to the value of SSVIABLG (decimal 24).

All other fields in the SSVI control block are not set by the function routine.

SMF SUBPARM Option Change Call — SSI Function Code 58

The SMF SUBPARM Option Change call (SSI function code 58) allows a user subsystem to be notified that the SUBPARM option in the SMF parmlib member for their subsystem has been changed.

Type of Request

Directed SSI call.

Use Information

Your subsystem can use SSI function code 58 when it wants to be notified of changes that have been made to the SMF SUBPARM parameter. The SMF SUBPARM parameter is used to pass accounting information to the subsystem.

Issued to

- The subsystem whose SUBPARM option was changed by the SET SMF or SETSMF command.

Related SSI Codes

None.

Related Concepts

You need to understand:

- The interaction between the SMF parmlib option (SUBPARM), the SMF macros (SMFSUBP and SMFCHSUB) and this function code. See “Passing Accounting Parameters to Your Subsystem” on page 164 for a description of this relationship and an example of the associated processing.

Environment

Review “Function Routines/Function Codes” on page 157, which describes both the general environment on entry to your function routine and other programming considerations that your function routine should take into account.

If you decide to set up your subsystem to handle SMF SUBPARM option change calls, make sure that your function routine is in place before you enable the subsystem to handle SSI function code 58. IBM recommends that you use the IEFSSVT macro to notify MVS that your subsystem should be given control whenever SMF SUBPARM Option Change calls are made. IEFSSVT macro services are available only to dynamic subsystems. Subsystems that are not dynamic can still use the IEFJSVEC service; see “Building the SSVT” on page 279 and “Enabling Your Subsystem for New Functions” on page 284 for more information.

Data areas commonly used by SSI function code 58 are mapped by the following mapping macros. IBM recommends you include them in your function routine:

- IEFSSOBH
- IEFJSSIB
- IEFSSSM

The function routine receives control in the following environment:

Minimum authorization	Supervisor state with PSW Key 0
Dispatchable unit mode	Task

AMODE	24-bit or 31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control parameters	The SSOB, SSIB, and SSSM control blocks reside in storage below 16 megabytes.
Recovery	None

Figure 29 shows the environment at the time of the call for SSI function code 58.

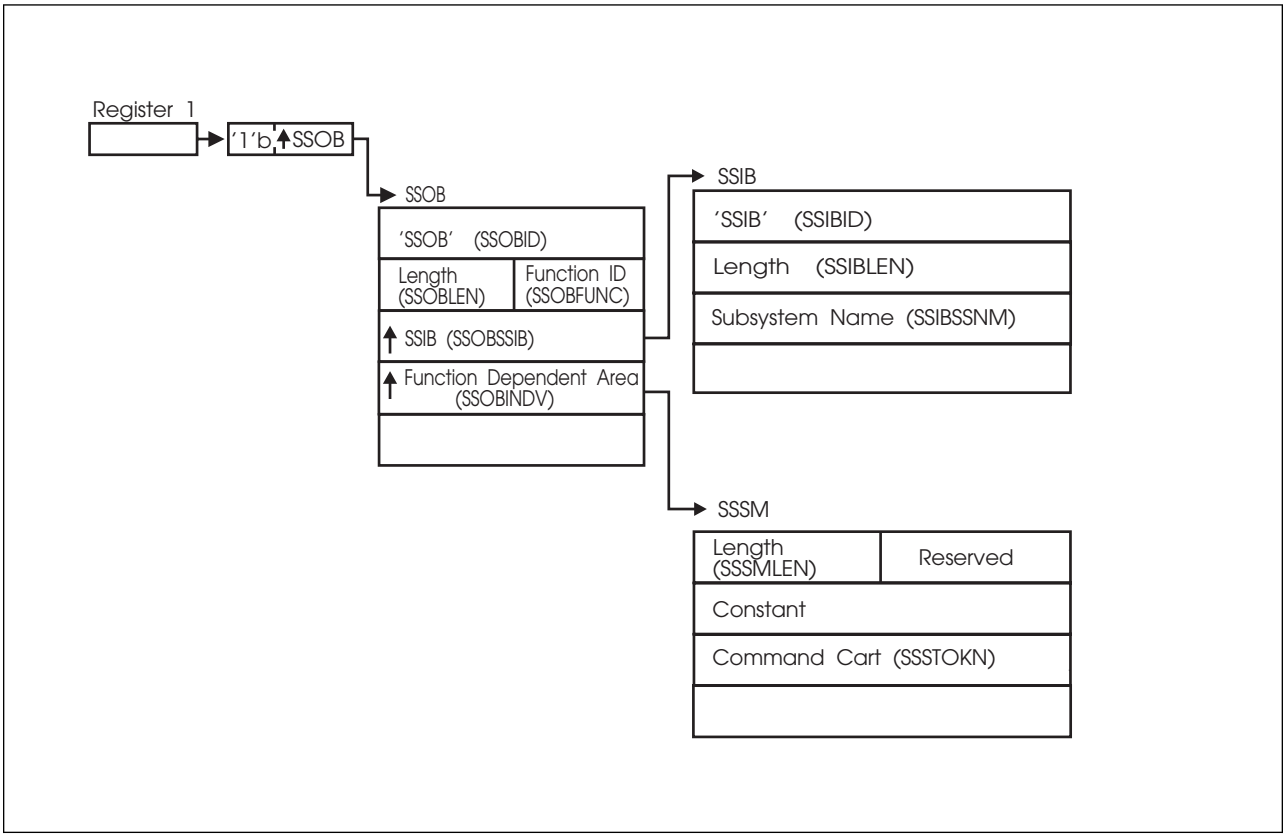

Figure 29. Environment at Time of Call for SSI Function Code 58

Input Register Information

On entry to the function routine the general purpose registers contain:

Register	Contents
0	Address of the SSCVT
1	Address of the SSOB control block
13	Address of a standard 18-word save area
14	Return address
15	Entry point address

On entry to the function routine the access registers are unused.

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSSM

SSOB Contents: SMF sets the following fields in the SSOB control block on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB control block
SSOBFUNC	SSI function code 58 (SSOBSMAC)
SSOBSSIB	Address of SSIB control block
SSOBINDV	Address of the function dependent area (SSSM control block)

SSIB Contents: SMF sets the following fields in the SSIB control block on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB control block
SSIBSSNM	Subsystem name — name of the subsystem that this SMF SUBPARM Option Change call is directed to.

SSSM Contents: SMF sets the following fields in the SSSM control block on input:

Field Name	Description
SSSMLEN	Length of the SSSM control block
SSSMFLGS	Flags • **SSSMSMFA** — SMF is active The following flags identify the source of the SUBPARM parameter value for the subsystem: • **SSSMMEMB** — Value from the parmlib member • **SSSMRPLY** — Value from the operator reply • **SSSMDFLT** — Value from the default table • **SSSMCONF** — Value changed due to conflicts • **SSSMCHNG** — Value changed by IPL or SET processing

You can use the following fields to communicate with the console that issued the SET SMF=xx or SETSMF command being processed:

SSSMCNID Command console ID
SSSTOKN Command CART

Output Register Information

Upon exit from the function routine, the general purpose registers must contain:

Register	Contents
0 — 12	Restored to contents on entry
14	Return address

15 Return code

Return Code Information

Upon return to the caller of SSI function code 58 (MVS or SMF), register 15 contains the smallest return code from the SSI and SSOBRETN contains the largest return code associated with the smallest return code from the SSI.

For MVS to process broadcast functions properly, you must use the following return code conventions for function routines that handle broadcast calls. When a routine returns control to the SSI:

- Set register 15 to 0.
- Set the SSOBRETN field in the SSOB control block to one of the following:

Return Code (Decimal)	Meaning
0	The function routine recognized the request but did not process it.
4	The function routine recognized the request and processed it.

Restrictions

The SMF SUBPARM Option Change call cannot be made to subsystems with the following names:

- SYS
- JES2
- JES3
- STC
- TSO
- ASCH.

Example

See “Passing Accounting Parameters to Your Subsystem” on page 164 for an example of the use of the SMF SUBPARM Option Change call.

Installation Supplied Subsystem

See “Passing Accounting Parameters to Your Subsystem” on page 164 for an example of the relationship of this function code to the options specified in the SMF parmlib member.

Tape Device Selection Call — SSI Function Code 78

The Tape Device Selection call (SSI function code 78) allows the subsystem function routine to receive control at least once for each job step JCL request or dynamic allocation invocation for a tape device. The function routine can then change the criteria the system uses when it selects tape devices to allocate.

Type of Request

Broadcast SSI call.

Use Information

Use SSI function code 78 to allow a subsystem to get control to influence the criteria the system uses in selecting the tape devices to allocate.

Issued to

- All active subsystems that indicate they support the Tape Device Selection call (SSI function code 78).

Related SSI Codes

None.

Related Concepts

You should understand the process the system uses to select the tape devices to be allocated. The following steps describe how the system processes the tape requests for each job step:

1. The system initializes fields in the tape allocation subsystem interface mapping (IEFSSTA, called SSTA in this section). The SSTA mapping consists of:
   - An SSTA header (one for each jobstep) that contains general information about the jobstep
   - A DD section (one for each DD statement or dynamic allocation request requiring a non-SMS managed tape device) that contains information about the DD
   - A device request section (one for each device indicated on the DD statement) that contains information about the tape device request
   - An eligible device array entry (one for each eligible device) that contains selection criteria.

   In initializing the eligible device array entry, the system considers the following facts about the tape device requests and the characteristics of available devices:
   - The type of requests (such as a request for a private, scratch, or specific volume)
   - Unit information on the requests

     The system uses the eligible device table (EDT) to determine which devices are eligible to satisfy the request.
   - Characteristics of each eligible tape device, such as:
     - Does the device already have the requested volume mounted
     - Is the device online or offline
     - Is the device dedicated or automatically switchable.

These characteristics are reflected in bits in the SSTAIBMM field.

Several of the IBM eligibility bits are set based on whether a volume is already mounted on the device. The following helps you understand the conditions that can cause a volume to be already mounted on a device.

A volume may already be mounted for any one of the following conditions:

- A volume was premounted as the result of a MOUNT command issued by the operator
- A volume was inserted into the drive by the operator, but no MOUNT command was issued by the operator
- A volume is mounted on a drive because a prior step in the same job passed a data set to a subsequent step or the request specified RETAIN
- A volume is mounted on a drive because it is in use by another job

Within an eligible device array entry, the order of the characteristics reflects their relative importance. For example, whether a specific device is mounted is more important than whether the device is automatically switchable.

The system then builds a list of eligible tape devices and associated eligibility values generated from bits in the SSTAIBMM field in the eligible device array entry.

At this point, the system issues SSI function code 78, passes the SSTA (including the eligible device array), and gives your Tape Device Selection function routine a chance to affect the selection. When the function routine gets control, it can set bits in the SSTAUSRM field. If SSTAUSRM bits are set, the system generates eligibility values that combine SSTAUSRM settings and SSTAIBMM settings.

2. Based on the list of eligible devices and associated eligibility values built in step 1, the system selects the optimal device to allocate for the request.

Figure 30 shows the logical relationship between the system settings and the user settings in the eligible device array entry. The first column shows the 1-bit fields the system sets in SSTAIBMM; the second column shows the 1-bit fields the function routine can set in SSTAUSRM. The criteria are listed in order of importance, from top to bottom. For example, the most important criteria are:

- SSTAINEL, a user field that can remove the device from consideration
- SSTADMND, a system field that identifies the device as the one specified on the DD statement.

The table shows how the user criteria interleave with system criteria.

Figure 30 (Page 1 of 2). Relationship between System and User Criteria

Importance	System criteria (SSTAIBMM)	User criteria (SSTAUSRM)
1		SSTAINEL
2	SSTADMND	
3		SSTAUS01
4		SSTAUS02

Figure 30 (Page 2 of 2). Relationship between System and User Criteria

Importance	System criteria (SSTAIBMM)	User criteria (SSTAUSRM)
5	SSTAONUN	
6		SSTAUS03
7		SSTAUS04
8	SSTANAFH	
9		SSTAUS05
10		SSTAUS06
11	SSTASPCM	
12		SSTAUS07
13		SSTAUS08
14	Generic device type not specified by a bit	
15		SSTAUS09
16		SSTAUS10
17	SSTAACL1	
18		SSTAUS11
19		SSTAUS12
20	SSTAACL2	
21		SSTAUS13
22		SSTAUS14
23	SSTAACL3	
24		SSTAUS15
25		SSTAUS16
26	SSTAVOLM	
27		SSTAUS17
28		SSTAUS18
29	SSTANVOL	
30		SSTAUS19
31		SSTAUS20
32	SSTAWVOL	
33		SSTAUS21
34		SSTAUS22
35	SSTAAVOL	
36		SSTAUS23
37		SSTAUS24
38	SSTAANAS	
39		SSTAUS25
40		SSTAUS26

Descriptions of SSTAIBMM fields are found in “Input Parameters” on page 244; descriptions of SSTAUSRM fields are found in “Output Parameters” on page 248.

Environment

Review “Function Routines/Function Codes” on page 157, which describes both the general environment on entry to your function routine and other programming considerations that your function routine should take into account.

If you decide to set up your subsystem to handle tape device selection calls, make sure that your Tape Device Selection function routine is in place before you enable the subsystem to receive SSI function code 78. IBM recommends that you use the IEFSSVT macro to notify MVS that your subsystem should be given control only when tape selection calls are made. IEFSSVT macro services are available only to dynamic subsystems. Subsystems that are not dynamic can still use the IEFJSVEC service. See “Building the SSVT” on page 279 and “Enabling Your Subsystem for New Functions” on page 284 for more information.

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your function routine:

- CVT
- IEFJESCT
- IEFSSOBH
- IEFJSSIB
- IEFSSTA

The function routine receives control in the following environment:

Minimum authorization	Supervisor state with PSW key 1
Dispatchable unit mode	Task
AMODE	31-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	Serialization for allocation resources is held by Allocation
Control parameters	The SSIB, SSOB, and SSTA control blocks can reside either above or below 16 megabytes.
Recovery	The function routine must provide an ESTAE-type recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide*.

The following figures show the environment at the time of the call for SSI function code 78.

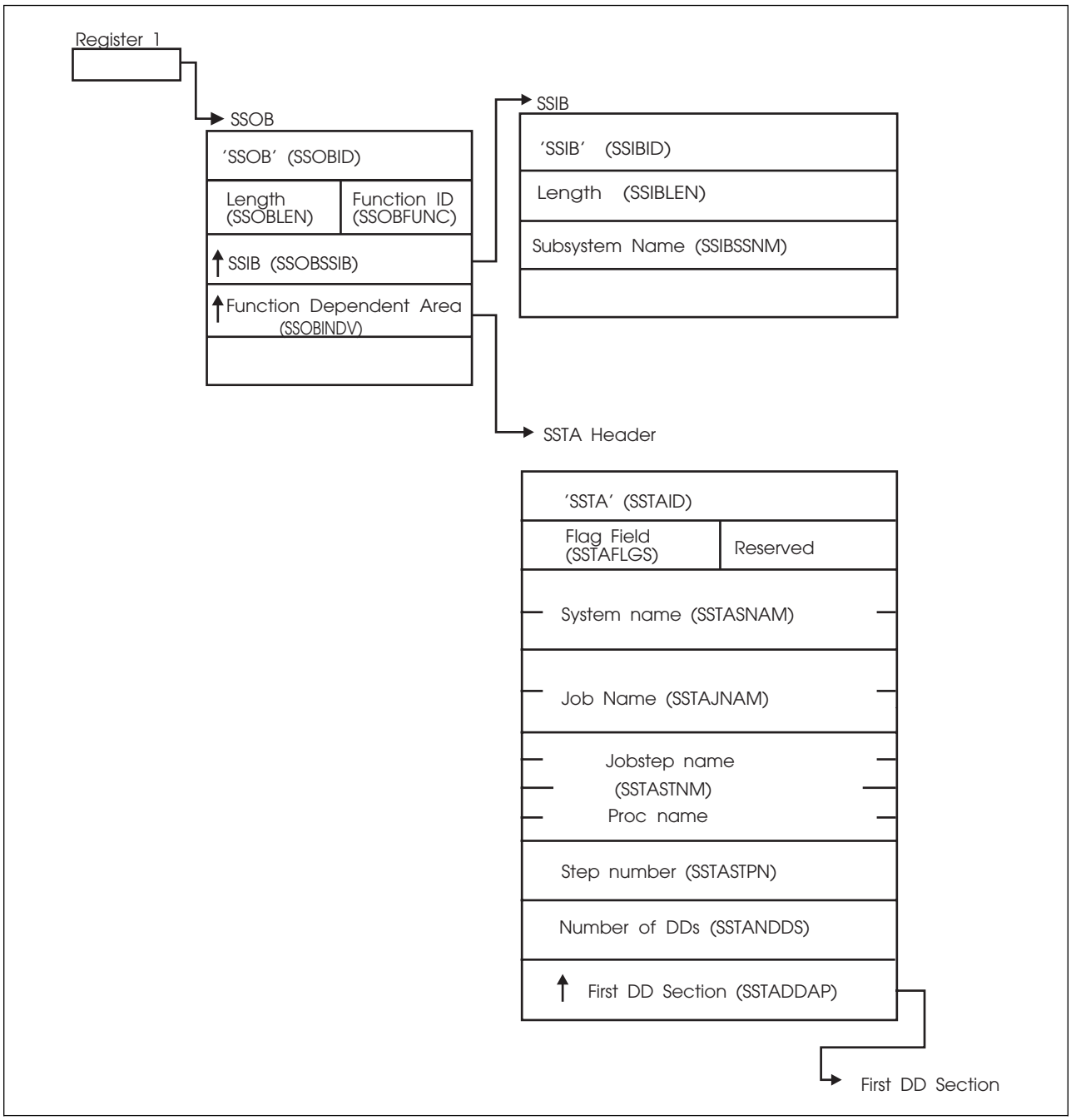

Figure 31. Environment at Time of Call for SSI Function Code 78

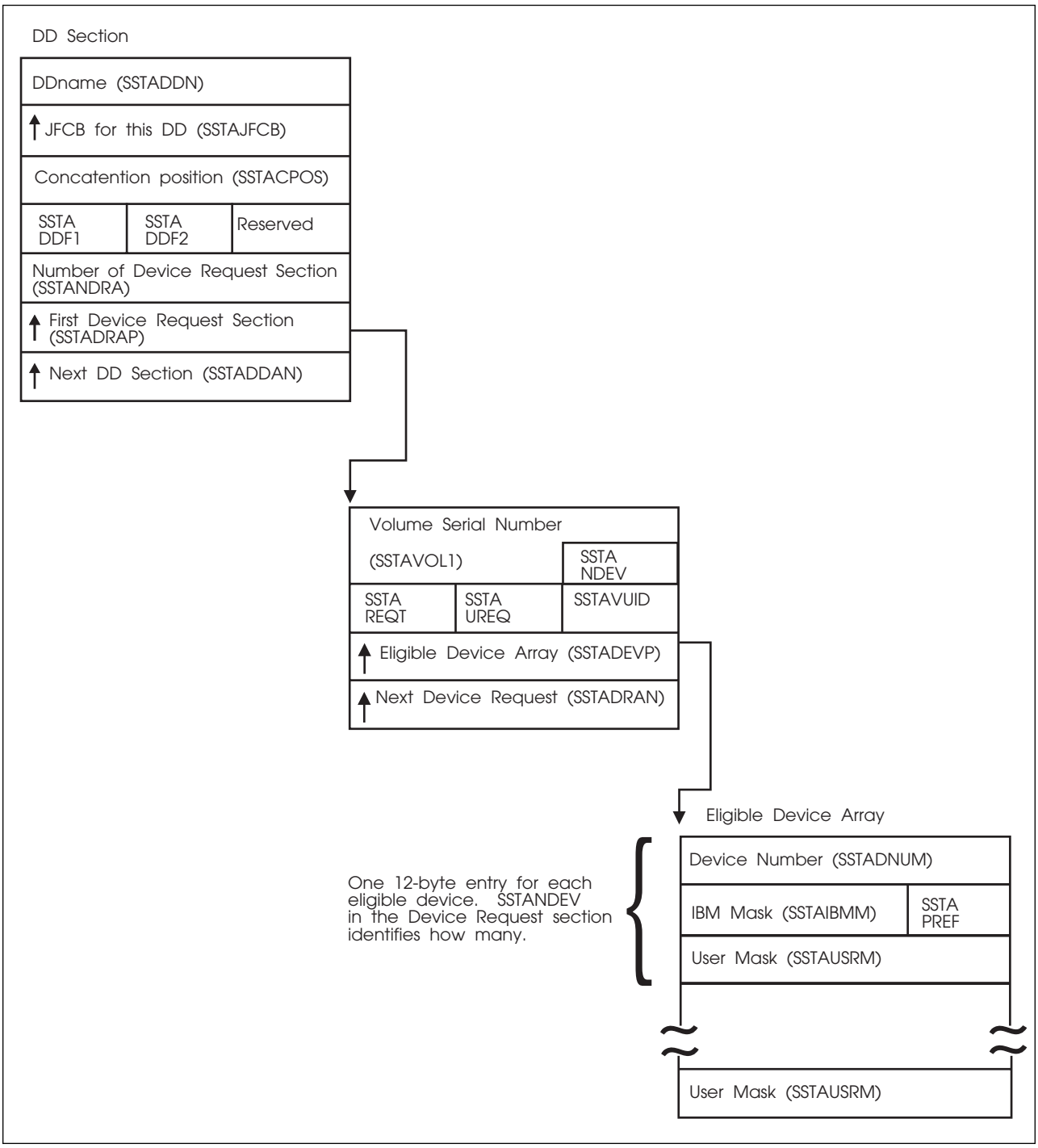

Figure 32. Continuation of Environment at Time of Call for SSI Function Code 78

Input Register Information

On entry to the function routine the general purpose registers contain:

Register	Contents
0	Address of the subsystem's SSCVT
1	Address of the SSOB
13	Address of a standard 18-word save area

14 Return address

15 Entry point address

Input Parameters

Input parameters for the function routine are:

- SSOB
- SSIB
- SSTA

SSOB Contents: MVS sets the following fields in the SSOB on input:

Field Name	Description
SSOBID	Identifier 'SSOB'
SSOBLEN	Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC	SSI function code 78 (SSOBTALC)
SSOBSSIB	Address of the SSIB control block
SSOBRETN	Return code value set by previously invoked function routine, or zero
SSOBINDV	Address of the function-dependent area (SSTA control block)

SSIB Contents: MVS sets the following fields in the SSIB on input:

Field Name	Description
SSIBID	Identifier 'SSIB'
SSIBLEN	Length of the SSIB (SSIBSIZE)
SSIBSSNM	Name of the subsystem enabled to receive this function code

SSTA Header Contents: There is one SSTA header for each job step or dynamic allocation that requests at least one non-SMS managed, non-DUMMY, non-SUBSYStem tape device. IBM sets the following fields on input:

Field Name	Description
SSTAID	Identifier 'SSTA'
SSTAVERS	Current SSTA version number
SSTAFLGS	Type of call, such as: • First call for this job step or dynamic allocation invocation • Call from allocation recovery • Call from tape allocation retry processing
SSTASNAM	System name
SSTAJNAM	Job name
SSTASTNM	Job step name or procedure name and job step name. If the job step is not a procedure step, SSTASTNM is an 8-byte job step name and an 8-byte reserved field. If the job step is a procedure step, SSTASTNM is an 8-byte procedure step name and an 8-byte job step name of the step that called the procedure.
SSTASTPN	Step number

SSTANDDS Number of DDs

SSTADDAP Pointer to the first DD array for this job step. Set to zero if no DD array entries exist.

SSTAHDRL Length of the SSTA header

DD Array Entry: There is one DD array entry for each DD statement or dynamic allocation that requests a non-SMS managed, non-DUMMY, non-SUBSYStem tape device. IBM sets the following fields on input:

Field Name	Description
SSTADDN	DD name. Blank if other than the first DD in a concatenation
SSTAJFCP	Pointer to the JFCB for this DD section
SSTACPOS	Concatenation position. Set to 1 for the first DD in a concatenation, 2 for the second DD, etc. Set to 1 for a DD that is not part of a concatenation
SSTADDF1	DD level information, including DISP and GDG specifications
SSTADDF2	DD level information byte 2, including unit affinity indicator
SSTANDRA	Number of devices requested
SSTADRAP	Pointer to the first device request section for this DD
SSTADDAN	Pointer to the next device request section
SSTADDAL	Length of one DD array entry

Device Request Array Entry: There is one device request array entry for each device or unit requested on a non-SMS managed, non-DUMMY, non-SUBSYStem DD statement or dynamic allocation reqest. For example, UNIT=(TAPE,2) would generate two device request array entries. IBM sets the following fields on input:

Field Name	Description
SSTAVOLI	Volume serial number. Relevant only for a specific request (indicated by bit SSTASPEC in field SSTAREQT)
SSTANDEV	Number of eligible devices
SSTAREQT	Device request information flags: • SSTAPRV — indicates a private request • SSTASPEC — indicates a specific request • SSTADEFR — indicates volume mounting is deferred
SSTAVUID	Volume unit ID for affinity
SSTADEVP	Pointer to the eligible device array for this DD
SSTADRAN	Pointer to the next device request array entry
SSTADRAL	Length of one device request array entry

The function routine can set the SSTAUDFR and SSTAUPRF fields in the device request section. See “Output Parameters” on page 248.

Eligible Device Array Entry: There is one eligible device array entry for each device eligible for a particular DD array entry. IBM sets the following fields on input:

Field Name	Description
SSTADNUM	Device number, in EBCDIC. Example: The representation of device number 5B0 would be F0F5C2F0. You can use this number as input to EDTINFO to obtain further information about the device, such as its generic device type and any esoteric service groups of which this device is a part. (See *OS/390 MVS Programming: Assembler Services Reference*, GC28-1910, for additional information about EDTINFO.)
SSTAIBMM	Following are the eligibility bits that the system sets. (Unless otherwise specified, the IBM eligibility bits apply to both dedicated and automatically switchable devices.)

Field Name	Description
SSTADSK	This device is skipped for this request
SSTADMND	This device is demanded by this request. (A request is a demand request when the UNIT parameter contains a specific device number, for example, UNIT=237.)
SSTAONUN	The device is online and unallocated
SSTANAFH	This automatically switchable device is not assigned to another system
SSTASPCM	The volume mounted on this device is the one requested
SSTAACL1	Either no automatic cartridge loader (ACL) is installed and this is a specific request, or the ACL is active and the request is nonspecific (public or private)
SSTAACL2	The installed ACL is inactive
SSTAACL3	Either the ACL is active and this is a specific request, or no ACL is installed and this is a nonspecific request (public or private) request
SSTAVOLM	One of the following conditions can occur: • A volume is not mounted on this device. This is a specific volume request. The device is automatically switchable and the last volume dismounted from this device is the needed volume. • A volume is mounted on this device. This is a non-specific request for a public volume and the volume currently mounted on this device is public.
SSTANVOL	A volume is not mounted on this device for one of these possible conditions: • This is a specific volume request. The device is automatically switchable and the last volume dismounted from this device is not the needed volume.

- This is a non-specific request.
- This device is not automatically switchable.

SSTAWVOL A volume is mounted on this automatically switchable device, and it matches the volume needed for this specific request. However, the last volume dismounted from this device also matches.

SSTAAVOL A volume is mounted on this device and one of the following conditions is true:

- This is a specific volume request and the volume currently mounted on this device is automatically switchable and the last volume dismounted from this device is not the needed volume.
- This is a specific volume request and there is a volume currently mounted on this device, but it is not the requested volume.
- This is a non-specific, private request for any volume.
- This is a non-specific, public request and the volume currently mounted is private.

SSTANAS This device is not automatically switchable

The function routine can set the SSTAPREF and SSTAUSRMM fields in the eligible device array entry. See “Output Parameters” on page 248.

Output Register Information

Upon exit from the function routine, the general purpose registers must contain:

Register	Contents
0	Used as a work register by the system
1	Address of the SSOB
2 — 13	Restored to contents on entry
14	Return address
15	Return code

Return Code Information

For MVS to process broadcast functions properly, you must use the following return code conventions for function routines that handle broadcast calls. When a routine returns control to the SSI:

- Set register 15 to 0.
- Set the SSOBRETN field in the SSOB to one of the following:

  Return Code

(Decimal)	Meaning
0	The function routine recognized the request but did not process it.
4	The function routine recognized the request and processed it.

Output Parameters

Output parameters for the function routine are:

Field Name	Description
SSTAUDFR	The field in device request section that forces a request to have mounting deferred until the dataset is actually opened
SSTAUPRF	The field in the device request section that indicates that the function routine is to override the actions the system takes if many devices have the same eligibility value. In other words, the system turns to this field to break a tie when more than one tape device has the same attributes. • If the function routine does not code this field, the system makes a random selection from among the devices with equal attributes. • If the function routine codes this field, it must tell the system, in the SSTAPREF entry for each eligible device, how to break a tie. With SP5.2 and any subsequent releases, allocation examines all eligible devices on an individual basis. Therefore, it is unlikely that the system will need a tie-breaker.
SSTAPREF	The field in the eligible device array entry that contains the preference value for the system to use. This field allows the function routine to influence the allocation of devices when all other attributes are the same. Use this field only if you set SSTAUPRF.
SSTAUSRM	The field in the eligible device array entry that allows the function routine to add its own criteria to the eligibility mask that the system associates with each eligible device:

Field Name	Description
SSTAINEL	Mark the device ineligible
SSTAUSnn	The remaining 1-bit fields, SSTAUS01 through SSTAUS26, can be defined and set by your function routine. Figure 30 on page 239 shows how each of these bits relates to the system mask SSTAIBMM.

SSTAEDAL	Length of one device request array entry

Restrictions

SSI function code 78 is not available to change the selection of SMS-managed or JES3-managed tape devices

Note that while MVS allocation processes your function routine, it is not processing other allocation requests. This might degrade performance.

Example

An installation writes a Tape Device Selection function routine to ensure that tape devices 270 and 271 are available only for HSM tape requests. (This example is included in SYS1.SAMPLIB as member IEFTASSI.)

```
TAPESSI  CSECT
TAPESSI  AMODE  31
TAPESSI  RMODE  ANY
****************** START OF SPECIFICATIONS ***************************
*                                                                     *
*01*   NAME=                                                         *
*                                                                     *
*01*   TYPE= Sample Subsystem                                        *
*                                                                     *
*01*   FIRST ELIGIBLE PRODUCT= HBB5520                               *
*                                                                     *
*01*   FIRST INELIGIBLE PRODUCT= HBB5510                             *
*                                                                     *
*01*   OPERATION=                                                    *
*       This is a sample taple allocation subsystem. It will         *
*       reserve devices 270 and 271 for only HSM jobs.                *
*                                                                     *
*       1.  Chain save areas                                          *
*       2.  See if the JOBNAME is HSM*                                *
*       3.  If it is not then will ensure that                        *
*           devices 270 and 271 are not eligible                      *
*       4.  Return to SSI                                             *
*                                                                     *
*03*    SOFTWARE DEPENDENCIES:                                       *
*                                                                     *
*04*           REQUIRED PRODUCTS= HBB5520                            *
*                                                                     *
*02*    OUTPUT:                                                      *
*                                                                     *
*03*           MSGIDS= NONE                                          *
*                                                                     *
*03*           ABENDCODES=  NONE                                     *
*                                                                     *
******************* END OF SPECIFICATIONS ****************************
```

```
*********************************************************************
*                                                                   *
*  Base register:  12                                               *
*                                                                   *
*  Other register use:                                              *
*                  10   SSCVT                                       *
*                   2   SSOB                                        *
*                   9   SSIB                                        *
*                   8   SSTA                                        *
*                                                                   *
*  Attributes:                                                      *
*     This routine must be reentrant and reside in a library        *
*     accessible at the time subsystem initialization occurs.       *
*     Supervisor state, AMODE(31), RMODE(ANY)                       *
*                                                                   *
*********************************************************************

*********************************************************************
* Chain saveareas                                                   *
*********************************************************************
         USING TAPESSI,12
         SAVE  (14,12)                Save caller registers
         LR    12,15                  Establish module base register
*
         LR    10,0                   Establish addressability
         USING SSCT,10                 to the SSCVT
         LR    2,1                    Establish addressability
         USING SSOB,2                  to the SSOB
*
         GETMAIN R,LV=84,SP=230       Get working storage
         ST    13,4(1)                Chain saveareas forward
         ST    1,8(13)                Chain saveareas backward
         LR    13,1                   Point to this module's savearea
         LR    11,1                   Point to dynamic storage
         USING DYNAM,11               Base dynamic storage
*
*********************************************************************
* Validate the request                                              *
*********************************************************************
         L     9,SSOBSSIB             Establish addressability
         USING SSIB,9                  to the SSIB
         CLC   SSIBSSNM,SSCTSNAM      Verify the subsystem name
         BNE   ERROR                  This should never happen
         L     8,SSOBINDV             Pointer to function dependent
*                                       area
*********************************************************************
* Check for job name beginning with HSM                             *
*********************************************************************
         USING SSTA,8                 Set basing
         CLC   HSMNAME,SSTAJNAM       Check job name
         BE    NOCHECK                Skip checks if not HSM*
```

```
*********************************************************************
* Job name does not begin with HSM so must not allow devices        *
* 0270 and 0271 to be eligible to satisfy request.                  *
*********************************************************************
         L     7,SSTADDAP              Get address of DD entries
         USING SSTADDA,7               Base DD entries
         L     4,SSTANDDS              Get number of DDs
         LTR   4,4                     Check for zero
         BZ    NOCHECK                 If zero then no DDs to check
         ST    4,NUMDDS                Else save in local storage
DDLOOPS  EQU   *                       Start looping through DDs
         L     6,SSTADRAP              Get address of first request
         USING SSTADRA,6               Base request entries
         L     4,SSTANDRA              Get number of requests
         LTR   4,4                     Check for zero
         BZ    DDLOOPE                 If zero then no requests
         ST    4,NUMREQS               Else save in local storage
REQLOOPS EQU   *                       Start looping through requests
         L     5,SSTADEVP              Get address of first device
         USING SSTAEDA,5               Base device entries
         LH    4,SSTANDEV              Get number of devices eligible
         LTR   4,4                     Check for zero
         BZ    REQLOOPE                If zero then no devices
         ST    4,NUMDEVS               Else save in local storage
```

```
*********************************************************************
* Check each eligible device entry to make sure that devices        *
* 0270 and 0271 are not eligible to this request.                   *
*********************************************************************
DEVLOOPS EQU   *                       Start looping through devices
         CLC   SSTADNUM,HSMDEV1        Is device reserved for HSM?
         BE    MAKEINEL                Yes, go make ineligible
         CLC   SSTADNUM,HSMDEV2       Is device reserved for HSM?
         BNE   DEVLOOPE                No, bypass making ineligible
MAKEINEL EQU   *
         OI    SSTAUSE1,B'10000000'    Mark device ineligible
DEVLOOPE EQU   *                       End of eligible device loop
         LA    3,12                    Get size of SSTAEDA entry
         ALR   5,3                     Add to pointer to get next
         L     4,NUMDEVS               Get local counter
         LA    3,1                     Get amount to decrement count
         SLR   4,3                     Decrement count
         ST    4,NUMDEVS               Save device count
         LTR   4,4                     Check device count
         BNZ   DEVLOOPS                Loop back if more to process
REQLOOPE EQU   *                       End of request loop
         L     6,SSTADRAN              Get address of next request
         L     4,NUMREQS               Get local counter
         LA    3,1                     Get amount to decrement count
         SLR   4,3                     Decrement count
         ST    4,NUMREQS               Save request count
         LTR   4,4                     Check request count
         BNZ   REQLOOPS                Loop back if more to process
DDLOOPE  EQU   *                       End of DD loop
         L     7,SSTADDAN              Get address of next DD entry
         L     4,NUMDDS                Get local counter
         LA    3,1                     Get amount to decrement count
         SLR   4,3                     Decrement count
         ST    4,NUMDD                 Save DD count
         LTR   4,4                     Check DD count
         BNZ   DDLOOPS                 Loop back if more to process
NOCHECK  EQU   *
         MVC   SSOBRETN,=F'0'          Indicate function success
         B     RETURN
ERROR    EQU   *
         MVC   SSOBRETN,=F'20'         Indicate function failure
*********************************************************************
```

```
*********************************************************************
* Return to the SSI                                                 *
*********************************************************************
RETURN   EQU   *
         L     8,4(13)                 Pointer to caller's savearea
         FREEMAIN R,LV=84,A=(13),SP=230
         LR    13,8
         LM    14,12,12(13)            Restore caller' registers
         LA    15,0                    RC=0
         BSM   0,14                    Return to the SSI
*
HSMNAME  DC    CL3'HSM'                HSM Jobname
HSMDEV1  DC    CL4H'270'                  Device reserved for HSM
HSMDEV2  DC    CL4H'271'                  Device reserved for HSM
*
DYNAM    DSECT                         Dynamic storage
SAVEAREA DS    18F                     Module save area
NUMDDS   DS    F                       Number of DDs
NUMREQS  DS    F                       Number of requests
NUMDEVS  DS    F                       Number of eligible devices
*
         IEFJSCVT
         IEFSSOBH
         IEFJSSIB
         IEFSSTA
         END
```

Troubleshooting Errors in Your Subsystem

This chapter describes common types of errors that occur when you are using subsystems, and includes steps you can take to troubleshoot these errors. Errors can occur when you are:

- Defining your subsystem to MVS
- Processing a subsystem function request

Handling Initialization Errors

If you specified a suffix on the SSN system parameter and it does not exist, the system issues the following message:

```
IEF758I   SUBSYSTEM AVAILABILITY LIMITED
          DESCRIPTION NOT FOUND IN SYS1.PARMLIB
```

For an IPL, do not define a subsystem more than once in a combination of IEFSSNxx members that can be used together or within a single member. (The same subsystem can appear in two different IEFSSNxx members when the members will not be used together.) If MVS detects a duplicate name, the duplicate subsystem is not defined and its initialization routine does not receive control. The system issues the following message:

```
IEFJ003I  DUPLICATE SUBSYSTEM subname NOT INITIALIZED
```

If you specified an initialization routine (yyyyyyyy) in IEFSSNxx but the system could not locate the initialization routine, the system issues the following message:

```
IEFJ004I   SUBSYSTEM subname NOT INITIALIZED - initrtn NOT FOUND
```

If you get this message, the subsystem will be defined to the system but not initialized, so jobs which require the functions of this subsystem may fail.

If you specify an initialization routine in IEFSSNxx but an abend occurs in the initialization routine (while the system was initializing the subsystem), the system issues the following message:

```
IEFJ005I   subname INITIALIZATION ROUTINE initrtn ABENDED
```

If you get this message, examine the DUMP data set to find which subsystem initialization routine failed. If the abend occurred during the processing of an initialization routine specified in IEFSSNxx, a dump is requested only if the initialization routine does not request one first. If you are coding an initialization routine, you should provide recovery and consider whether you want a dump if a problem occurs.

If problems occur when the system tries to obtain storage to build control blocks for a subsystem, the system issues the following message:

```
IEFJ006I   subname SUBSYSTEM UNAVAILABLE, INSUFFICIENT STORAGE
```

If you get this message, see *OS/390 MVS System Messages, Vol 4 (IEC-IFD)* for more information.

If an abend occurred while the system was initializing a subsystem and the system requests a dump, the system issues the following message:

```
IEFJ007I   A SYSTEM ERROR HAS OCCURRED DURING INITIALIZATION OF
           SUBSYSTEM subname
```

If you get this message, examine the DUMP data set to identify the problem.

If an incorrect keyword is found in IEFSSNxx, the following message is written:

```
IEFJ001I   memname LINE line-number:  ERROR IN SUBSYSTEM DEFINITION,
           REFER TO HARDCOPY LOG
```

If you get this message, the system continues processing the rest of IEFSSNxx, and you should correct the keyword indicated. The system does not process the subsystem definition containing the incorrect keyword.

Diagnosis, Modification or Tuning Information

Handling Function Request Errors

When you are troubleshooting errors during SSI function request processing, do the following:

- Capture the system dump
- Identify the type of error
- Determine the cause of the error.

Capturing the System Dump

If an abend occurs while processing a subsystem function request, the SSI requests a dump (unless a subsystem function routine takes one first). The dump title is similar to the following:

```
TITLE=COMPON=SSI,COMPID=5752SC1B6,ISSUER=IEFJSaaa,
MODULE=IEFJbbbb,ABEND=xxx,REASON=yyyyyyyy DUMP
```

The issuer is one of the following:

- IEFJSARR, if the caller of the SSI is in task mode and holds no locks
- IEFJSFRR, if the caller of the SSI is in SRB mode or holds a lock
- IEFJSPCE, if the error is a recursive failure in the SSIs recovery.

For function request errors, the module is one of the following:

- IEFJRASP, for broadcast function requests
- IEFJSRE1, for directed function requests or for broadcast function requests that have not yet been passed to IEFJRASP.

Other module names may appear for errors in SSI services other than routing function requests.

Another variation of the dump title is the following:

```
DUMP TITLE=COMPON=SSI,COMPID=5752SC1B6,ISSUER=IEFJSaaa,
MODULE=IEFJbbbb,ABEND=xxx,REASON=yyyyyyyy,SNAME=zzzz
```

This variation will appear when SSI has determined that the error occurred in a subsystem function routine. The dump title identifies the name of the failing subsystem. SNAME refers to the subsystem, while zzzz is the name of the subsystem.

After creating a subsystem vector table, the SSI retains only the addresses of the function routines represented in the table, and therefore cannot identify the failing routine by name.

The dump title indicates an SSI routine as the failing CSECT, even when the error occurred in a subsystem function routine. After creating a subsystem vector table, the SSI retains only the address of the function routines represented in the table, and therefore cannot identify the failing routine by name.

Identifying the Type of Error

The most common causes of errors while processing function requests are:

- Function routine error
- Function routine address that is not valid
- Vector table address that is not valid
- Control block chain that is not valid
- Parameter list passed to the SSI that is not addressable
- SSI error

Identifying the Problem Type when the VRA is Available

You can identify the type of error when you examine the variable recording area (VRA) in the summary dump or in the output from EREP. The available information may include:

- A footprint area that contains a set of footprints and pointers describing the status of the SSI request
- An English translation of the footprints
- The address of the SSOB control block describing the request
- The address of the SSIB control block identifying the subsystem to which the request is directed
- The address of the SSCVT associated with the target subsystem
- The address of the active SSVT being used by the target subsystem to route function requests
- The address of the target subsystem function routine
- The name of the failing IEFJFRQ exit routine
- The return address of the SSI's caller

The actual information may vary, depending on the type and location of the error.

The English translation of the footprints identifies the point at which the error occurred, and may include one of the following:

- Abend in the function routine

  The error occurred when the SSI transferred control to the subsystem function routine. The error is probably due to one of the following:

  – The function routine address in the subsystem vector table is not valid

  – The function routine failed. In this case, either the function routine did not establish its own recovery, or it percolated to the SSI's recovery.

- Abend in IEFJFRQ routine

  The error occurred in an exit routine associated with the IEFJFRQ exit point. The VRA contains the name of the failing exit routine.

- Error referencing the SSVT

  The error occurred when the SSI tried to reference an SSVT control block that was not SSI-managed, but that was being used by the subsystem to route its requests.

- Error referencing the SSCVT

  The error occurred when the SSI tried to reference the SSCVT describing the target subsystem. The target subsystem is either not dynamic, or is dynamic but is not using an SSI-managed SSVT control block to route function requests.

- Error locating the subsystem

  The error occurred when the SSI tried to locate system control blocks associated with the target subsystem.

- Error validating the request

  The error occurred when the SSI tried to validate the SSOB/SSIB control block chain describing the function request.

Contact the IBM Support Center for any other footprints that you may receive.

Identifying Problem Type when the VRA is not Available

You can identify the type of error when the VRA is not available by checking the PSW and the registers at the time of the error as follows:

- If the PSW equals register 15, it probably indicates that the subsystem function routine address in the SSVT is not valid.
- If the PSW contains a valid address in a module other than IEFJSRE1 or IEFJRASP, it is probably a subsystem function routine error. The error occurred in this routine.
- If the PSW contains a valid address in IEFJSRE1 or IEFJRASP, the error occurred while referencing subsystem related control blocks, the input parameter list, or in the SSI. Examine the SSCVT chain pointed to by the JESSSCT field for pointers that are not valid. The SSIDATA IPCS subcommand displays the subsystems defined to the SSI based on this chain, and may help identify a problem. See *OS/390 MVS IPCS Commands* or *OS/390 MVS Diagnosis: Reference* for more information.

Determining the Cause of the Error

You can determine the cause of the error by collecting the following information:

- Identity of the failing subsystem (or subsystem targeted by the request)
- Identity of the subsystem function requested
- Identity of the subsystem function routine
- Identity of the caller of the SSI
- Identity of the failing IEFJFRQ exit routine (if applicable)

Identifying the Failing Subsystem

The SSIBSSNM field of the SSIB control block identifies the subsystem targeted by the current SSI request. The VRA contains the address of the SSOB control block used to route the current request, and also contains the address of the SSIB if the error did not occur while validating the SSOB control block chain. Note that the SSIB and SSOB control blocks pointed to by the VRA may be copies of the control blocks originally provided by the SSIs caller, and may contain information other than what was provided in the original control blocks. The VRA contains the address of the SSOB control block, and the SSOBSSIB field of the SSOB control block locates the SSIB control block. The SSOBINDV field, if non-zero, points to the SSOB extension originally provided by the caller.

You can also use the current SSCVT to identify the current subsystem. If the address of the SSCVT appears in the VRA, the SSCTSNAM field identifies the subsystem.

If the footprints indicate that the error occurred while locating the target subsystem, and the SSI was processing a broadcast request, the VRA identifies the last successfully processed subsystem. The VRA section with the header 'LAST PROCESSED SSCVT', lists the address of the last subsystem to which the current request was successfully routed. Subsystems receive broadcast requests in the order in which they appear in the SSCVT chain (anchored by the JESSSCT field of the JESCT data area). The failing subsystem should be the next one in the SSCVT chain.

Identifying the Requested Subsystem Function

To identify the requested subsystem function, check the SSOBFUNC field of the SSOB control block. If the function code is not discussed in “SSI Function Codes Your Program Can Request” on page 13 or “SSI Function Codes Your Subsystem Can Support” on page 185, you may be able to identify the function request type by checking the SSOB extension pointed to by the SSOBINDV field. If the extension contains an eyecatcher, the format is normally SSxx, and the mapping macro for the extension is IEFSSxx. The mapping macro defines the value contained in the SSOBFUNC field, and describes the SSOB extension.

Identifying the Subsystem Function Routine

To identify the subsystem function routine, check the VRA. It contains the address of the failing routine. Identify the failing function routine by browsing backward in storage to find an eyecatcher. The information in the eyecatcher should also help identify the product with which the failing subsystem and function routine are associated.

Note: The high-order bit of the function routine address in the VRA or SSVT indicates the AMODE in which the routine receives control. When the high-order bit is set, the SSI passes control to the function routine in AMODE 31.

Identifying the Caller of the SSI

To identify the caller of the SSI, check the VRA. It contains the return address of the invoker of the IEFSSREQ macro (the caller of the SSI).

If the VRA is not available, locate the linkage stack associated with the work unit that was in control at the time of the error, and use the IPCS linkage stack formatting support to analyze the entries. The PSW from the current linkage stack

entry is the caller's return address (assuming that the subsystem function routine did not issue any instructions that caused additional linkage stack entries).

Browse backward through storage from the PSW address to find an eyecatcher and identify the caller.

Identifying the Failing Exit Routine

To identify the failing exit routine, check the VRA. It contains the name of the routine if the error occurred in an IEFJFRQ exit routine. Search for the module name in the dump or review IBM or vendor product documentation to identify the product or application with which it is associated. If the failing exit routine is associated with a vendor product, contact the vendor to determine the cause of the error.

End of Diagnosis, Modification or Tuning Information

Appendix A. Examples — Subsystem Interface Routines

This appendix has the following coding examples for the TSYS sample subsystem.

- "Example 1 — Subsystem Initialization Routine (TSYSINIT)"

  This example documents Product-Sensitive Programming Interfaces and Associated Guidance Information.

- "Example 2 — Subsystem Function Routine (WRITEIT)" on page 268

  This example documents General-Use Programming Interfaces and Associated Guidance Information.

- "Example 3 — Subsystem Function Routine (DELETEIT)" on page 270

  This example documents General-Use Programming Interfaces and Associated Guidance Information.

- "Example 4 — Subsystem Function Routine (LISTEN)" on page 272

  This example documents Product-Sensitive Programming Interfaces and Associated Guidance Information.

- "Example 5 — Subsystem Requesting Routine (TSYSCALL)" on page 274

  This example documents General-Use Programming Interfaces and Associated Guidance Information.

See Introduction to Subsystems and the Subsystem Interface (SSI), Setting Up Your Subsystem, and Making a Request of a Subsystem for information on coding subsystem routines.

Example 1 — Subsystem Initialization Routine (TSYSINIT)

```
TSYSINIT RSECT
TSYSINIT AMODE ANY
TSYSINIT RMODE ANY
*********************************************************************
*  Function:                                                        *
*   This is the TSYS subsystem initialization routine.  It is       *
*   called as the result of subsystem definition in any of the      *
*   following ways:                                                 *
*                                                                   *
*     IEFSSNxx parmlib member                                       *
*     SETSSI ADD command                                            *
*     IEFSSI REQUEST=ADD macro                                      *
*                                                                   *
*   Initialization for the TSYS subsystem consists of the following *
*   steps:                                                          *
*                                                                   *
*     1. Establish recovery                                         *
*     2. Issue the IEFSSVT REQUEST=CREATE macro to create the       *
*        subsystem vector table                                     *
*     3. Issue the IEFSSI REQUEST=OPTIONS macro to specify          *
*        optional information specific to the TSYS subsystem        *
*     4. Issue the IEFSSI REQUEST=PUT macro to store information    *
*        for use by the TSYS subsystem function routines            *
*     5. Issue the IEFSSI REQUEST=ACTIVATE macro to enable the      *
*        TSYS subsystem to receive function requests                *
*     6. Cancel recovery and return                                 *
*                                                                   *
*   INPUT                                                           *
*     Register 1 points to a two-word parameter list                *
*     - Word 1 = address of the SSCVT for the TSYS subsystem        *
*     - Word 2 = address of the JSIPL                               *
*                                                                   *
*   REGISTER USE                                                    *
*      1 - TSYSCB                                                   *
*     10 - SSCVT                                                    *
*     11 - JSIPL                                                    *
*     12 - Code register                                            *
*     13 - Data register                                            *
*                                                                   *
*   MACROS                                                          *
*     CVT                                                           *
*     ESTAE                                                         *
*     FREEMAIN                                                      *
*     GETMAIN                                                       *
*     IHASDWA                                                       *
*     IEFJESCT                                                      *
*     IEFJSCVT                                                      *
*     IEFSSI                                                        *
*     IEFSSVT                                                       *
*     IEFSSVTI                                                      *
*     RETURN                                                        *
*     SETRP                                                         *
*     WTO                                                           *
*                                                                   *
*********************************************************************
```

```
*
*********************************************************************
* Chain saveareas.                                                  *
*********************************************************************
         USING TSYSINIT,12
         SAVE  (14,12)                 Save caller's registers
         LR    12,15                   Establish module base register
         LR    10,1                    Save pointer to parameter list
         GETMAIN R,LV=WORKALEN         Get working storage
         ST    13,4(1)                 Chain saveareas backward
         ST    1,8(13)                 Chain saveareas forward
         LR    13,1                    Point to this module's savearea
*
         USING WORKAREA,13             Addressability to work area
         L     11,4(10)                Establish addressability
         USING JSIPL,11                 to the JSIPL
         L     10,0(10)                Establish addressability
         USING SSCT,10                  to the SSCVT
*
*********************************************************************
* Establish ESTAE                                                   *
*********************************************************************
         XC    ESTAED,ESTAED           Clear ESTAE parameter list
         L     8,=A(TSYSERR)           Address of ESTAE routine
         ESTAE (8),CT,PARAM=ARETRY,MF=(E,ESTAED)
         LTR   15,15                   If ESTAE failed
         BNZ   ESTAERR                  report it and return
*
*********************************************************************
* Invoke the IEFSSVT REQUEST(CREATE) macro to build and initialize  *
* the vector table, using the static function routine input table.  *
* The function routines reside in LINKLIB and must be loaded to     *
* global storage to make them available to all address spaces.      *
* Register notation is used to identify the output token for        *
* demonstration purposes.                                           *
*********************************************************************
         LA 2,TOKEN1
*
         IEFSSVT REQUEST=CREATE,SUBNAME=SSCTSNAM,SSVTDATA=ROUTINE1,    *
               OUTTOKEN=(2),LOADTOGLOBAL=YES,MAXENTRIES=ENTRIES,       +
               RETCODE=RC,RSNCODE=REASON,                              +
               MF=(E,VTPARMS)
*
         B     TESTVTCR(15)            Check return code
*
TESTVTCR EQU   *
         B     ANCHORCB                 0 - Processing successful
         B     VTERR                    4 - Warning
         B     VTERR                    8 - Invalid parameters
         B     VTERR                   12 - Request failure
         B     VTERR                   16 - Error loading subsystem
         B     VTERR                   20 - System error
         B     VTERR                   24 - SSI service not available
*
ANCHORCB EQU   *                       Entry for vector table created
```

```
*********************************************************************
* Initialize and anchor the subsystem-specific control block used   *
* by TSYS and its function routines.                                *
*********************************************************************
         GETMAIN R,LV=CBLEN,SP=241     Get storage for TSYS control    +
                                         block
         USING TSYSCB,1
         XC    TSYSCB,TSYSCB           Clear control block
         MVC   TSYSID(4),CBACRO        Move in eye-catcher
         LA    7,1                     Version 1
         STH   7,TSYSVER               Put version number in control   +
                                         block
         LA    7,CBLEN                 Get control block length
         STH   7,TSYSLEN               Put length in control block
         ST    1,CBADDR                Save control block address
         DROP  1
*
         IEFSSI REQUEST=PUT,SUBNAME=SSCTSNAM,SUBDATA1=CBADDR,          +
               RETCODE=RC,RSNCODE=REASON,                              +
               MF=(E,SSIPARMS)
*
         B     TESTPUT(15)             Check return code
*
TESTPUT  EQU   *
         B     OPTIONS                 0 - Processing successful
         B     SSIERR                  4 - Warning
         B     SSIERR                  8 - Invalid parameters
         B     SSIERR                 12 - Request failure
         B     SSIERR                 16 - Not defined
         B     SSIERR                 20 - System error
         B     SSIERR                 24 - SSI service not available
*
*********************************************************************
* Inform the SSI that TSYS will respond to the SETSSI command.      *
*********************************************************************
OPTIONS  EQU   *                       Entry for successful PUT
*
         IEFSSI REQUEST=OPTIONS,SUBNAME=SSCTSNAM,COMMAND=YES,          +
               RETCODE=RC,RSNCODE=REASON,                              +
               MF=(E,SSIPARMS)
*
         B     TESTOPT(15)             Check return code
*
TESTOPT  EQU   *
         B     ACTIVATE                0 - Processing successful
         B     SSIERR                  4 - Warning
         B     SSIERR                  8 - Invalid parameters
         B     SSIERR                 12 - Request failure
         B     SSIERR                 16 - Not defined
         B     SSIERR                 20 - System error
         B     SSIERR                 24 - SSI service not available
*
ACTIVATE EQU   *                       Entry for successful OPTIONS
```

```
*********************************************************************
* Activate the subsystem.                                           *
*********************************************************************
         IEFSSI REQUEST=ACTIVATE,SUBNAME=SSCTSNAM,INTOKEN=TOKEN1,      +
               RETCODE=RC,RSNCODE=REASON,                              +
               MF=(E,SSIPARMS)
*
         B     TESTACT(15)
*
TESTACT  EQU   *
         B     ACTIVEOK               0 - Processing successful
         B     SSIERR                 4 - Warning
         B     SSIERR                 8 - Invalid parameters
         B     SSIERR                12 - Request failed
         B     SSIERR                16 - Not defined
         B     SSIERR                20 - System error
         B     SSIERR                24 - SSI service not available
*
ACTIVEOK EQU   *
         WTO 'TSYS - SUBSYSTEM INITIALIZED'
         B     DONE
*
VTERR    EQU   *                      Entry for IEFSSVT error
         MVC   FAILSRV(L'SSVTSRV),SSVTSRV  Get name of failing service
         B     ERRMSG                 Issue error message
*
SSIERR   EQU   *                      Entry for IEFSSI error
         MVC   FAILSRV(L'SSISRV),SSISRV  Get name of failing service
*
*********************************************************************
* Convert the return and reason code and issue an error message.    *
*********************************************************************
ERRMSG   EQU   *
         MVC   SERVERRD(SERVMSGL),SERVERRS  Copy static message
*
         L     7,RC                   Get return code
         CVD   7,DOUBLE               Convert to decimal
         UNPK  RCODE1,DOUBLE          Make return code printable
         MVZ   RCODE1+3,RCODE1
         MVC   SERVERRD+43(2),RCODE1+2  Put return code in message
*
         L     7,REASON               Get reason code
         CVD   7,DOUBLE               Convert to decimal
         UNPK  RCODE1,DOUBLE          Make reason code printable
         MVZ   RCODE1+3,RCODE1
         MVC   SERVERRD+55(4),RCODE1   Put reason code in message
*
         MVC   SERVERRD+18(L'FAILSRV),FAILSRV  Put name of failing    ++
                                        service in message
         WTO   MF=(E,SERVERRD),CONSNAME=JSICNAME  Issue message
         B     DONE
```

```
*
INITERR  EQU   *
         MVC   INITERRD(INITMSGL),INITERRS  Copy static message
         WTO   MF=(E,INITERRD),CONSNAME=JSICNAME  Issue message
         B     DONE
*
ESTAERR  EQU   *
         MVC   ESTAERRD(ESTAMSGL),ESTAERRS  Copy static message
         WTO   MF=(E,ESTAERRD),CONSNAME=JSICNAME  Issue message
         B     RETURN
*
*********************************************************************
* Cancel the ESTAE and return to caller.                            *
*********************************************************************
DONE     EQU   *
         ESTAE 0
RETURN   EQU   *
         L     8,4(13)                 Pointer to caller's savearea
         FREEMAIN R,LV=WORKALEN,A=(13)
         LR    13,8
         RETURN (14,12),RC=0
*
*********************************************************************
* ESTAE routine.                                                    *
*********************************************************************
TSYSERR  EQU   *
         DROP  12                      Drop current addressability
         USING TSYSERR,15              Set addressability to TSYSERR
         LR    12,15                   Copy address of TSYSERR
         S     12,=A(TSYSERR-TSYSINIT) Reestablish code register
         DROP  15                      Drop addressability to TSYSERR
         USING TSYSINIT,12             Reset addressability
         CL    0,=F'12'                If no SDWA provided
         BE    TSYSERRA                Branch to percolate
         USING SDWA,1
         L     4,SDWAPARM
         L     4,0(4)
         DROP  1
         SETRP WKAREA=(1),RC=4,RETADDR=(4),FRESDWA=YES,RETREGS=YES
TSYSERRA EQU   *
         XR    15,15                   Indicate percolation
         BR    14
*
*********************************************************************
* Define static function routine input table.                       *
*********************************************************************
         IEFSSVTI TYPE=INITIAL,SSVTDATA=ROUTINE1,TABLEN=STABLEN
         IEFSSVTI TYPE=ENTRY,NUMFCODES=1,FCODES=254,FUNCNAME=WRITEIT
         IEFSSVTI TYPE=ENTRY,NUMFCODES=1,FCODES=255,FUNCNAME=DELETEIT
         IEFSSVTI TYPE=ENTRY,NUMFCODES=1,FCODES=9,FUNCNAME=LISTEN
         IEFSSVTI TYPE=FINAL
*
```

```
*********************************************************************
* Function routine data.                                            *
*********************************************************************
WRITEIT  DC    CL8'WRITEIT '
LISTEN   DC    CL8'LISTEN  '
DELETEIT DC    CL8'DELETEIT'
ENTRIES  DC    H'4'
SSVTSRV  DC    CL7'IEFSSVT'
SSISRV   DC    CL7'IEFSSI '
CBACRO   DC    CL4'TSCB'
*
ARETRY   DC    A(INITERR)
*
SERVERRS WTO 'TSYS ERROR IN xxxxxxx SERVICE, RETCODE xx, RSNCODE xxxx',+
               CONSNAME=,MF=L
SERVMSGL EQU   *-SERVERRS
*
INITERRS WTO 'TSYS - SUBSYSTEM INITIALIZATION FAILED',                +
               CONSNAME=,MF=L
INITMSGL EQU   *-INITERRS
*
ESTAERRS WTO 'TSYS - SUBSYSTEM ESTAE FAILED',                         +
               CONSNAME=,MF=L
ESTAMSGL EQU   *-ESTAERRS
*
*
         LTORG
*
WORKAREA DSECT
SAVEAREA DS    18F
         DS    0D
DOUBLE   DS    CL8                    CVD work area
RCODE1   DS    F                      Return/reason code in message
RC       DS    F                      Return code
REASON   DS    F                      Reason code
CBADDR   DS    F                      Control block address
FAILSRV  DS    CL7                    Name of failing service
         DS    0F
TOKEN1   DS    F                      Vector table token
```

```
*
         IEFSSVT MF=(L,VTPARMS)
*
         IEFSSI MF=(L,SSIPARMS)
*
SERVERRD WTO 'TSYS ERROR IN xxxxxxx SERVICE, RETCODE xx, RSNCODE xxxx',+
               CONSNAME=,MF=L
INITERRD WTO 'TSYS - SUBSYSTEM INITIALIZATION FAILED',                 +
               CONSNAME=,MF=L
ESTAERRD WTO 'TSYS - SUBSYSTEM ESTAE FAILED',                          +
               CONSNAME=,MF=L
*
ESTAED   ESTAE PARAM=ARETRY,MF=L
*
WORKALEN EQU *-WORKAREA
*
TSYSCB   DSECT 0D
TSYSID   DS    CL4                     Acronym
TSYSVER  DS    H                       Version
TSYSLEN  DS    H                       Length
*
CBLEN    EQU *-TSYSCB
*
         CVT DSECT=YES                 CVT
*
         IEFJESCT                      JESCT
*
         IEFJSCVT                      SSCVT
*
         IEFJSRC                       SSI return and reason codes
*
         IEFJSIPL                      Initialization routine          +
                                         parameter list
*
         IHASDWA
*
         IEFSSVTI TYPE=LIST
*
         END
```

Example 2 — Subsystem Function Routine (WRITEIT)

```
WRITEIT  CSECT
WRITEIT  AMODE  ANY
WRITEIT  RMODE  ANY
*********************************************************************
*                                                                   *
*  Function:                                                        *
*                                                                   *
*     This function routine of the TSYS subsystem issues a WTO      *
*     to indicate that it has been entered.  The message identifier *
*     of the WTO is returned to the caller in a function dependent  *
*     area.                                                         *
*                                                                   *
*********************************************************************
*                                                                   *
*  Name of the module:  WRITEIT                                     *
*                                                                   *
*  System macros used:                                              *
*                       FREEMAIN                                    *
*                       GETMAIN                                     *
*                       IEFJSCVT                                    *
*                       IEFJSSIB                                    *
*                       IEFSSOBH                                    *
*                       WTO                                         *
*                                                                   *
*  Base register:  12                                               *
*                                                                   *
*  Other register use:                                              *
*                   10   SSCVT                                      *
*                   11   SSOB                                       *
*                    9   SSIB                                       *
*                                                                   *
*  Attributes:                                                      *
*     This routine must be reentrant and reside in a library        *
*     accessible at the time subsystem initialization occurs.       *
*                                                                   *
*********************************************************************
*
*********************************************************************
* Chain saveareas                                                   *
*********************************************************************
         USING WRITEIT,12
         SAVE  (14,12)                Save caller registers
         LR    12,15                  Establish module base register
*
         LR    10,0                   Establish addressability
         USING SSCT,10                 to the SSCVT
         LR    11,1                   Establish addressability
         USING SSOB,11                 to the SSOB
*
         GETMAIN R,LV=72              Get working storage
         ST    13,4(1)                Chain saveareas forward
         ST    1,8(13)                Chain saveareas backward
         LR    13,1                   Point to this module's savearea
*
```

```
*************************************************************************
* Validate the request and issue a WTO for message TSYS001             *
*************************************************************************
         L     9,SSOBSSIB              Establish addressability
         USING SSIB,9                   to the SSIB
         CLC   SSIBSSNM,SSCTSNAM       Verify the subsystem name
         BNE   ERROR                   This should never happen
         WTO 'TSYS001 - WRITEIT FUNCTION EXECUTED',ROUTCDE=(2)
         L     8,SSOBINDV              Pointer to function dependent
*                                        area
         ST    1,2(8)                  Save message identification
*                                        returned by WTO
         MVC   SSOBRETN,=F'0'          Indicate function success
         B     RETURN
*
ERROR    EQU   *
         MVC   SSOBRETN,=F'4'          Indicate function failure
*
*************************************************************************
* Return to the SSI                                                     *
*************************************************************************
RETURN   EQU   *
         L     8,4(13)                 Pointer to caller's savearea
         FREEMAIN R,LV=72,A=(13)
         LR    13,8
         LM    14,12,12(13)            Restore caller' registers
         LA    15,0                    RC=0
         BSM   0,14                    Return to the SSI
*
         IEFJSCVT
*
         IEFSSOBH
*
         IEFJSSIB
*
         END
```

Example 3 — Subsystem Function Routine (DELETEIT)

```
DELETEIT CSECT
DELETEIT AMODE  ANY
DELETEIT RMODE  ANY
*********************************************************************
*                                                                   *
*  Function:                                                        *
*                                                                   *
*     This function routine of the TSYS subsystem deletes a WTO.    *
*     The message identifier of the WTO is passed in a function     *
*     dependent area.                                               *
*                                                                   *
*********************************************************************
*                                                                   *
*  Name of the module:  DELETEIT                                    *
*                                                                   *
*  System macros used:                                              *
*                       DOM                                         *
*                       FREEMAIN                                    *
*                       GETMAIN                                     *
*                       IEFJSCVT                                    *
*                       IEFJSSIB                                    *
*                       IEFSSOBH                                    *
*                                                                   *
*  Base register:  12                                               *
*                                                                   *
*  Other register use:                                              *
*                  10   SSCVT                                       *
*                  11   SSOB                                        *
*                   9   SSIB                                        *
*                                                                   *
*  Attributes:                                                      *
*     This routine must be reentrant and reside in a library        *
*     accessible at the time subsystem initialization occurs.       *
*                                                                   *
*********************************************************************
*
*********************************************************************
* Chain saveareas                                                   *
*********************************************************************
         USING DELETEIT,12
         SAVE  (14,12)                Save caller registers
         LR    12,15                  Establish module base register
*
         LR    10,0                   Establish addressability
         USING SSCT,10                 to the SSCVT
         LR    11,1                   Establish addressability
         USING SSOB,11                 to the SSOB
*
         GETMAIN R,LV=72              Get working storage
         ST    13,4(1)                Chain saveareas foreword
         ST    1,8(13)                Chain saveareas backward
         LR    13,1                   Point to this module's savearea
*
```

```
*************************************************************************
* Validate the request and delete the critical eventual action message *
*************************************************************************
         L     9,SSOBSSIB              Establish addressability
         USING SSIB,9                   to the SSIB
         CLC   SSIBSSNM,SSCTSNAM       Verify the subsystem name
         BNE   ERROR                   This should never happen
         L     8,SSOBINDV              Pointer to function dependent
*                                        area
         L     1,2(8)                  Get message identification
*                                        returned by WTO
         DOM MSG=(1)
         MVC   SSOBRETN,=F'0'          Indicate function success
         B     RETURN
*
ERROR    EQU   *
         MVC   SSOBRETN,=F'4'          Indicate function failure
*
*************************************************************************
* Return to the SSI                                                     *
*************************************************************************
RETURN   EQU   *
         L     8,4(13)                 Pointer to caller's savearea
         FREEMAIN R,LV=72,A=(13)
         LR    13,8
         LM    14,12,12(13)            Restore caller' registers
         LA    15,0                    RC=0
         BSM   0,14                    Return to the SSI
*
         IEFJSCVT
*
         IEFSSOBH
*
         IEFJSSIB
*
         END
```

Example 4 — Subsystem Function Routine (LISTEN)

```
LISTEN   CSECT
LISTEN   AMODE ANY
LISTEN   RMODE ANY
*********************************************************************
*                                                                   *
*  Function:                                                        *
*                                                                   *
*     This function routine of the TSYS subsystem is invoked by the *
*     SSI broadcast of WTO.  When it detects the WTO message issued *
*     by the WRITEIT routine, it alters the attributes of the WTO to*
*     be a non-rollable message.                                    *
*                                                                   *
*********************************************************************
*                                                                   *
*  Name of the module:  LISTEN                                      *
*                                                                   *
*  System macros used:                                              *
*                       FREEMAIN                                    *
*                       GETMAIN                                     *
*                       IEFJSSOB                                    *
*                       IHAWQE                                      *
*                                                                   *
*  Base register:  12                                               *
*                                                                   *
*  Other register use:                                              *
*                   10   SSOB                                       *
*                   11   SSOBEXT                                    *
*                    9   WQE                                        *
*                                                                   *
*  Attributes:                                                      *
*     This routine must be reentrant and reside in a library        *
*     accessible at the time subsystem initialization occurs.       *
*                                                                   *
*********************************************************************
*
*********************************************************************
* Chain saveareas                                                   *
*********************************************************************
         USING LISTEN,12
         SAVE  (14,12)                Save caller registers
         LR    12,15                  Establish module base register
*
         LR    10,1                   Establish addressability
         USING SSOB,10                 to the SSOB
*
         GETMAIN R,LV=72              Get working storage
         ST    13,4(1)                Chain saveareas foreword
         ST    1,8(13)                Chain saveareas backward
         LR    13,1                   Point to this module's savearea
*
```

```
*********************************************************************
* Alter message number TSYS001 to be a critical eventual action     *
* message (descriptor code of 11)                                   *
*********************************************************************
         L     11,SSOBINDV             Chain through
         USING SSOBEXT,11               the SSWT to
         L     9,SSWTWQE                establish addressability
         USING WQE,9                    to the WQE
         CLC   WQETXT+2(8),=C'TSYS001 ' Check for desired message
         BNE   MSGDONE
         TM    WQEDC2,WQEDCK           Check for DESC(11) already set
         BO    MSGDONE
         OI    WQEDC2,WQEDCK           Alter message to be DESC(11)
         OI    WQEML1,WQEMLCE           and eventual critical
         OI    WQEMCSF1,WQEMCSA        Indicate descriptor codes
*                                       present
         MVC   SSOBRETN,=F'4'          Indicate function recognized
*                                       request, and processed it
         B     RETURN
*
MSGDONE  EQU   *
         MVC   SSOBRETN,=F'0'          Indicate function recognized
*                                       request, but did not care
*
*********************************************************************
* Return to the SSI                                                 *
*********************************************************************
RETURN   EQU   *
         L     8,4(13)                 Pointer to caller's savearea
         FREEMAIN R,LV=72,A=(13)
         LR    13,8
         LM    14,12,12(13)            Restore the caller's registers
         LA    15,0                    RC=0
         BSM   0,14                    Return
*
*
         IEFJSSOB (WT),CONTIG=NO
*
         IHAWQE
*
         END
```

Example 5 — Subsystem Requesting Routine (TSYSCALL)

```
TSYSCALL CSECT
TSYSCALL AMODE ANY
TSYSCALL RMODE ANY
*************************************************************************
*                                                                       *
*  Function:                                                            *
*                                                                       *
*     This routine runs as a problem program and invokes the TSYS       *
*     subsystem.  It requests the SSI to invoke the WRITEIT function    *
*     to issue its WTO.  Ten seconds later it requests the SSI to       *
*     invoke the DELETEIT function to delete the WTO.                   *
*                                                                       *
*     For the WTO to be broadcast to all subsystems, this routine       *
*     must be run SUB=MSTR.                                             *
*                                                                       *
*************************************************************************
*                                                                       *
*  Name of the module:  TSYSCALL                                        *
*                                                                       *
*  System macros used:                                                  *
*                       ABEND                                           *
*                       CVT                                             *
*                       IEFJESCT                                        *
*                       IEFJSSIB                                        *
*                       IEFSSOBH                                        *
*                       IEFSSREQ                                        *
*                       RETURN                                          *
*                       STIMER                                          *
*                                                                       *
*  Base register:  12                                                   *
*                                                                       *
*  Other register use:                                                  *
*                   10   SSOB                                           *
*                   11   SSIB                                           *
*                                                                       *
*  Attributes:                                                          *
*     None                                                              *
*                                                                       *
*************************************************************************
```

```
*
*********************************************************************
* Chain saveareas                                                   *
*********************************************************************
         USING TSYSCALL,12
         SAVE  (14,12)                 Save caller registers
         LR    12,15                   Establish module base register
         LR    1,13
         LA    13,SAVEAREA             Point to this module's savearea
         ST    13,8(1)                 Chain saveareas foreword
         ST    1,SAVEAREA+4            Chain saveareas backward
*
         LA    10,SSOBD                Establish addressability
         USING SSOB,10                  to the SSOB
         LA    11,SSIBD                Establish addressability
         USING SSIB,11                  to the SSIB
*
*********************************************************************
* Format the SSOB                                                   *
*********************************************************************
         MVC   SSOBID,=C'SSOB'         Set control block identifier
         LA    8,SSOBHSIZ
         STH   8,SSOBLEN               Set control block size
         ST    11,SSOBSSIB             Set pointer to SSIB
         MVC   SSOBINDV,=A(MSGIDEXT)   Set pointer to function
*                                        dependent area
*
*********************************************************************
* Format the SSIB                                                   *
*********************************************************************
         MVC   SSIBID,=C'SSIB'         Set control block identifier
         LA    8,SSIBSIZE
         STH   8,SSIBLEN               Set control block size
         MVC   SSIBSSNM,=C'TSYS'       Set subsystem name
*
*********************************************************************
* Call the TSYS subsystem                                           *
*********************************************************************
         MVC   SSOBFUNC,WRITEIT        Request the TSYS001 WTO message
         OI    PARMLST,X'80'           Mark end of parameter list
         LA    1,PARMLST               Point to the parameter list
         IEFSSREQ
         LTR   15,15                   Check return code from SSI
         BNZ   ERROR
         CLC   SSOBRETN,=F'0'          Check return code from subsystem
         BNZ   ERROR
*
         STIMER WAIT,BINTVL=TENSEC
```

```
*
         MVC   SSOBFUNC,DELETEIT       Request DOM of the TSYS001 WTO
*                                       message
         LA    1,PARMLST               Point to the parameter list
         IEFSSREQ
         LTR   15,15                   Check return code from SSI
         BNZ   ERROR
         CLC   SSOBRETN,=F'0'          Check return code from subsystem
         BNZ   ERROR
         B     RETURN
*
ERROR    EQU   *
         ABEND 1001,,,USER             Indicate function failure
*
*********************************************************************
* Restore registers and return                                      *
*********************************************************************
RETURN   EQU   *
         L     13,SAVEAREA+4           Pointer to caller's savearea
         RETURN (14,12),RC=0
*
*
TENSEC   DC    F'1000'                 Ten seconds in 1/100ths
WRITEIT  DC    H'254'
DELETEIT DC    H'255'
*
SAVEAREA DC    18F'0'
*
PARMLST  DC    A(SSOBD)                IEFSSREQ parameter list
*
SSOBD    DS    0F                      SSOB data
         DC    (SSOBHSIZ)X'00'
*
MSGIDEXT DS    0F                      Function dependent area
MSGIDLEN DC    AL2(MSGIDSIZ)
MSGIDENT DC    F'0'                    Message identifier from TFUNC1
MSGIDSIZ EQU   *-MSGIDEXT
*
SSIBD    DS    0F                      SSIB data
         DC    (SSIBSIZE)X'00'
*
         IEFSSOBH
*
         IEFJSSIB
*
         CVT DSECT=YES
*
         IEFJESCT
*
         END
```

Appendix A — Examples

Appendix B. Using IEFJSVEC with Your Subsystem

This appendix describes using the IEFJSVEC service to help build and use your subsystems when performing the following tasks:

- Defining what your subsystem can do:
  - Building your subsystem's SSVT
- Changing what your subsystem can do:
  - Enabling your subsystem for new functions
  - Disabling previously supported functions

IBM recommends that you use the dynamic SSI services that are described in “Services for Building and Using Your Subsystem” on page 167 instead of using IEFJSVEC. The dynamic SSI services provide new capabilities and are easier to use.

Defining What Your Subsystem Can Do

To define what your subsystem can do, you can use IEFJSVEC to build an SSVT for your subsystem.

Building the SSVT

The IEFJSVEC service allows you to build an SSVT for your subsystem.

When preparing to build your subsystem's SSVT, consider:

- When you want to invoke IEFJSVEC. You can invoke IEFJSVEC either through a subsystem initialization routine specified in parmlib member IEFSSNxx or through a subsystem routine invoked during START command processing, as described under “Providing a Routine to Initialize Your Subsystem” on page 162.
- Which common storage subpool your subsystem's SSVT is to be built in. Note that the system uses the mode and key of the caller to access the SSVT and invoke the function routines. Therefore, the storage subpool specified for the SSVT must be a common subpool. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on selecting a common storage subpool.
- What are the maximum number of function routines you expect the subsystem to need. The maximum number of function routines you specify applies to the function routines you define on this build request, and also to any function routines that you define on the enable function or disable function of the IEFJSVEC service.
- What are the actual number of function routines you want to specify on the current request.
- What is the name of each function routine and the function code it supports.
- Where the subsystem function routines are to reside. See “Placement of Function Routines” on page 159 for more information.

Environment

The following mapping macros are supplied by IBM and may be included in your program when invoking IEFJSVEC:

- IEFVTSPL
- IEFJSBVT

The requirements for the caller of IEFJSVEC are:

Minimum Authorization	Supervisor state with any PSW key
Dispatchable unit mode	Task
AMODE	24-bit
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Control Parameters	The VTSPL and JSBVT control blocks must reside in storage below 16 megabytes.
Recovery	The caller of IEFJSVEC should provide an ESTAE-type of recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on an ESTAE-type of recovery environment.

Input Register Information

Before invoking the IEFJSVEC service, you must ensure that the following general purpose registers contain:

Register	Contents
1	Address of fullword that contains the address of the subsystem VTSPL
13	Address of a standard 18-word save area
14	Return address

Input Parameters

Input parameters for the IEFJSVEC service are:

- VTSPL
- JSBVT — both fixed and variable sections

VTSPL Contents: Your program sets the following fields in the VTSPL control block on input:

Field Name	Description
VTSID	Identifier 'VTSP'
VTSLEN	Length of the VTSPL (VTSSIZE) control block
VTSVER	Version number of the VTSPL (VTSCVER) control block
VTSCONID	The 1-byte console ID of the console that the subsystem initialization routine issues messages to. If your program sets this field to zero, the VTSCNSID field is used. This field exists for versions of MVS previous to SP410. IBM recommends that you specify a 4-byte console ID as defined by the VTSCNSID field.

VTSFLAGS Flags

- **VTSGLOAD** — load-to-global indicator.

  To eliminate the need to have subsystem function routines reside in LPA, the subsystem can request that IEFJSVEC issue a load-to-global for those function routines by setting the VTSGLOAD indicator. If load-to-global is used for the subsystem function routines, the function routines are loaded into pageable CSA and the loaded routines are associated with the requesting task. When the task ends, the module's use count is reduced by the number of outstanding LOADs. When the module's use count reaches zero, the module is deleted, leaving an invalid function routine address in the SSVT. Therefore, the load-to-global option should only be used by programs running under a task that never ends. For example, if IEFJSVEC is invoked by the subsystem initialization routine which is given control out of early system initialization (that is, those subsystem initialization routines specified in IEFSSNxx parmlib members) the requesting task is the master scheduler, which never goes away.

  If you set the load-to-global indicator, all function routines which are specified on a single request to IEFJSVEC are loaded into pageable CSA. If you want to have some function routines loaded into CSA and others that are not, issue separate invocations of IEFJSVEC, one with the VTSGLOAD indicator set and the other with the VTSGLOAD indicator not set. Because your subsystem can only have one SSVT, for subsequent calls to IEFJSVEC, you need to use the enable function code request option available through the IEFJSVEC service. See “Enabling Your Subsystem for New Functions” on page 284 for more information.

VTSREQ Request flags — defines the operation that this call performs

- **VTSCREAT** — SSVT build indicator

VTSNAME Subsystem name. The name of the subsystem for which the SSVT is being built. The subsystem name can be up to four characters. It must be left-justified and padded to the right with blank (X'40') characters.

VTSSVTD Address of SSVT table data (mapped by IEFJSBVT)

VTSCNSID 4-byte console ID that the SSI uses for any messages issued on this invocation of IEFJSVEC. If this field is set to zero, the messages go to the master console.

Provide a CART and a console ID if IEFJSVEC is invoked while running under a command processor. For example, if a subsystem is initialized through START command processing. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for information on how to obtain the CART and console ID from the command input buffer (CIB) control block.

VTSCART Command and response token (CART). If a CART is provided, the SSI uses it for any messages it issues for this invocation of IEFJSVEC.

Set all other fields in the VTSPL control block to binary zeros.

JSBVT Contents — Fixed Header Section: Your program sets the following fields in the JSBVT control block on input:

Field Name	Description
JSBID	Identifier 'JSBV'
JSBLEN	Length of the JSBVT (fixed header section) control block
JSBVERS	Version number of the JSBVT (JSBCVERS) control block
JSBFUN	Number of function routines specified in this table of data
JSBSPL	Subpool number from which the SSVT is to be built. Note that the system uses the mode and key of the caller to access the SSVT and invoke the function routines.
JSBMAXFR	Maximum number of function routines you expect the subsystem to need

Set all other fields in the fixed header section of the JSBVT control block to binary zeros.

JSBVT Contents — Variable Length Section: The JSBVT fixed header is followed by a variable length function routine data area (one for each function routine). Your program sets the following fields on input:

Field Name	Description
JSBLGTH	Length of this function routine's data area and function code area (also see JSBFCOD)
JSBNME	Name of the function routine. The function routine name can be up to eight characters. It must be left-justified and padded to the right with blank (X'40') characters.
JSBNUM	Number of function codes the function routine supports

JSBVT Contents — Variable Length Section: The function routine data area is follow by a variable length function code area (one for each function routine). Your program sets the following fields on input:

Field Name	Description
JSBFCOD	Function code (repeat if more than one function code is supported by the same function routine). The value specified for each function code must be in the range 1-255.

Output Register Information

When control returns to caller of the IEFJSVEC service, the general purpose registers contain:

Register	Contents
0 — 14	Same as on entry to call
15	Return code

Return Code Information

IEFJSVEC returns one of the following return codes in register 15:

Return Code (Decimal)	Meaning
VTSSUCES (0)	Successful completion. The request to build an SSVT was successfully processed.
VTSINVID (4)	An incorrect identifier was specified in VTSPL or JSBVT. Check the input parameter areas to make sure that you specified the proper identifiers, and that the pointers to the input parameter areas are properly defined.
VTSINVIN (8)	An incorrect subsystem name was specified. Check to make sure that you specified a valid subsystem name in the VTSNAME field. Consult with your system programmer to make sure that it matches the name of a valid subsystem defined in the IEFSSNxx parmlib member that is currently in use.
VTSGETFL (12)	Unable to obtain storage for the SSVT. Consult with your system programmer to verify that sufficient storage is available for the subpool specified in the JSBSPL field.
VTSLOGER (16)	Logic error. Contact your IBM service support center.
VTSLOADF (20)	An abend occurred when trying to load the function routine. The VTSFUNCT field contains the name of the function routine being loaded when the problem occurred.
VTSINVBI (24)	An incorrect bit was set in the request flags. Verify that you have set only the VTSCREAT indicator and that you have not set any other bits in the VTSREQ flag byte.
VTSINCR (28)	Unable to process the SSVT build request. The SSVT already exists. Verify that you have specified the correct subsystem name for which an SSVT is to be built. Also ensure that your subsystem initialization code is not accidentally attempting to build an SSVT twice for the same subsystem (specified in VTSNAME).

Output Parameters

Output parameters for the IEFJSVEC service are:

- VTSPL

VTSPL Contents: The VTSPL control block contains the following information upon return from your build SSVT request:

Field Name	Description
VTSSVTAD	Address of the SSVT, if the SSVT build request was successful (Register 15=0)
VTSSSCVT	Address of the SSCVT, if the SSVT build request was successful (Register 15=0)
VTSFUNCT	Name of the function routine processed, if an error occurred when trying to load a function routine (Register 15=20)

Changing What Your Subsystem Can Do

To change what your subsystem can do, you can use IEFJSVEC to:

- Enable your subsystem for new functions
- Disable a previously supported function

Enabling Your Subsystem for New Functions

You can use the enable function of the IEFJSVEC service to:

- Dynamically add one or more function codes to an existing function routine. This function routine might have been specified on the original build SSVT request or might have been added by a previous enable request.

  When preparing to enable additional function codes, consider:

  – When you will invoke IEFJSVEC.

    If you are invoking IEFJSVEC while running under a command processor, for example, from a subsystem routine invoked during START command processing, provide a console ID and CART, as described in “Input Parameters” on page 285.

  – Which existing function routines will support which additional function codes.

- Dynamically add one or more new function routines, and, for each function routine, one or more function codes that the function routine is to support.

  When preparing to enable additional function routines and function codes, consider the following:

  – When you will be invoking IEFJSVEC.

    If you are invoking IEFJSVEC while running under a command processor, for example, from a subsystem routine invoked during START command processing, then provide a console ID and CART, as described in “Input Parameters” on page 285.

  – What are the actual number of function routines your subsystem currently supports and is it less the maximum number allowed.

    To dynamically add more function routines to your subsystem, the actual number of function routines your subsystem currently supports must be less than the maximum number of function routines that was specified when your subsystem's SSVT was built. See the description for the JSBMAXFR field in “Building the SSVT” on page 279.

  – What is the name of each additional function routine and the function codes it is to support.

  – Where your subsystem function routines are to reside. See “Setting Up Your Subsystem” on page 157 for more information on where your function routines can reside.

Environment

The following mapping macros are supplied by IBM and may be included in your program when invoking IEFJSVEC:

- IEFVTSPL
- IEFJSBVT

The requirements for the caller of IEFJSVEC are:

Minimum Authorization	Supervisor state with any PSW key
Dispatchable unit mode	Task
AMODE	24-bit
Control Parameters	The VTSPL and JSBVT control blocks must reside in storage below 16 megabytes.
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Recovery	The caller of IEFJSVEC should provide an ESTAE-type of recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on an ESTAE-type of recovery environment.

Restrictions

The number of function routines supported by a subsystem must not exceed 255.

Input Register Information

Before you invoke IEFJSVEC, you must ensure that the following general purpose registers contain:

Register	Contents
1	Address of fullword that contains the address of the subsystem VTSPL
13	Address of a standard 18-word save area
14	Return address

Input Parameters

Input parameter areas for the IEFJSVEC service are:

- VTSPL
- JSBVT — both fixed and variable sections

VTSPL Contents: Your program must set the following fields in the VTSPL control block on input:

Field Name	Description
VTSID	Identifier 'VTSP'
VTSLEN	Length of the VTSPL (VTSSIZE) control block
VTSVER	Version number of the VTSPL (VTSCVER) control block
VTSCONID	The 1-byte console ID of the console that the subsystem initialization routine issues messages to. If your program sets this field to zero, the VTSCNSID field is used.

This field exists for versions of MVS previous to SP410. IBM recommends that you specify a 4-byte console ID as defined by the VTSCNSID field.

VTSFLAGS Flags

- **VTSGLOAD** — load-to-global indicator.

  This indicator applies only when you are adding new function routines to your subsystem and does not apply when you are adding new function codes to an existing function routine. If the VTSGLOAD indicator is set, the SSI loads all of the function routines into pageable CSA. Each loaded routine is associated with the task under which the call to IEFJSVEC was made. The VTSGLOAD indicator applies to all function routines specified on a single invocation of IEFJSVEC.

  Only use the VTSGLOAD indicator when invoking the enable function under a system address space that does not end. If the subsystem invokes the enable function from its own address space or task, those routines are deleted from CSA when the task ends, causing invalid function routine addresses in the SSVT. IBM recommends that you use the VTSGLOAD indicator only when invoking IEFJSVEC from an initialization routine named in IEFSSNxx. Subsystems initialized through START commands should ensure that the function routines are in commonly addressable storage, that is, in the link pack area (LPA, MLPA, FLPA).

  If you want to have some function routines that are loaded into CSA and others that are not, issue separate invocations of IEFJSVEC, one with the VTSGLOAD indicator set and the other with the VTSGLOAD indicator not set. You may use the SSVT build function for one of the requests, if an SSVT does not already exist. However, for any subsequent calls you will need to use the enable function.

  See “Placement of Function Routines” on page 159 to determine whether the load-to-global indicator should be used.

VTSREQ Request flags - defines the operation that this call performs

- **VTSFCEN** — Enable indicator

VTSNAME Subsystem name. The name of the subsystem for which additional function codes or function routines are to be added. The subsystem name can be up to four characters. It must be left-justified and padded to the right with blank (X'40') characters.

VTSSVTD Address of SSVT table data (see JSBVT content)

VTSCNSID 4-byte console ID that the SSI uses for any messages issued on this invocation of IEFJSVEC. If this field is set to zero, the messages go to the master console.

Provide a CART and a console ID if IEFJSVEC is invoked while running under a command processor. For example, if a subsystem is initialized through START command processing. See *OS/390 MVS Programming: Authorized Assembler Services*

Guide for information on how to obtain the CART and console ID from the command input buffer (CIB) control block.

When IEFJSVEC is invoked during early system initialization, that is, the subsystem is initialized through an initialization routine specified in the IEFSSNxx parmlib member, set the VTSCNSID field to zero.

VTSCART Command and response token (CART). If a CART is provided, the SSI uses it for any messages it issues for this invocation of IEFJSVEC.

Provide a CART and a console ID when IEFJSVEC is invoked while running under a command processor, as when a subsystem is initialized through START command processing. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for information on how to obtain the CART and console ID from the command input buffer (CIB) control block.

Set the VTSCART field to zero when IEFJSVEC is invoked during early system initialization, that is, when the subsystem is initialized through an initialization routine specified in an IEFSSNxx parmlib member.

All other fields in the VTSPL control block must be set to binary zeros.

JSBVT Contents — Fixed Header Section: Your program must set the following fields in the JSBVT control block on input:

Field Name	Description
JSBID	Identifier 'JSBV'
JSBLEN	Length of the JSBVT (fixed header section) control block
JSBVERS	Version number of the JSBVT (JSBCVERS) control block
JSBFUN	Number of function routines specified in this table of data

All other fields in the fixed header section of the JSBVT control block must be set to binary zeros.

JSBVT Contents — Variable Length Section: The JSBVT fixed header is followed by a variable length function routine data area (one for each function routine). Your program must set the following fields on input:

Field Name	Description
JSBLGTH	Length of this function routine's data area and its function code area (see the JSBFCOD field)
JSBNME	Name of a function routine. The function routine name specified should be either the name of a new function routine to be supported by the subsystem or the name of an existing function routine to which additional function codes are to be added. The function routine name can be up to eight characters. It must be left-justified and padded to the right with blank (X'40') characters.
JSBNUM	Number of function codes specified for this function routine. If this enable request is being used to add a new function routine to a subsystem or is being used to add new function codes to an existing function routine, the JSBNUM field should be set to the

number of new function codes to be supported by the function routine as specified in the JSBFCOD field on this invocation of IEFJSVEC.

JSBVT Contents — Variable Length Section: The function routine data area is follow by a variable length function code area (one for each function routine). Your program must set the following field on input:

Field Name	Description
JSBFCOD	Function code(s) (repeat if more than one function code is supported by the same function routine). The value specified for each function code, must be in the range 1-255.

Output Register Information

When control returns to caller of IEFJSVEC, the general purpose registers contain:

Register	Contents
0 — 14	Same as on entry to call
15	Return code

Return Code Information

IEFJSVEC returns one of the following return codes in register 15:

Return Code (Decimal)	Meaning
VTSSUCES (0)	Successful completion. The request to enable was successfully processed and the SSVT has been updated.
VTSINVID (4)	An incorrect identifier was specified in VTSPL or JSBVT. Check the input parameter areas to make sure that you specified the proper identifiers, and that the pointers to the input parameter areas are properly defined.
VTSINVIN (8)	An incorrect subsystem name was specified. Check to make sure that you specified a valid subsystem name in the VTSNAME field. Consult with your system programmer to make sure that it matches the name of a valid subsystem defined in the IEFSSNxx parmlib member that is currently in use.
VTSLOGER (16)	Logic error. Contact your IBM service support center.
VTSLOADF (20)	An abend occurred when trying to load the function routine. The VTSFUNCT field contains the name of the function routine being loaded when the problem occurred.
VTSINVBI (24)	An incorrect bit was set in the request flags. Verify that you have set only the VTSFCEN indicator and that you have not set any other bits in the VTSREQ flag byte.
VTSINVED (32)	Unable to process enable request; no SSVT found. Verify that you specified a valid subsystem name in the VTSNAME field. If the subsystem name is valid, make sure that the subsystem's SSVT has been built and is properly pointed to from your subsystem's SSCVT prior to any IEFJSVEC enable calls being made.

VTSNOSPA (36) Unable to process enable request; insufficient space in the SSVT for additional function routine addresses. The VTSFUNC field contains the name of the function routine being loaded when the problem occurred. The maximum number of function routines which can be supported by your subsystem has been exceeded. Increase the maximum allowed on your build SSVT by increasing JSBMAXFR.

VTSSIVT (40) Target vector table is SSI-managed and can only be updated through the IEFSSVT macro.

VTSNOSUB (44) Target Subsystem does not exist.

Output Parameters

Output parameters for the IEFJSVEC service are:

- VTSPL

VTSPL Contents: The VTSPL control block contains the following information upon return from your enable request:

Field Name	Description
VTSSSCVT	Address of the SSCVT, if the enable request was successful (Register 15=0)
VTSFUNCT	Name of the function routine being processed, if an error occurred when trying to load a function routine (Register 15=20 or Register 15=36)

Disabling Previously Supported Functions

You can use the disable function of the IEFJSVEC service to dynamically disable a function code so that your subsystem no longer gets control for that function. Disabling a function is in effect a "logical delete."

Attention: Because there is no serialization on updating the table in the SSVT, other requests for the supported functions might be coming in asynchronously. Therefore, it is important to not remove the function routines from storage.

When preparing to disable one or more function codes, consider:

- When you will be invoking IEFJSVEC

  If you are invoking IEFJSVEC while running under a command processor, for example, from a subsystem routine invoked during START command processing, then a console ID and CART should be provided, as described in “Input Parameters” on page 290.

- Which of the existing function codes are no longer supported.

Environment

The following mapping macros are supplied by IBM and may be included in your program when invoking IEFJSVEC:

- IEFVTSPL
- IEFJSBVT

The requirements for the caller of IEFJSVEC are:

Minimum Authorization	Supervisor state with any PSW key
Dispatchable unit mode	Task
AMODE	24-bit
Control Parameters	The VTSPL and JSBVT control blocks must reside in storage below 16 megabytes.
Cross memory mode	PASN=HASN=SASN
ASC mode	Primary
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks held
Recovery	The caller of IEFJSVEC should provide an ESTAE-type of recovery environment. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for more information on an ESTAE-type of recovery environment.

Input Register Information

Before you invoke IEFJSVEC, you must ensure that the following general purpose registers contain:

Register	Contents
1	Address of fullword that contains the address of the subsystem VTSPL
13	Address of a standard 18-word save area
14	Return address

Input Parameters

Input parameter areas for the IEFJSVEC service are:

- VTSPL
- JSBVT — both fixed and variable sections

VTSPL Contents: Your program must set the following fields in the VTSPL control block on input:

Field Name	Description
VTSID	Identifier 'VTSP'
VTSLEN	Length of the VTSPL (VTSSIZE) control block
VTSVER	Version number of the VTSPL (VTSCVER) control block
VTSCONID	The 1-byte console ID of the console that the subsystem initialization routine issues messages to. If your program sets this field to zero, the VTSCNSID field is used. This field exists for versions of MVS previous to SP410. IBM recommends that you specify a 4-byte console ID as defined by the VTSCNSID field.
VTSREQ	Request flags - defines the operation that this call performs • **VTSFCDIS** — Disable indicator
VTSNAME	Subsystem name. The name of the subsystem for which one or more function codes are to be disabled. The subsystem name can be up to four characters. It must be left-justified and padded to the right with blank (X'40') characters.
VTSSVTD	Address of SSVT table data (see JSBVT contents)

VTSCNSID 4-byte console ID that the SSI uses for any messages issued on this invocation of IEFJSVEC. If this field is set to zero, the messages go to the master console.

Provide a CART and a console ID if IEFJSVEC is invoked while running under a command processor. For example, if a subsystem is initialized through START command processing. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for information on how to obtain the CART and console ID from the command input buffer (CIB) control block.

When IEFJSVEC is invoked during early system initialization, that is, the subsystem is initialized through an initialization routine specified in the IEFSSNxx parmlib member, set the VTSCNSID field to zero.

VTSCART Command and response token (CART). If a CART is provided, the SSI uses it for any messages it issues for this invocation of IEFJSVEC.

Provide a CART and a console ID when IEFJSVEC is invoked while running under a command processor, as when a subsystem is initialized through START command processing. See *OS/390 MVS Programming: Authorized Assembler Services Guide* for information on how to obtain the CART and console ID from the command input buffer (CIB) control block.

Set the VTSCART field to zero when IEFJSVEC is invoked during early system initialization, that is, when the subsystem is initialized through an initialization routine specified in an IEFSSNxx parmlib member.

All other fields in the VTSPL control block must be set to binary zeros.

JSBVT Contents — Fixed Header Section: Your program must set the following fields in the JSBVT control block on input:

Field Name	Description
JSBID	Identifier 'JSBV'
JSBLEN	Length of fixed header section
JSBVERS	Version number (JSBCVERS)
JSBFUN	Number of function routines specified in this table of data

All other fields in the fixed header section of the JSBVT control block must be set to binary zeros.

JSBVT Contents — Variable Length Section: The JSBVT fixed header is followed by a variable length function routine data area (one for each function routine). Your program must set the following fields on input:

Field Name	Description
JSBLGTH	Length of this function routine's data area and it's function code area (see the JSBFCOD field)
JSBNME	Name of a function routine. The function routine name can be up to eight characters. It must be left-justified and padded to the right with blank (X'40') characters.

JSBNUM Number of function codes

The JSBNUM field should be set to the number of function codes which are to be disabled for this function routine as specified in the JSBFCOD field on this invocation of IEFJSVEC.

JSBVT Contents — Variable Length Section: The function routine data area is followed by a variable length function code area (one for each function routine). Your program must set the following field on input:

Field Name	Description
JSBFCOD	Function code (repeat if more than one function code is to be disabled). The value specified for each function code, must be in the range 1-255.

Output Register Information

When control returns to caller of IEFJSVEC, the general purpose registers contain:

Register	Contents
0 — 14	Same as on entry to call
15	Return code

Return Code Information

IEFJSVEC returns one of the following return codes in register 15:

Return Code (Decimal)	Meaning
VTSSUCES (0)	Successful completion. The request to disable was successfully processed and the SSVT has been updated.
VTSINVID (4)	An incorrect identifier was specified in VTSPL or JSBVT. Check the input parameter areas to make sure that you specified the proper identifiers, and that the pointers to the input parameter areas are properly defined.
VTSINVIN (8)	An incorrect subsystem name was specified. Check to make sure that you specified a valid subsystem name in the VTSNAME field. Consult with your system programmer to make sure that it matches the name of a valid subsystem defined in the IEFSSNxx parmlib member that is currently in use.
VTSLOGER (16)	Logic error. Contact your IBM service support center.
VTSINVBI (24)	An incorrect bit was set in the request flags. Verify that you have set only the VTSFCDIS indicator and that you have not set any other bits in the VTSREQ flag byte.
VTSINVED (32)	Unable to process disable request; no SSVT found. Verify that you specified a valid subsystem name in the VTSNAME field. If the subsystem name is valid, make sure that the subsystem's SSVT has been built and is properly pointed to from your subsystem's SSCVT prior to any IEFJSVEC disable calls being made.

Output Parameters

Output parameters for the IEFJSVEC service are:

- VTSPL

VTSPL Contents: The VTSPL control block contains the following information upon return from your disable request:

Field Name	Description
VTSSSCVT	Address of the SSCVT, if the disable request was successful (Register 15=0)

Appendix B — Using IEFJSVEC

Appendix C. Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Programming Interface Information

This book is intended to help the customer to define and use a subsystem on an MVS operating system. It also describes services provided by IBM subsystems that a program or subsystem can use. This book primarily documents General-use Programming Interface and Associated Guidance Information provided by OS/390.

General-use programming interfaces allow the customer to write programs that obtain the services of OS/390.

However, this book also documents Product-sensitive Programming Interfaces and Associated Guidance Information and Diagnosis, Modification or Tuning Information provided by OS/390.

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of OS/390. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section, or by the following marking:

Product-sensitive programming interface

Product-sensitive Programming Interface and Associated Guidance Information...

End of Product-sensitive programming interface

Diagnosis, Modification or Tuning Information is provided to help the customer to diagnose problems in subsystems of OS/390.

Attention: Do not use this Diagnosis, Modification or Tuning Information as a programming interface.

Diagnosis, Modification or Tuning Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Diagnosis, Modification or Tuning Information

Diagnosis, Modification or Tuning Information...

End of Diagnosis, Modification or Tuning Information

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- AnyNet
- AT
- BookManager
- C++/MVS
- C/MVS
- CBIPO
- CICS
- CICS OS/2
- CICS/ESA
- CICSPlex

- CT
- DB2
- DFSMS/MVS
- DFSMSdfp
- DFSMSdss
- DFSMShsm
- DFSMSrmm
- DFSORT
- Extended Services
- ESCON
- FFST
- FFST/MVS
- GDDM
- Hiperbatch
- IBM
- IBMLink
- IMS
- IMS/ESA
- Language Environment
- MVS/ESA
- MVS/S
- NetView
- Open Class
- OPC
- OS/2
- OS/390
- Parallel Sysplex
- PR/SM
- PSF
- Resource Measurement Facility
- RACF
- RMF
- S/390
- System/390
- SystemView
- SOMobjects
- SP
- VisualLift
- VM/ESA
- VSE/ESA
- VTAM

UNIX is a registered trademark of The Open Group in the United States and other countries.

Index

F

I

J

L

M

N

P

R

S

T

U

V

W

Communicating Your Comments to IBM

OS/390
MVS Using the Subsystem Interface

Publication No. SC28-1789-07

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:

  1-(845)-432-9405
- If you prefer to send comments electronically, use this network ID:

  mhvrcfs@us.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

OS/390
MVS Using the Subsystem Interface
Publication No. SC28-1789-07

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC28-1789-07

Cut or Fold
Along Line

IBM®

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5647-A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC28-1789-07