MVS Using the Subsystem Interface
MVS Using the Subsystem Interface
Note
Before using this information and the product it supports, be sure to read the general information under Notices on page 419.

Tenth Edition, September 2009
This is a major revision of SA22-7642-08.
This edition applies to Version 1 Release 11 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. A form for readers’ comments may be provided at the back of this document, or you may address your comments to the following address:
International Business Machines Corporation
MHVRCFS, Mail Station P181
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: mhvrdfs@us.ibm.com

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 document
• 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.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
## Contents

**Figures** ................................. vii

**Tables** ................................. ix

**About this document** ........................ xi
- Who should use this document .................. xi
- How to use this document ....................... xi
- Where to find more information ................. xi
- Information updates on the web .................. xi

**Summary of Changes** ................... xiii

**Chapter 1. Introduction to Subsystems and the Subsystem Interface (SSI)** 1
- What is a Subsystem? .......................... 1
- What is the SSI? ................................ 1
  - Unique Attributes of the SSI ................. 1
  - Types of Subsystem Requests ................. 2
  - Controlling SSI Processing .................. 3
- Why Write Your Own Subsystem? ............... 3
- What is a Dynamic Subsystem? ................. 4

**Chapter 2. Making a Request of a Subsystem** .................. 7
- Set Up the Environment ....................... 7
  - Subsystem Options Block (SSOB) .............. 7
  - SSOB Function Dependent Area ................ 8
  - Subsystem Identification Block (SSIB) ........ 8
- Make the Request with the IEFSSREQ Macro .... 9
- Check the Return Code ......................... 10
- Summary of Steps ............................. 11

**Chapter 3. SSI Function Codes Your Program Can Request** ...... 13
- SSI Function Code Descriptions ............... 13
  - Process SYSOUT Data Sets Call — SSI Function Code 1 .... 14
  - User Destination Validation/Conversion — SSI Function Code 11 ... 40
  - Verify Subsystem Function Call — SSI Function Code 15 .... 45
  - Request Job ID Call — SSI Function Code 20 .... 49
  - Return Job ID Call — SSI Function Code 21 .... 56
  - Request Subsystem Version Information Call — SSI Function Code 54 .... 60
  - Scheduler Facilities Call - SSI Function Code 70 .... 77
  - JES Job Information Services— SSI Function Code 71 .... 87
  - Notify User Message Service Call — SSI Function Code 75 .... 128
  - SYSOUT Application Program Interface (SAPI) — SSI Function Code 79 .... 134
  - Extended Status Function Call — SSI Function Code 80 .... 172
  - JES Properties— SSI Function Code 82 .... 217

**Chapter 4. Setting Up Your Subsystem** .......................... 291
- Function Routines/Function Codes .............. 291
  - Environment ................................ 291
  - Recovery and Integrity ....................... 292
  - Placement of Function Routines ............ 293
- Do You Need a Subsystem Address Space? .... 293
- Defining Your Subsystem ....................... 294
- Providing a Routine to Initialize Your Subsystem .. 296

© Copyright IBM Corp. 1988, 2009
Chapter 8. Examples — Subsystem Interface Routines .................. 389
Example 1 — Subsystem Initialization Routine (TSYSINIT) .......... 389
Example 2 — Subsystem Function Routine (WRITEIT) ............... 394
Example 3 — Subsystem Function Routine (DELETEIT) ............. 396
Example 4 — Subsystem Function Routine (LISTEN) ................. 397
Example 5 — Subsystem Requesting Routine (TSYSCALL) ........... 399

Chapter 9. Using IEFJSVEC with Your Subsystem ...................... 403
Defining What Your Subsystem Can Do .................................. 403
Building the SSVT ....................................................... 403
Changing What Your Subsystem Can Do .................................. 407
Enabling Your Subsystem for New Functions ......................... 407
Disabling Previously Supported Functions ......................... 413

Appendix. Accessibility ................................................... 417
Using assistive technologies ................................................. 417
Keyboard navigation of the user interface ................................ 417
z/OS information ......................................................... 417

Notices ................................................................. 419
Programming Interface Information ...................................... 420
Policy for unsupported hardware ........................................ 420
Trademarks ............................................................... 421

Index ................................................................. 423
Figures

1. Processing for a Directed Request ................................................. 2
2. Processing for a Broadcast Request .............................................. 3
3. Making a Subsystem Request ..................................................... 10
4. Environment at Time of Call for SSI Function Code 1 .......................... 16
5. Environment at Time of Call for SSI Function Code 11 ....................... 41
6. Environment at Time of Call for SSI Function Code 15 ....................... 46
7. Environment at Time of Call for SSI Function Code 20 ....................... 51
8. Environment at Time of Call for SSI Function Code 21 ....................... 57
9. Environment at Time of Call for SSI Function Code 54 ....................... 62
10. Environment at Time of Call for SSI Function Code 70 ....................... 78
11. Environment at Time of Call for SSI Function Code 71, Spool Read Subfunction 89
12. Environment at Time of Call for SSI Function Code 71, JES2 Health Monitor Information Subfunction .................................................. 96
13. Environment at Time of Call for SSI Function Code 71, Job Class Information Subfunction .................................................. 107
14. Environment at Time of Call for SSI Function Code 71, Convert Device ID Service Subfunction .................................................. 117
15. Environment at Time of Call for SSI Function Code 71, Checkpoint Versions Subfunction .................................................. 122
16. Environment at Time of Call for SSI Function Code 75 ....................... 129
17. Protocol for the SAPI PUT/GET Call .............................................. 138
18. Control Blocks of DYNALLOC Call for SAPI-Provided Data Set ................ 142
19. Protocol for the SAPI COUNT Call ............................................... 143
20. Protocol for the SAPI BULK MODIFY Call ...................................... 145
22. Environment at Time of Call for SSI Function Code 80 ....................... 176
23. Environment at Time of Call for SSI Function Code 82, NJE Node Information Subfunction .................................................. 219
24. Environment at Time of Call for SSI Function Code 82, SPOOL Volume Information Subfunction .................................................. 235
25. Environment at Time of Call for SSI Function Code 82, Initiator Information Subfunction .................................................. 249
26. Environment at Time of Call for SSI Function Code 82, JESplex Information Subfunction .................................................. 264
27. Environment at Time of Call for SSI Function Code 82, Job Class Information Subfunction .................................................. 276
28. Initializing Your Subsystem ....................................................... 297
29. Input to the Initialization Routine ............................................... 303
30. Subsystem Affinity Service ....................................................... 313
31. Environment on Entry to the Function Routine for SSI Function Code 4 .................. 320
32. Environment on Entry to the Function Routine for SSI Function Code 8 .................. 324
33. Environment for a Single-line WTO in the SSWT Control Block ............... 331
34. Environment for the First Line of a Multi-line WTO in the SSWT Control Block ............... 334
35. Environment for Minor Lines of a Multi-line WTO in the SSWT Control Block ............... 336
36. Environment for a WTOR in the SSWT Control Block ....................... 338
37. Environment on Entry to the Function Routine for SSI Function Code 10 .................. 343
38. Environment on Entry to the Function Routine for SSI Function Code 48 .................. 352
39. Environment on Entry to the Function Routine for SSI Function Code 50 .................. 356
40. Environment on Entry to the Function Routine for SSI Function Code 54 .................. 359
41. Environment at Time of Call for SSI Function Code 58 ....................... 365
42. Environment at Time of Call for SSI Function Code 78 ....................... 372
43. Continuation of Environment at Time of Call for SSI Function Code 78 .................. 373
| 1. | IBM-Defined Keywords | 68 |
| 2. | JES Job Information Services Request Types | 87 |
| 3. | Examples of jobs returned for SSS2JBIL when SSS2JBIH is blank | 155 |
| 4. | Summary of output data elements based on input type requested | 174 |
| 5. | Examples of jobs returned for STATJBIL when STATJBIH is blank | 184 |
| 6. | SSI Function Code 80 Filters | 191 |
| 7. | JES3 Unsupported Flags and Fields | 213 |
| 8. | JES Properties Request Types | 217 |
| 9. | SSI updating of multi-line messages | 337 |
| 10. | Relationship between System and User Criteria | 369 |
About this document

This document 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 document.

In addition, this document 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 document

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

This document 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 document

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 Chapter 1, “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:
- Chapter 2, “Making a Request of a Subsystem,” on page 7

If you are familiar with the SSI, and you are writing your own subsystem, see:
- Chapter 4, “Setting Up Your Subsystem,” on page 291
- Chapter 5, “Services for Building and Using Your Subsystem,” on page 301
- Chapter 6, “SSI Function Codes Your Subsystem Can Support,” on page 317

Where to find more information

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

Information updates on the web

For the latest information updates that have been provided in PTF cover letters and Documentation APARs for z/OS®, see the online document at:
http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/ZDOCAPAR
This document is updated weekly and lists documentation changes before they are incorporated into z/OS publications.
Summary of Changes

Summary of changes
for SA22-7642-09
z/OS Version 1 Release 11

This document contains information previously presented in z/OS MVS Using the Subsystem Interface, SA22-7642–08, which supports z/OS Version 1 Release 10.

New information: SSI Function Code 82 (JES Properties) is supported to allow a user-supplied program to obtain information about JES managed structures such as NJE nodes, SPOOL volumes, initiators, members in the JESPLEX, and job classes. See “JES Properties—SSI Function Code 82” on page 217.

Changed information:
• The Input Parameters and Output Parameters sections have been updated in “Scheduler Facilities Call - SSI Function Code 70” on page 77.
• The Output Parameters section has been updated in “JES Job Information Services—SSI Function Code 71” on page 87.
• The Input-Only Fields section has been updated in “Notify User Message Service Call — SSI Function Code 75” on page 128.
• The Input Parameters and Output Parameters have been updated in “Extended Status Function Call — SSI Function Code 80” on page 172.

You may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of changes
for SA22-7642-08
z/OS Version 1 Release 10
as updated April 2009

This document contains information previously presented in z/OS MVS Using the Subsystem Interface, SA22-7642–07, which supports z/OS Version 1 Release 10.

Changed information The Use Information, Environment, Input Parameters, and Output Parameter sections have been updated in “Extended Status Function Call — SSI Function Code 80” on page 172.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes
for SA22-7642-07
z/OS Version 1 Release 10
This document contains information previously presented in z/OS MVS Using the Subsystem Interface, SA22-7642, which supports z/OS Version 1 Release 9.

New information

The SSS2SENL and SSS2SENP input fields are created under the SSS2SEL5 input field in “SYSOUT Application Program Interface (SAPI) — SSI Function Code 79” on page 134.

Changed information

- The USE Information and Input Parameters sections have been updated in “Process SYSOUT Data Sets Call — SSI Function Code 1” on page 14.
- The Environment section has been updated in “User Destination Validation/Conversion — SSI Function Code 11” on page 40.
- The Format of the Variable Output section has been updated in “Request Subsystem Version Information Call — SSI Function Code 54” on page 60.
- The Use Information, Related SSI Codes, Input Parameters, and Return Code Information sections have been updated in “Scheduler Facilities Call - SSI Function Code 70” on page 77.
- The Minimum Authorization and Input-Only Fields (Optional) sections have been updated in “SYSOUT Application Program Interface (SAPI) — SSI Function Code 79” on page 134.
- The Use Information, Use Information for data set list requests, Input Register Information, Job Information Elements, and SYSOUT Information Elements sections have been updated in “Extended Status Function Call — SSI Function Code 80” on page 172.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of changes
for SA22-7642-06
z/OS Version 1 Release 9

This document contains information previously presented in z/OS MVS Using the Subsystem Interface, SA22-7642, which supports z/OS Version 1 Release 8.

New information

- “User Destination Validation/Conversion — SSI Function Code 11” on page 40 has been added.
- “Scheduler Facilities Call - SSI Function Code 70” on page 77 has been added.
- “JES Job Information Services— SSI Function Code 71” on page 87 previously documented the subfunction SPOOL Read Service. It now has documentation for four other subfunctions, see the following subfunctions for more information:
  - “JES2 Health Monitor Information” on page 94
  - “Job Class Information” on page 105
  - “Convert Device ID Service” on page 116
  - “Checkpoint Version Information Service” on page 121.

Changed information
• “Request Job ID Call — SSI Function Code 20” on page 49 has been updated. See Input Parameters for updates that have been made to the SSRRSECB field in the SSRR control block.

• “Request Subsystem Version Information Call — SSI Function Code 54” on page 60 has been updated. See Fixed Header Output Section.

• “Notify User Message Service Call — SSI Function Code 75” on page 128 has been updated. See the following sections for updates:
  – Input Parameters contains updates to the SSNU control block on input.
  – Output Parameters contains updates to the SSOBRETN contents and the SSNU contents.

• “SYSOUT Application Program Interface (SAPI) — SSI Function Code 79” on page 134 has been updated. See the following sections for updates:
  – “BULK MODIFY Requests” on page 143 has been updated.
  – “Environment” on page 147 contains an update to the Minimum Authorization.

• “Extended Status Function Call — SSI Function Code 80” on page 172 has been updated. See the following updates:
  – “Environment” on page 175 contains updates to the Minimum Authorization.
  – Input Parameters contains updates to IAZSSST contents.
  – Output Register Information contains updates to STATREAS contents, Job Queue Element Terse Section, Job Queue Element JES2 Terse Section, Job Queue Element Verbose Section, SYSOUT Element Terse Section, SYSOUT Element JES2 Terse Section, SYSOUT Element Verbose Section and SYSOUT Element Security Section (mapped by SAF token)

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.
Chapter 1. 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 section 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 Chapter 3, “SSI Function Codes Your Program Can Request,” on page 13 for more information on the services available to your program using directed requests.

Directed Request

![Directed Request Diagram]

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 on page 3 shows the processing for a broadcast request.

See Chapter 6, “SSI Function Codes Your Subsystem Can Support,” on page 317 for more information on the broadcast function codes your subsystem can support.
Controlling SSI Processing

The IEFJFRQ installation exit provides a way to examine and modify subsystem function requests. See z/OS 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 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 z/OS 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 z/OS MVS Programming: Authorized Assembler Services Guide for more information on the RESMGR macro.

If you decide that you need a subsystem, see Chapter 4, “Setting Up Your Subsystem,” on page 291 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 Chapter 9, “Using IEFJSVEC with Your Subsystem,” on page 403, however IBM recommends that you use the services described in Chapter 5, “Services for Building and Using Your Subsystem,” on page 301 instead of IEFJSVEC. These services provide an easier way to define or change the functional response of a subsystem.
Chapter 2. 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 Chapter 3, “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 Chapter 3, “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.

NOT Programming Interface Information

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

End of NOT Programming Interface 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 SSOBFRTRY. 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 (SSOBSIB) 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 (including a WLM-initiated job), a started task, or a time-sharing LOGON as follows:

- **Batch jobs**
  
  A batch job is initiated under the JES that selects the job, that is, either the primary or alternate JES. In a JES initiator, the initiator’s life-of-job SSIB contains the JES subsystem name. In a WLM initiator, the initiator’s life-of-job SSIB contains the master subsystem name. The SMF exits IEFACTRT, IEFUJI, IEFUSI, and IEFUTL receive control in the initiator’s environment with the initiator’s life-of-job SSIB active. If your SMF exit makes an SSI request that depends on JES, it will not be successful in a WLM initiator.

- **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 [z/OS MVS JCL Reference](https://www.ibm.com/support/knowledgecenter/SSS5GL_2.2.0/com.ibm.mvs.jcl.doc/welcome.html) 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 Chapter 3, “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, specify it 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 SSI request (defined by IBM, a vendor, or the installation) may require your program to set the SSIBSUSE field. That field is available for the subsystem to use for an SSIB that a program provides in response to the SSI request. A subsystem (whether defined by a vendor or the installation) must not use the SSIBSUSE field in the life-of-job SSIB.

7. Store the address of the SSIB in the SSOBSSIB field of the SSOB.

### 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
</tbody>
</table>
Making a Request

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.

![Diagram of making 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:

<table>
<thead>
<tr>
<th>Return Code (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>Successful completion — the request went to the subsystem.</td>
</tr>
</tbody>
</table>
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 Chapter 8, "Examples — Subsystem Interface Routines" shows a coding example of a routine making a request of a subsystem.
Chapter 3. 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.

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Requested Function</th>
<th>Subsystem*</th>
<th>Type of Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Process SYSOUT data sets</td>
<td>JES2/JES3</td>
<td>Directed</td>
</tr>
<tr>
<td>11</td>
<td>User Destination Validation/Conversion</td>
<td>JES2/JES3</td>
<td>Directed</td>
</tr>
<tr>
<td>15</td>
<td>Verify subsystem function</td>
<td>Master</td>
<td>Directed</td>
</tr>
<tr>
<td>20</td>
<td>Request job ID</td>
<td>JES2/JES3</td>
<td>Directed</td>
</tr>
<tr>
<td>21</td>
<td>Return job ID</td>
<td>JES2/JES3</td>
<td>Directed</td>
</tr>
<tr>
<td>54</td>
<td>Request subsystem version information</td>
<td>JES2/JES3/Master</td>
<td>Directed</td>
</tr>
<tr>
<td>70</td>
<td>Scheduler Facilities services</td>
<td>JES2/JES3</td>
<td>Directed</td>
</tr>
<tr>
<td>71</td>
<td>JES JOB information</td>
<td>JES2</td>
<td>Directed</td>
</tr>
<tr>
<td>75</td>
<td>Notify user message service</td>
<td>JES2/JES3</td>
<td>Directed</td>
</tr>
<tr>
<td>79</td>
<td>SYSOUT Application Program Interface (SAPI)</td>
<td>JES2/JES3</td>
<td>Directed</td>
</tr>
<tr>
<td>80</td>
<td>Extended Status Function Call</td>
<td>JES2/JES3</td>
<td>Directed, Broadcast</td>
</tr>
<tr>
<td>82</td>
<td>JES Properties</td>
<td>JES2/JES3</td>
<td>Directed</td>
</tr>
</tbody>
</table>

*Not all supported levels of the IBM subsystems support all the function codes available with the current release of z/OS.

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-JES2/JES3 environment. This section contains detailed descriptions of the SSI function codes listed at the beginning of this chapter.

See example 5 in Chapter 8, "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 30 for detailed information on the definition of a standard external writer. See also z/OS JES Application Programming 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 z/OS 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:

<table>
<thead>
<tr>
<th>Minimum Authorization</th>
<th>Supervisor state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatchable unit mode</td>
<td>Task</td>
</tr>
<tr>
<td>AMODE</td>
<td>24-bit or 31-bit</td>
</tr>
<tr>
<td>Cross memory mode</td>
<td>PASN=HASN=SASN</td>
</tr>
<tr>
<td>ASC mode</td>
<td>Primary</td>
</tr>
<tr>
<td>Interrupt status</td>
<td>Enabled for I/O and external interrupts</td>
</tr>
<tr>
<td>Locks</td>
<td>No locks held</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 1 (SSOBSOUT)</td>
</tr>
</tbody>
</table>
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 about the life-of-job SSIB.

SSOBINDV  Address of the function dependent area (SSSO 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, your external writer must provide an SSIB and set the following fields in the SSIB control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier ‘SSIB’</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>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.</td>
</tr>
<tr>
<td>SSIBJBID</td>
<td>Job identifier — the job ID that was returned upon completion of the Request Job ID call (SSI function code 20).</td>
</tr>
<tr>
<td>SSIBSUSE</td>
<td>(JES3 only) Subsystem use — the SSIBSUSE value that was returned upon completion of the Request Job ID call (SSI function code 20).</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSOSEN</td>
<td>Length of the SSSO control block — set with SSSOSIZE value.</td>
</tr>
</tbody>
</table>
| SSSOUFLG   | User Selection Flags — defines the operation this call performs. The following options are available: 
  **Flag Value is X’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 SSSOCCTRL 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 28 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 SSSOCLAS.

**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 SSOSFRM bit must also be set to indicate selection by either 4-character or 8-character forms.

**SSSOFLG2** Flag byte
SSI Function Code 1

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 SSSUSER and SSSOWTRN bits.

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

If your external writer sets the SSSUSER bit, the value contained in the SSSOPGMN field is a userid. Setting the SSSUSER 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 [z/OS Security Server RACF Security Administrator's Guide](https://publib.boulder.ibm.com/infocenter/iseries/v2r1/index.jsp).

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 SSOUSER 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 SSOUSER 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 SSOFLGA 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 SSOUSER bit in the SSOFLGA 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 SSOSPGM bit set on but have not set the SSSOWTRN bit or the SSOUSER 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 28 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.

SSSOWTRC  Pointer to writer communications area.

For the initial retrieval request, your external writer sets this field to binary zeros. For JES3 only, for subsequent requests, your external writer must not change the SSSOWTRC field.

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
IBM recommends that you specify SOEXT=YES on the IEFSSSO invocation, as additional information is returned to the external writer.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSOFLG5</td>
<td>Flag byte</td>
</tr>
<tr>
<td></td>
<td>Your external writer can set one or more of the following selection bits:</td>
</tr>
<tr>
<td></td>
<td>• SSSOTKR — Security token length and security token version information set.</td>
</tr>
<tr>
<td></td>
<td>This bit determines whether the caller has supplied the security token length and security token version information in the field pointed to by SSSOEXIT. JES provides the security token of the returned data set (mapped to the requested version and length) upon return from the retrieval request. See <a href="https://www.ibm.com/support/docview/z?rs=5040&amp;id=SS98-0200">z/OS Security Server RACROUTE Macro Reference</a> for more information.</td>
</tr>
<tr>
<td>SSSOEXIT</td>
<td>Address of a caller-supplied area that contains a security token (returned only if your external writer specifies the SSSouser bit).</td>
</tr>
<tr>
<td></td>
<td>If the SSSOTKR 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 SSSOEXIT field. JES returns the security token in the format specified. See the SSSOTKR bit and the SSSOTKNG bit on output for additional information.</td>
</tr>
<tr>
<td></td>
<td>If the SSSOTKR bit has also been set off:</td>
</tr>
<tr>
<td></td>
<td>• The returned token is at the current level of the security authorization facility (SAF) security tokens</td>
</tr>
<tr>
<td></td>
<td>• The external writer is responsible for providing enough storage for the transfer to be made.</td>
</tr>
<tr>
<td>SSSOFOR8</td>
<td>8-character form name selected</td>
</tr>
<tr>
<td></td>
<td>Your external writer must have set both the SSOSFRM and SSOSFR8 bits for this selection to occur.</td>
</tr>
<tr>
<td></td>
<td>This field contains an 8-character form name that is left-justified and padded to the right with blank (X'40') characters.</td>
</tr>
<tr>
<td></td>
<td>If the SSOSFR8 bit is also set, the 4-character form name in SSOSFRM is ignored. JES uses the name in the SSSOFOR8 field as the forms selection criteria.</td>
</tr>
</tbody>
</table>

Your external writer must set all other fields in the SSO 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>
SSI Function Code 1

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 32 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.

SSS OIDST (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 [z/OS 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
SSI Function Code 1

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.

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSOFLG2</td>
<td>Flag byte</td>
</tr>
<tr>
<td></td>
<td>• SSSODDST — DD name set in the extension.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>SSSOCOPY</td>
<td>Number of copies.</td>
</tr>
<tr>
<td></td>
<td>JES provides the data set to your external writer as many times as the copy count from the creating JCL specifies it.</td>
</tr>
<tr>
<td></td>
<td>The value of this field depends on which JES is being used:</td>
</tr>
<tr>
<td></td>
<td>JES2: This field is always set to ‘1’ on a retrieval request.</td>
</tr>
<tr>
<td></td>
<td>JES3: This field is always set to ‘1’ on a retrieval request.</td>
</tr>
<tr>
<td>SSSOJOBN</td>
<td>Job name associated with the returned data set (retrieval request).</td>
</tr>
<tr>
<td>SSSOJOBI</td>
<td>Job ID associated with the returned data set (retrieval request).</td>
</tr>
<tr>
<td>SSSOCLAS</td>
<td>Class associated with the returned data set (retrieval request).</td>
</tr>
<tr>
<td></td>
<td>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).</td>
</tr>
<tr>
<td>SSSOMLRL</td>
<td>Maximum logical record length associated with the returned data set.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>SSSODEST</td>
<td>Destination associated with the returned data set (retrieval request).</td>
</tr>
<tr>
<td>SSSOPGMN</td>
<td>Writer name or userid associated with the returned data set (retrieval request, if available).</td>
</tr>
<tr>
<td></td>
<td>The specific information returned depends on the setting of the SSSOWTRN or SSSOUSER bits (retrieval request).</td>
</tr>
<tr>
<td></td>
<td>JES2: If neither the SSSOWTRN or SSSOUSER bits are specified, then this field contains the writer name associated with this data set.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>SSSODSN</td>
<td>Returned data set name (retrieval request).</td>
</tr>
</tbody>
</table>
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 28 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 32 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 IEFSSSO invocation. The external writer sets the SSSOPSEE bit. IBM recommends that you specify SOEXT=YES on the IEFSSSO invocation, as additional information is returned to the external writer.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSOFLG5</td>
<td>Flag byte</td>
</tr>
</tbody>
</table>
|            | • 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. |
SSI Function Code 1

- 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.

**SSOOSECT**
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:

```c
INDCB DCB DSORG=PS,MACRF=GL,BUFNO=1,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 [z/OS DFSMS Macro Instructions for Data Sets](https://www.ibm.com/support/knowledgecenter/STQDQG_6.1.0/com.ibm.sla.pdf/contents/sli_gen000.htm).

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.
SSI Function Code 1

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 28 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:
SSI Function Code 1

- 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.
• The installation dependent value from JMRUSEID. 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.

```plaintext
SSIREQ01 TITLE '- DOCUMENTATION'
SSIREQ01 AMODE 31
SSIREQ01 RMODE 24
SLEVEL 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 'PRT03'. *
* (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) *
* *
*********************************************************************
```
Chapter 3. SSI Function Codes Your Program Can Request 35

Chapter 3. SSI Function Codes Your Program Can Request

SSI Function Code 1

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 '- SSBO'
      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 OBTAINED, 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 SSBO
USING SSOBGN,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'SSBO' 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 SSBO ADDRESS

TITLE '- SSOB PROCESSING'

*********************************************************************
*  NOW WORK ON THE SSO. SELECT A SELECTION CRITERIA BASES ON  *
*  AN EXTERNAL WRITER NAME OF 'ANDREW'.                         *
*********************************************************************

XC SSOBGN(SSSOsize),SSOBGN CLEAR THE SSO
MVC SSOSLEN,=AL2(SSSOsize) SET THE SIZE OF THE SSO
MVI SSOSVER,SSSOVER SET THE VERSION NUMBER
OI SSOSFGL1,SSOSPGM+SSSOHLD SELECT BY WRITER NAME AND
*    THE HOLD QUEUE
*    SSOSFGLA,SSSOWTRN IND. THAT SELECTION IS BY
*    WRITER NAME, NOT USERID
*    SSOSPGMN,=CL8'ANDREW' IND. CORRECT WRITER NAME
*    THAT IS USED AS SELECTION
OI SSOSFLG2,SSSOPSEE IND. LONG FORM OF IEFSSSO

*********************************************************************
*  NOW GO TAP JES ON THE SHOULDER FOR A DATASET!                *
*********************************************************************

NEXTDS DS 0H GET NEXT DNAME FROM JES
MODESET MODE=SUP GET INTO SUPERVISOR STATE
LR R1,R2 R1=ADDRESS OF SSOB
SSI Function Code 1

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,BADRRET  ASSUME BAD SSORETN
ICM R9,'B1111',SSORETN  CHECK OUT SSORETN
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,BADRLLEN  ASSUME SIZE TOO LARGE FOR WTO
SR R6,R6  CLEAR REG 6
ICM R6,'B0011',SSSOMRLRL  GET MAX RECORD LENGTH
CH R6,'H150'  IS MAX RCD LENGTH>150??
BC GT,ABEND  YES - TIME FOR US TO GO HOME
STH R6,RECLN  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 TO PTRS
ST R1,S99TXTPP  STORED IN RB
LA R1,MY99RBPPT  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,SSSOMRLRL  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
SSIT031A DC A(SSIT031+IOHOBON) FOR MODE SWITCHING
*********************************************************************
* RETURN TO 31 BIT MODE AND CONTINUE                           *
*********************************************************************
SSIT031 DS 0H
EX R6,MOVEIT MOVE UP TO 150 BYTES OF REC
LA R11,RECLEN POINT TO RECORD FOR OUTPUT
WTO TEXT=(11),ROUTCODE=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

MYEOAD DS 0H END-OF-DATASET
CLOSE IND CB CLOSE THE INPUT DCB
DROP R4 IHADC B

*********************************************************************
* UNALLOCATE THE SYSOUT DATASET, CHANGING CLASS + DESTINATION     *
*********************************************************************

XC S99RB(RBLEN),S99RB ZERO THE RB
MVI S99RLN,RBLEN RB LENGTH
MVI S99VRB,S99VRBUN RB VERB CODE=UNALLOC
LA R1,MY99PTU ADDR SVC 99 ALLOC TU PTRS
ST R1,S99TXTPP STORED IN RB
LA R1,MY99RBP PT TO RB POINTER

DYNAALLOC 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

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,SSSONTRC 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 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.  RB CONTAINS THE PROGRAM REASON CODE, *
* R9 CONTAINS SPECIFIC ERROR/REASON CODE AS RETURNED BY THE      *
* SERVICE ROUTINE.                                                *
*********************************************************************
ABEND  DS  0H   ISSUE THE ABEND MACRO
       (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.                                                   *
*********************************************************************
MY99RBPT  DC  A(EQUIEHOB+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(EQUIEHOB+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(EQUIEHOB+TXTDUNDS) CHANGE THE DEST TU
*********************************************************************
* TEXT UNIT POINTERS FOR ALLOCATION
*********************************************************************
TXTDALDS  DC  AL2(DALDSNAM)   DATASET NAME KEY
             DC  X'0001'   NUMBER
             DC  AL2(44)   DNAME LENGTH
TXTDSNAM  DS  CL44' '   DNAME 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
TXTCLSEDCDC  DC  AL2(DALRTDDN) RETURN DDNAME FIELD
             DC  X'0001'   # FIELD (0001 REQUIRED)
             DC  X'0008'   LEN OF PARM
TXTDALDCDC  DC  CL8' '   RETURNED DDNAME PARM FIELD
*********************************************************************
* TEXT UNIT POINTERS FOR UNALLOCATION
*********************************************************************
TXTDUNDD  DC  AL2(DUNDDNAM)   TU FOR DDNAME UNALLOC
             DC  X'0001'   NUMBER
DC AL2(8) DDNAME LENGTH
TXTDUN99 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
TXTDUNHS 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(SSSDEOOS) 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, DDNAME=WILLCHNG
TITLE '- LITERALS'
LTORG ,
END

Chapter 3. SSI Function Codes Your Program Can Request 39
User Destination Validation/Conversion — SSI Function Code 11

The user destination validation/conversion (SSI function code 11) provides a requesting program the ability to convert and/or validate a remote destination.

Type of Request
Directed SSI call.

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
- IEFSSUS

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 SSUS control blocks can reside in storage above or below 16 megabytes.
Recovery: The caller should provide an ESTAE-type recovery environment. See z/OS MVS Programming: Assembler Services Guide for more information on an ESTAE-type recovery environment.

Figure 5 on page 41 shows the environment at the time of the call for SSI function code 11.
Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
</tbody>
</table>

**Input Parameters**

Input parameters for the function routine are:
- SSOB
- SSIB
- SSUS

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSize) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 11 (SSOBFUNC)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of an SSIB control block or zero (if this field is zero, the life-of-job SSIB is used). See &quot;Subsystem Identification Block (SSIB)&quot; on page 8 for more information on the life-of-job SSIB.</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function dependent area (SSUS control block)</td>
</tr>
</tbody>
</table>

Set all other fields in the SSOB control block to binary zeros before issuing the IEFSSREQ macro.
SSI Function Code 11

**SSI 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem to which this user destination validation/conversion 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 user destination validation/conversion. You must use the same subsystem name in this SSIBSSNM field as you used for the Request Job ID call.</td>
</tr>
<tr>
<td>SSIBSUSE</td>
<td>(JES3 only) Subsystem use – the SSIBSUSE value that was returned upon completion of the Request Job ID call (SSI function code 20).</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSUSLEN</td>
<td>Length of the SSUS (SSUSIZE) control block</td>
</tr>
<tr>
<td>SSUSFLG1</td>
<td>Flag Byte</td>
</tr>
<tr>
<td>SSUS1NOD</td>
<td>Return the node name</td>
</tr>
<tr>
<td>SSUSCVXE</td>
<td>Destination conversion extension exists</td>
</tr>
<tr>
<td>SSUSVER</td>
<td>Version of mapping for the caller – Set this field to SSUSCVER (an IBM-defined integer constant within the SSUS control block).</td>
</tr>
<tr>
<td>SSUSUSER</td>
<td>Remote destination to be verified.</td>
</tr>
<tr>
<td>SSUSFLG2</td>
<td>Conversion flag byte (JES2 only)</td>
</tr>
<tr>
<td>SSUS1TO2</td>
<td>Convert one 18 byte field to two 8 byte fields. The input field is SSUSDEST and the output fields are SSUSDST1 and SSUSDST2.</td>
</tr>
<tr>
<td>SSUS2TO1</td>
<td>Convert two 8 byte fields to one 18 byte field. The input fields are SSUSDST1 and SSUSDST2 and the output field is SSUSDEST.</td>
</tr>
<tr>
<td>SSUSITOC</td>
<td>Convert a 4 byte internal destination with a 8 byte user destination to a 18 byte character field. The input fields are SSUSIDST and SSUSUDST and the output field is SSUSDEST.</td>
</tr>
</tbody>
</table>
SSUSCTOI
Convert an 18 byte field to a 4 byte internal destination with an 8 byte user destination. The input field is SSUSDEST and the output fields are SSUSIDST and SSUSUDST.

SSUSGENC
Generic characters (‘*’ and ‘?’) are to be allowed as part of a user destination. This is only valid if SSUS1TO2, SSUS2TO1 or SSUSCTOI is set.

SSUSIPAD
IP-format destination included. Only valid if SSUS1TO2, SSUSCTOI or SSUS2TO1 is set.

SSUSDEST
Destination when SSUS1TO2 or SSUSCTOI is set. If SSUSIPAD is set, the first four bytes must contain the address of the full destination.

SSUSDLEN
Length of input destination when SSUSIPAD is set.

SSUSDST1
Destination part 1 when SSUS2TO1 is set.

SSUSDST2
Destination part 2 when SSUS2TO1 is set. If SSUSIPAD is set, the first four bytes must contain the address of the full destination.

SSUSDSLN
Length of input destination when SSUSIPAD is set.

SSUSIDST
Internal destination, if SSUSITOC is set.

SSUSUDST
User destination, if SSUSITOC is set.

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

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>The user destination validation/conversion was processed. Check the SSOBRETN field for specific function information.</td>
</tr>
<tr>
<td>SSRTNSUP (4)</td>
<td>The subsystem specified in the SSIBSSNM field does not support this function</td>
</tr>
<tr>
<td>SSRTNTUP (8)</td>
<td>The subsystem specified in the SSIBSSNM field exists, but is not active.</td>
</tr>
</tbody>
</table>
SSI Function Code 11

**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
- SSUS

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

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSUSRTOK (0)</td>
<td>Valid request.</td>
</tr>
<tr>
<td>SSUSNOUS (4)</td>
<td>Invalid destination.</td>
</tr>
<tr>
<td>SSUSINCP (8)</td>
<td>Subsystem could not complete the validity check.</td>
</tr>
</tbody>
</table>

**SSUS Contents:** Various fields can be output fields depending on which conversion was requested:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSUSDEST</td>
<td>Destination if SSUS2TO1 or SSUSITOC is set.</td>
</tr>
<tr>
<td>SSUSDST1</td>
<td>Destination part 1 if SSUS1TO2 is set.</td>
</tr>
<tr>
<td>SSUSDST2</td>
<td>Destination part 2 if SSUS1TO2 is set.</td>
</tr>
<tr>
<td>SSUSIDST</td>
<td>Internal destination if SSUSCTOI is set.</td>
</tr>
<tr>
<td>SSUSUDST</td>
<td>User destination if SSUSCTOI is set.</td>
</tr>
</tbody>
</table>
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 294.

For more information, see "Maintaining Information About the Callers of Your Subsystem" on page 313.

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 313 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:

<table>
<thead>
<tr>
<th>Minimum Authorization</th>
<th>Problem state, any PSW key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatchable unit mode</td>
<td>Task or SRB</td>
</tr>
<tr>
<td>AMODE</td>
<td>24-bit or 31-bit</td>
</tr>
<tr>
<td>Cross memory mode</td>
<td>PASN=HASN=SASN</td>
</tr>
<tr>
<td>ASC mode</td>
<td>Primary</td>
</tr>
<tr>
<td>Interrupt status</td>
<td>Enabled for I/O and external interrupts</td>
</tr>
<tr>
<td>Locks</td>
<td>No locks held</td>
</tr>
<tr>
<td>Control Parameters</td>
<td>The SSOB, SSIB, and SSVS control blocks can reside above or below 16 megabytes.</td>
</tr>
</tbody>
</table>
Recovery

The caller should provide an ESTAE-type recovery environment. See *z/OS MVS Programming: Assembler Services Guide* for more information on an ESTAE-type recovery environment.

Figure 6 shows the environment at the time of the call for SSI function code 15.

**Figure 6. 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
<tr>
<td>SSOBLLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 15 (SSOBVERS)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of an SSIB control block or zero (if this field is zero, the life-of-job SSIB is used). See &quot;Subsystem Identification Block (SSIB)&quot; on page 8 for more information on the life-of-job SSIB.</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function dependent area (SSVS control block)</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem that this verify subsystem function call is directed to (MSTR).</td>
</tr>
<tr>
<td>SSIBJBID</td>
<td>Name of the subsystem to be verified</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSVSLEN</td>
<td>Length of the SSVS (SSVSSIZE) control block</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>
SSI Function Code 15

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
SSRTO (0)           The Verify Subsystem function call completed. Check the SSOBRETN field for specific function information.
SSRNTSUP (4)       The subsystem specified in the SSIBSSNM field does not support the Verify Subsystem function call.
SSRNTNUP (8)       The subsystem specified in the SSIBSSNM field exists, but is not active.
SSRNOSS (12)       The subsystem specified in the SSIBSSNM field is not defined to MVS.
SSRDTDIST (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
SSVS (0)                 Valid subsystem name
SSVJB (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 313 for more information on the SSAFF macro.
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 z/OS 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 JOBCLASS(STC) 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 JOBCLASS(STC) 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:

<table>
<thead>
<tr>
<th>Minimum Authorization</th>
<th>Supervisor state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatchable unit mode</td>
<td>Task</td>
</tr>
<tr>
<td>AMODE</td>
<td>24-bit or 31-bit</td>
</tr>
<tr>
<td>Cross memory mode</td>
<td>PASN=HASN=SASN</td>
</tr>
<tr>
<td>ASC mode</td>
<td>Primary</td>
</tr>
<tr>
<td>Interrupt status</td>
<td>Enabled for I/O and external interrupts</td>
</tr>
<tr>
<td>Locks</td>
<td>No locks held</td>
</tr>
<tr>
<td>Control Parameters</td>
<td>The SSOB, SSIB, and SSRR control blocks can reside in storage above 16 megabytes.</td>
</tr>
</tbody>
</table>

**Recovery**

The caller should provide an ESTAE-type recovery environment. See [z/OS MVS Programming: Authorized Assembler Services Guide](https://www.ibm.com/support/knowledgecenter/ST40aghan/com.ibm.zos.r2.2.9/cm_mvsasg/zasas_overview.html) for more information on an ESTAE-type recovery environment.

Figure 7 on page 51 shows the environment at the time of the 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

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:
### SSI Function Code 20

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 20 (SSOBREQST)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of an SSIB control block</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function dependent area (SSRR control block)</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>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.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRRLEN</td>
<td>Length of the SSRR (SSRRSIZE) control block</td>
</tr>
<tr>
<td>SSRRFLG1</td>
<td>Flag byte</td>
</tr>
</tbody>
</table>

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 SSRRVCVER (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). Normal $PJES2 processing will hang if the application does not issue this call. JES2 will also issue message HASP715 when the proper Return Job ID call is not made in a timely manner to alert the operator of a Return Job ID call being needed.

**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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>The Request Job ID call completed. Check the SSOBRETN field for specific function information.</td>
</tr>
<tr>
<td>SSRTNSUP (4)</td>
<td>The subsystem specified in the SSIBSSNM field does not support this function.</td>
</tr>
<tr>
<td>SSRTNTUP (8)</td>
<td>The subsystem specified in the SSIBSSNM field exists, but is not active.</td>
</tr>
<tr>
<td>SSRTNOSS (12)</td>
<td>The subsystem specified in the SSIBSSNM field is not defined to MVS.</td>
</tr>
</tbody>
</table>
SSRRTDIST (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:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRROK (0)</td>
<td>Successful completion. JES assigned a job ID to the caller. The job ID is available in the SSIBJBID field. See &quot;Restrictions&quot; on page 55 for information on the processing that takes place after successful completion has been obtained.</td>
</tr>
<tr>
<td>SSRRFAIL (4)</td>
<td>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.</td>
</tr>
<tr>
<td>SSRRFREQ (8)</td>
<td>The Request Job ID call is already known to this JES, and may not have a second job ID established.</td>
</tr>
<tr>
<td>SSRRNNOEC (16)</td>
<td>For JES2 only, an ECB was not supplied through the SSRRSECB pointer field on the Request Job ID call.</td>
</tr>
<tr>
<td>SSRRPUME (20)</td>
<td>There is an error in the SSRR data area. For example, both the SSRRJOBL bit and the SSRRNJBL bits may be set.</td>
</tr>
<tr>
<td>SSRRPERR (36)</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.
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 z/OS MVS Programming: Authorized Assembler Services Guide for more information on an ESTAE-type recovery environment.

Figure 8 shows the environment at the time of the 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 21 (SSOBFRTRN)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of an SSIB control block</td>
</tr>
</tbody>
</table>

Figure 8. Environment at Time of Call for SSI Function Code 21
SSI Function Code 21

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier ‘SSIB’</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>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).</td>
</tr>
<tr>
<td>SSIBJBID</td>
<td>Returned job ID</td>
</tr>
<tr>
<td>SSIBSUSE</td>
<td>(JES3 only) Subsystem use — the SSIBSUSE value that was returned upon completion of the Request Job ID call (SSI function code 20).</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRRLEN</td>
<td>Length of the SSRR (SSRRSIZE) control block</td>
</tr>
<tr>
<td>SSRRVER</td>
<td>Version of mapping for the caller. Set this field to SSRRCVER (an IBM-defined integer constant within the SSRR control block).</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>
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.
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 z/OS 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)

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
Dispatchable unit mode
AMODE
Cross memory mode
ASC mode
Interrupt status
Locks
Control Parameters

Problem state, any PSW key
Task
24-bit or 31-bit
PASN=HASN=SASN
Primary
Enabled for I/O and external interrupts
No locks held
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 Z/OS MVS Programming: Assembler Services Guide for more information on an ESTAE-type recovery environment.

Figure 9 on page 62 shows the environment at the time of the 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function Code 54 (SSOBSSVI)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of the SSIB control block or zero (if this field is zero, the life-of-job SSIB is used). See &quot;Subsystem Identification Block (SSIB)&quot; on page 8 for more information on the life-of-job SSIB.</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function dependent area (SSVI control block)</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSVILEN</td>
<td>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.</td>
</tr>
<tr>
<td>SSVIVER</td>
<td>Version of mapping for the caller — Set this field to SSVICVER (an IBM-defined integer constant within the SSVI control block).</td>
</tr>
<tr>
<td>SSVIID</td>
<td>Identifier 'SSVI'</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
</tbody>
</table>
SSI Function Code 54

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.

**SSVIABLLOG (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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSVIRLEN</td>
<td>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).</td>
</tr>
</tbody>
</table>

To determine whether the SSVIRLEN field contains returned or required storage, check the return code in SSOBRETN, which indicates:

<table>
<thead>
<tr>
<th>Decimal Value Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSVIOK (0)</td>
</tr>
<tr>
<td>Request was successful. The SSVIRLEN field contains the length, in bytes, of the returned data.</td>
</tr>
</tbody>
</table>

| SSVINSTR (8)          |
| Request failed. The caller did not specify enough storage. |
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 SSVIABL (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, OS 1.1.0, OS 2.1.0, z/OS 1.4, or z/OS 1.9. The master subsystem returns the same value as that contained in CVTPRODN.

**SSVFIMID**
An 8-byte character field that specifies the FMID of the subsystem (for example, HBB5510, HJE5510, HJS5511, HJE7730, HJS7730, or HBB7730).

**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 OS/390® Release 10 JES2, the relative subsystem product level value is decimal ‘32’ and for z/OS Release 7 JES2, the relative subsystem product level value is decimal ‘36’. For more information, see topic “Determining the JES2 Release Level” in z/OS 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 z/OS 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 [z/OS JES2 Installation Exits](z/OS JES2 Installation Exits) or [z/OS JES3 Customization](z/OS 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."

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**
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."

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 [z/OS JES2 Installation Exits]. For more information about using JES3 IATUX63, see [z/OS JES3 Customization].

**Format of the Variable Output Sections**
The following is a description of the subsystem and installation variable output sections:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSVIVLEN</td>
<td>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.</td>
</tr>
<tr>
<td>SSVIDAT</td>
<td>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 [Table 1 on page 68](Table 1 on page 68) are returned to the SSI code 54 caller.</td>
</tr>
</tbody>
</table>
**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:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>,AUTO_RESTART_MANAGER='YES</td>
<td>NO'</td>
</tr>
<tr>
<td>,CLIENT_PRINT='YES'</td>
<td>Indicates that the JES supports the creation of a client token in support of client printing.</td>
</tr>
<tr>
<td>,COMMAND_PREFIX='prefix'</td>
<td>Indicates the operator command prefix that is registered for this subsystem. For JES2, this is the value of CONDEF CONCHAR=. JES3 supplies the first system scoped synonym.</td>
</tr>
<tr>
<td>,DYNAMIC_OUTPUT='YES</td>
<td>NO'</td>
</tr>
<tr>
<td>,EXW_SYSOUT_CLASS='classes'</td>
<td>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.</td>
</tr>
<tr>
<td>,FOUR_DIGIT_DEVNUMS='YES</td>
<td>NO'</td>
</tr>
<tr>
<td>,GLOBAL_PLEVEL='mmm'</td>
<td>The JES3 global product level in decimal EBCDIC digits. (JES3 only).</td>
</tr>
<tr>
<td>,GLOBAL_SLEVEL='mmm'</td>
<td>The JES3 global service level in decimal EBCDIC digits. (JES3 only).</td>
</tr>
<tr>
<td>,GLOBAL_RELEASE='release'</td>
<td>The JES3 release running on the JES3 Global. (JES3 only.)</td>
</tr>
<tr>
<td>,GLOBAL='system name'</td>
<td>The system name of the JES3 Global. (JES3 only).</td>
</tr>
<tr>
<td>,INITIATOR_RESTART='YES</td>
<td>NO'</td>
</tr>
<tr>
<td>,JES_NODE='name'</td>
<td>Specifies the network node name of the JES.</td>
</tr>
<tr>
<td>,JES_MEMBERNAME='name'</td>
<td>Specifies the member name of a particular JES2 in a multi-JES configuration or the JES3 main name in a JES3 complex.</td>
</tr>
<tr>
<td>,MULTIPLE_STCSTSO='YES</td>
<td>NO'</td>
</tr>
<tr>
<td>,PLEXSYN='list of plex synonyms'</td>
<td>The list of sysplex scoped command prefix synonyms from the CONSTD PLEXSYN= definition. (JES3 only.)</td>
</tr>
<tr>
<td>,SAPI_CHARS='NO'</td>
<td>Indicates selection by characters not supported.</td>
</tr>
</tbody>
</table>
Table 1. IBM-Defined Keywords (continued)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>'SAPI_IP_SELECT='NO'</td>
<td>Indicates selection by IP address (Internet protocol) not supported.</td>
</tr>
<tr>
<td>'SAPI_MOD_SELECT='NO'</td>
<td>Indicates selection by modification id not supported.</td>
</tr>
<tr>
<td>'SAPI_PRTY_SELECT='NO'</td>
<td>Indicates selection by priority not supported.</td>
</tr>
<tr>
<td>'SAPI_VOL_SELECT='NO'</td>
<td>Indicates selection by volume not supported.</td>
</tr>
<tr>
<td>'SAPI='YES'</td>
<td>Indicates SAPI is supported by this JES.</td>
</tr>
<tr>
<td>'SPOOL_BROWSE='YES'</td>
<td>Indicates that spool browse is supported on this release of JES3. (JES3 only. On JES2 all releases support spool browse.) The text SPOOL_BROWSE='NO' is never supplied. On the JES3 releases that do not support spool browse, the entire SPOOL_BROWSE keyword is omitted.</td>
</tr>
<tr>
<td>'SYN='list of system synonyms'</td>
<td>The list of system scoped command prefix synonyms from the CONSTD SYN= definition. (JES3 only.)</td>
</tr>
<tr>
<td>'TSO_SYSOUT_CLASS='classes'</td>
<td>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.</td>
</tr>
<tr>
<td>'WTR_SYSOUT_CLASS='classes'</td>
<td>Indicates the SYSOUT class for which output is placed on the JES3 writer queue. (See note.) For JES2, classes are for non-held SYSOUT.</td>
</tr>
</tbody>
</table>

**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
  SLEVEL 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 IFFSSREQ/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 'FF' ALL BITS ON

EQUHOBON EQU '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
**SSI Function Code 54**

```
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 IEFSSOB 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
LDSTART EQU *        START OF LOCAL DATA AREA
LDASA DS 9D           SAVE AREA FOR LOWER CALLERS
LDAILDS DS CL8'LDAAREA ' IDENTIFICATION OF LDA AREA
LDAPSSOB 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
LDAIZE EQU LDASEN-LDSTART 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
```
Chapter 3. SSI Function Codes Your Program Can Request

SSI Function Code 54

```
storage obtain,
length=ldasize,  get a save area that's reentrant  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,=clb'ldarea'  indication of local data area
wto 'ssireq54 executing v1', let op know  x
routcode=(2.11)
title '- ssof/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 oh
storage obtain,
length=(5),  get a save area that's reentrant  x
cond=no  unconditional req - no rc info
lr r4,r1  point to the ssvi
using ssvi,r4  addressability
space 2
******************************************************************************
* when issuing the iefsreq 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 *
* (lda0ssob) 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
0 r2,a(equhobon)  set high order bit on
st r2,lda0ssob  store for iefsreq'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 ssoblend,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(ssibsiz),ssibegin  clear ssib
mvc ssibid,='c'ssibid'  move identifier in
mvc ssiblen=y(ssibsiz)  move size of the ssib in
mvc ssibssnm,='c'mstr'  show master subsystem to be
* used to get the info
******************************************************************************
```
THESE STORAGE NOW IS

IEFSSREQ

THE DONE

ICM SPACE

B

STORAGE LH SPACE BC C ICM LA BC SPACE

IEFSSREQ

REGISTER SSOB, SIZE HAVE 2-. R2 R15,0 R6,B'1111',SSVIUDOF R9,B'1111',SSOBRETN R5,SSVILEN R6,SSVIRLEN NE,ABEND NZERO,ABEND R1,LDA@SSOB R9,=A(SSVINSTR) ADDR=(4) LENGTH=(5),

IEFSSREQ MACRO IS READY TO GO.

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

*********************************************************************

********************************************************************
SSI Function Code 54

Chapter 3. SSI Function Codes Your Program Can Request

Chapter 3. SSI Function Codes Your Program Can Request

---

```assembly
LA R7,SSVI(R6)          R7==>USER VARIABLE DATA AREA

SHOWIT1 DS OH          RB=NUMBER OF CHARS TO DISPLAY

SHOWIT2 DS OH          RB=NUMBER OF CHARS TO DISPLAY

SHOWSYS DS OH         WTO 'SSIREQ54 returning', let OP know

SHOWSYS2 DS OH         WTO 'SSIREQ54 NO USER DATA PRESENT', let OP know

SHOWSYS3 DS OH         WTO 'SSIREQ54 NO SYSTEM DATA', let OP know

SHOWSYS2 BRANCH AROUND WTO

SPACE,

RETURN DS OH          STORAGE RELEASE, FREE MY INFO AREA

SPACE,

SLR R15,R15          SET RETURN CODE OF ZERO

PR                RETURN TO CALLER USING STACK, SET RETURN CODE OF ZERO

ABEND (RB),DUMP,STEP LET THE USER IN ON THE BAD NEWS
```

---

GIVE BACK THE STORAGE WE BOUGHT EARLIER.

Set Program Return Code.

Return to Caller Using Stack, 2-14, Addressing Mode, Asc Mode, and Return to Caller

Abend Routines Follow

---

Chapter 3. SSI Function Codes Your Program Can Request

---
SSI Function Code 54

TITLE 'LOCAL DATA'
SPACE,
CLEAR XC 0(*=-,R4),0(R4)  CLEAR SSVI - OBJ OF EXECUTE
END ,
Scheduler Facilities Call - SSI Function Code 70

The scheduler facilities function (SSI function code 70) provides a requesting program the ability to modify or obtain those characteristics of sysout datasets that are controlled by subsystem maintained scheduler data (for example, SWBTU data).

Type of Request
Directed SSI call.

Use Information
Currently, SSI 70 supports the following function types:

- SWB Modify
- SWB Merge
- SWB Merge Cleanup

These functions allow a user to modify or obtain SWBTU data associated with a SYSOUT dataset. SWBTU data consists of dynamic output values such as ADDRESS, CLASS, COPIES, FORMS, NOTIFY, etc. The output values (‘keys’) that can be modified by SSI 70 are listed in the IEFDOKEY macro. See the z/OS MVS Data Areas book for a description of the IEFDOKEY data area.

Issued to
A JES2 or JES3 subsystem (either primary or secondary). The subsystem does not have to be associated with the requesting address space.

Related SSI Codes
Extended Status Function Call — SSI Function Code 80

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
- IAZSSSF

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 SSSF control blocks can reside in storage above or below 16 megabytes. |
Recovery

The caller should provide an ESTAE-type recovery environment. See **z/OS MVS Programming: 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 70.

![Diagram of Environment at Time of Call for SSI Function Code 70](image)

**Input Register Information**

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
</tbody>
</table>

**Input Parameters**

Input parameters for the function routine are:

- SSOB
- SSIB
- SSSF

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
</tbody>
</table>
SSOBLEN  Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC SSI function code 70 (SSOBSFS)
SSOBSIB 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 (SSSF 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem to which this Scheduler Facilities 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 scheduler facilities call. You must use the same subsystem name in this SSIBSSNM field as you used for the Request Job ID call.</td>
</tr>
</tbody>
</table>

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

**SSSF Contents:** The caller must set the following fields in the IAZSSSF control block on input to a Scheduler Facilities call:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSFID</td>
<td>Control block eyecatcher, set to 'SSSF'.</td>
</tr>
<tr>
<td>SSSFLEN</td>
<td>Length of the SSSF control block. The length is the sum of the header length size, SSSFHSZE, and the size of the request dependent area. For the SWB modify function, the size of the request dependent area is SSSFMRSZ. For the SWB merge and SWB cleanup functions, the size of the request dependent area is SSSFFMSZ.</td>
</tr>
<tr>
<td>SSFVER</td>
<td>Version of mapping for the caller – Set this field to SSSFVER (an IBM-defined integer constant within the SSSF control block).</td>
</tr>
<tr>
<td>SSSFREQF</td>
<td>Function request value – designates the function type requested. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>SSSFSWBM   SWB modify of output descriptors</td>
</tr>
<tr>
<td></td>
<td>SSSFSWB    SWB merge of output descriptors</td>
</tr>
<tr>
<td></td>
<td>SSSFSWBC   Return storage used for SWB merge function. Use this</td>
</tr>
</tbody>
</table>
function when all SWB merge calls are finished. The format of the SSOB extension is identical to that used for the SSSFSWBF function.

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

The following section describes the sub-function dependent area for the SWB modify subfunction. The caller must set the following fields in the IAZSSSF control block on input to the SWB modify subfunction.

For the SWB modify function, the caller can optionally set the following fields in the IAZSSSF control block on input to the scheduler facilities call.

**SSSFFLG1**

Flag Byte - request dependent indicator.

For the SWB modify function, this flag indicates the type of security authorization checking requested for this request. Note that only one of the following two flags should be turned on.

**SSSFDEST**

Perform a destination check. This check ensures that the user has SAF authority to the ISFAUTH resource (JES2 only).

**SSSFSECL**

Perform a seclabel dominance check to insure that the SECLABEL of the user dominates the SECLABEL of the SYSOUT. This request is honored for authorized callers, and the authorization check will be performed only if the appropriate security environment exists (JES2 only).

If neither of the above values has been specified in SSSFFLG1 for the SWB modify function, the default security authorization check ensures that the user has SAF authority to the JESSPOOL resource.

**SSSFMTYP**

Indicates the type of modify data is being passed. OFF means that individual job identification data is being passed (JES2 only). ON indicates that a client token is being passed.

The following fields should be filled in when bit SSSFMTYP in SSSFFLG1 is OFF.

**SSSFJBNM**

The 1-8 character jobname associated with the sysout that is to be modified.

**SSSFJBID**

The 1-8 character jobid associated with the sysout that is to be modified.

**SSSFGRPN**

The 1-8 character output group name associated with the sysout that is to be modified.

**SSSFGRP1**

Output group id 1 associated with the sysout that is to be modified.

**SSSFGRP2**

Output group id 2 associated with the sysout that is to be modified.

The following field should be filled in when bit SSSFMTYP in SSSFFLG1 is ON.
SSI Function Code 70

SSSFMDST
Address of client token. The token is either a data set level token for JES3 or a group level token for JES2.

A data set level token is returned in field STVSCTKN when a verbose output request is made using SSI 80 (Extended Status). In addition, the address of a data set token is returned in field SSS2DSTR for each data set returned by SSI 79 (SAPI). (JES3 only)

A client token is returned by the DYNALLOC macro. The text unit DALRTCTK (key 0071) will return an 80 byte JES Client Token as the data of the text unit.

JES3 requires the use of SSSFMDST.

The following fields are common to all types of modify requests.

SSSFCART
The 1-8 character command and response token (CART) to be used for WTO responses. (JES2 only)

SSSFCNID
The console identifier to be used for WTO responses. (JES2 only)

SSSFMDAD
Address of output descriptor modify list in SWBTU format. This list is mapped as follows:

- SJPRFX area, mapped by macro IEFSJPFX. Note that field SJPRVERB should be set to 'OUTPUT' to indicate that this is an OUTPUT descriptor.
- Area for text units. See macro IEFDOTUM for a mapping of each text unit.

For the SWB modify function, the caller must set at least one of the following two pairs of fields in the IAZSSSF control block to be non-zero on input to the scheduler facilities call. The pairs are:

- Pair 1: SSSFMDAD and SSSFMDLN
- Pair 2: SSSFEFAD and SSSFERLN

Within a pair, both fields must be non-zero (for example, if the user just wants the 'erase' function in pair 2, then SSSFEFAD and SSSFERLN must be non-zero). The user can choose to fill in pair 1, pair 2, or both pairs.

SSSFMDLN
Length of modify list – includes length of prefix area.

SSSSFERAD
Address of output descriptor Erase list in TU format. The erase list is a list of fullword fields, each of which consists of a two byte key value, (defined in macro IEFDOKEY), and two bytes of zeroes.

For the SWB modify function, the caller must set at least one of the following two pairs of fields in the IAZSSSF control block to be non-zero on input to the scheduler facilities call. The pairs are:

- Pair 1: SSSFMDAD and SSSFMDLN
- Pair 2: SSSFEFAD and SSSFERLN

Within a pair, both fields must be non-zero (for example, if the user just wants the 'erase' function in pair 2, then SSSFEFAD and SSSFERLN must be non-zero). The user can choose to fill in pair 1, pair 2, or both pairs.
SSSFERLN
Length of erase list.

For the SWB modify function, the caller can optionally set the following fields in the IAZSSSF control block on input to the scheduler facilities call.

SSSFFLG1
Flag Byte – request dependent indicator.

For the SWB modify function, this flag indicates the type of security authorization checking requested for this request. Note that only one flag should be turned on.

SSSFDEST
Perform a destination check. This check insures that the user has SAF authority to the ISFAUTH resource (SDSF class).

SSSFSECL
Perform a seclabel dominance check to insure that the SECLABEL of the user dominates the SECLABEL of the SYSOUT. This request is honored for authorized callers, and the authorization check will be performed only if the appropriate security environment exists.

If no value has been specified in SSSFFLG1 for the SWB modify function, the default security authorization check will insure that the user has SAF authority to the JESSPOOL resource.

SSSFCART
The 1-8 character command and response token (CART) to be used for WTO responses.

SSSFNCNID
The console identifier to be used for WTO responses.

The following section describes the sub-function dependent area for the SWB merge subfunction. The caller must set the following fields in the IAZSSSF control block on input to the SWB merge subfunction.

SSSFFDTK
Address of the data set token representing the data set. The dataset token can be obtained, for example, from SSI function code 80 (STVSCTKN) or SSI function code 79 (SSS2DSTR).

For the SWB merge function, the caller can optionally set the following fields in the IAZSSSF control block on input to the scheduler facilities call.

SSSFGGTK
Address of a group token representing the data set. This is an optional value. If given, it must be the address of the group token. The group token is in field STSTCTKN returned by SSI Function Code 80. (JES2 only)

SSSFFLG1
Flag Byte - request dependent indicator.

SSSFFSWB
Provide non-SWA SWBs in addition to SWBTUs.

SSSFCPAT
Return compatibility SWBs.

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 scheduler facilities call 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 incorrect lengths or formats.
SSRTNSSI(24) The SSI has not been initialized.

Output Parameters
Output parameters for the function routine are:
- SSOBRETN
- SSSFREAS
- SSSFMREA

All three SSI 70 functions have a common set of R15 and SSOBRETN values. However, each function has its own set of output parameters.

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
SSSFOK (0) Request processed successfully.
SSSFUERR (8) Request rejected, see reason code in field SSSFREAS.
SSSFEXTE (12) No extension found.
SSI Function Code 70

**SSI FNOST (16)**: No storage to process request.

**SSI FREAS Contents**: When control returns to the caller of the SWB modify function and field SSOBRETN = SSSFUERR (8), the SSSFREAS field contains one of the following decimal values:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSFNOJ2 (16)</td>
<td>JES2 not up and running.</td>
</tr>
<tr>
<td>SSSFINVF (20)</td>
<td>Invalid function request.</td>
</tr>
<tr>
<td>SSSFINVE (24)</td>
<td>Invalid SSSF extension.</td>
</tr>
<tr>
<td>SSSFNOAU (32)</td>
<td>No authorization for request.</td>
</tr>
<tr>
<td>SSSFINRI (36)</td>
<td>Invalid input to request. See reason code in field SSSFMREA.</td>
</tr>
<tr>
<td>SSSFEXEC (40)</td>
<td>Exit cancelled request.</td>
</tr>
<tr>
<td>SSSFDSA (44)</td>
<td>Scheduler Services disabled.</td>
</tr>
<tr>
<td>SSSFGLBL(48)</td>
<td>Insufficient Global Level</td>
</tr>
</tbody>
</table>

**SSI FMREA Contents**: For the SWB modify function, when control returns to the caller and field SSSFREAS = SSSFINRI (36), the SSSFMREA field contains one of the following decimal values:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSFMOK (0)</td>
<td>Modify processing successful.</td>
</tr>
<tr>
<td>SSSFMTUE (4)</td>
<td>Modify/Erase TU error.</td>
</tr>
<tr>
<td>SSSFMJBE (8)</td>
<td>Modify jobname/jobid error.</td>
</tr>
<tr>
<td>SSSFMGRP (12)</td>
<td>Modify groupname error.</td>
</tr>
<tr>
<td>SSSFMNOS (16)</td>
<td>No storage to process request.</td>
</tr>
<tr>
<td>SSSFMSCI (20)</td>
<td>Invalid security request (SSSFGLG1).</td>
</tr>
<tr>
<td>SSSFMIIVX (24)</td>
<td>Invalid modify extension.</td>
</tr>
<tr>
<td>SSSFMTKE (28)</td>
<td>Modify data set token error.</td>
</tr>
<tr>
<td>SSSFMTCL (32)</td>
<td>No data set token provided.</td>
</tr>
</tbody>
</table>
### SSI Function Code 70

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSFJNF (36)</td>
<td>Job not found.</td>
</tr>
<tr>
<td>SSSFMDNF (40)</td>
<td>Data set not found.</td>
</tr>
<tr>
<td>SSSFMDSB (44)</td>
<td>Data set busy.</td>
</tr>
<tr>
<td>SSSFMDSQ (48)</td>
<td>Data set on BDT or TCP queue.</td>
</tr>
<tr>
<td>SSSFMDSF (52)</td>
<td>Data set referenced by JECL FORMAT statement.</td>
</tr>
<tr>
<td>SSSFMSJF (56)</td>
<td>SJFREQ (MERGE) error.</td>
</tr>
<tr>
<td>SSSFMSCPC (60)</td>
<td>SWBTUREQ (SPLICE) error.</td>
</tr>
<tr>
<td>SSSFMSPT (64)</td>
<td>SWBTUREQ (SPLIT) error.</td>
</tr>
<tr>
<td>SSSFMSTU (68)</td>
<td>SWBTUREQ (RETRIEVE) error.</td>
</tr>
<tr>
<td>SSSFMSPL (72)</td>
<td>Spool I/O error.</td>
</tr>
<tr>
<td>SSSFMTNU (76)</td>
<td>Token not usable for requested function.</td>
</tr>
</tbody>
</table>

When control returns to the caller of the SWB merge function, the SSSFREA field contains one of the following decimal values:

**Note:** The values in SSSFREA only apply to the SWB merge and SWB merge cleanup functions.

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSFDDST (4)</td>
<td>Data set token not given</td>
</tr>
<tr>
<td>SSSFDDSFG (8)</td>
<td>Data set gone</td>
</tr>
<tr>
<td>SSSFDDS夫 (12)</td>
<td>Failure obtaining checkpoint version (JES2 only)</td>
</tr>
<tr>
<td>SSSFJBG (16)</td>
<td>Job gone</td>
</tr>
<tr>
<td>SSSFWSI (20)</td>
<td>Invalid SWBTU buffer</td>
</tr>
<tr>
<td>SSSFDSG (24)</td>
<td>Invalid data set token</td>
</tr>
<tr>
<td>SSSFGE (28)</td>
<td>Invalid group token</td>
</tr>
<tr>
<td>SSSFFNOS (32)</td>
<td>No storage to process request</td>
</tr>
<tr>
<td>SSSFPL (36)</td>
<td>Spool I/O error</td>
</tr>
</tbody>
</table>
SSI Function Code 70

SSSFFTN (40)
Token not usable for requested function

When control returns to the caller of the SWB merge function:

SSSFFSWT
Token to be used by the calling program for SJFREQ services.

SSSFFSWU
Address of the SWBTU buffer.

SSSFFREA
Error reason code for Merge. Reported values are defined above.

SSSFWRTN
Return code from SWB services.

SSSFWOK (0)
success

SSSFWERR (4)
failed, see SSSFWRSN for the exact reason

SSSFWRSN
Reason code for SWB services failure 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=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 = SWB_MERGE
  16 = PUTSWB
  20 = JDTEXTRACT
  24 = SWBTUREQ RETRIEVE
  28 = SWBTUREQ SPLICE
  32 = SWBTUREQ SPLIT

SSSFRLG
Returned flag byte.

SSSFIPA
IP address available in SWBTU buffer.
The JES job information services (SSI function code 71) allows a user-supplied program to obtain information about jobs in the JES queues. Currently only JES2 supports this SSI function code. Most of the information provided via this SSI is very dependent on the version and level of JES2 you are currently running and requires a knowledge of JES2 data structures. Some of the information may be available in a version-independent format using other interfaces (such as SSI 80).

### JES Job Information Services Request Types

<table>
<thead>
<tr>
<th>Request Type</th>
<th>Function (SSJIFREQ)</th>
<th>Request Data Area Pointer (SSJIUSER)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“SPOOL Read Service”</td>
<td>SSJIOM/SSJISIRS</td>
<td>IAZSLPIO</td>
</tr>
<tr>
<td>“JES2 Health Monitor Information” on page 94</td>
<td>SSJIMNOD/SSJIMNRS</td>
<td>IAZMOND</td>
</tr>
<tr>
<td>“Job Class Information” on page 105</td>
<td>SSJIFJCO/SSJIFJCR</td>
<td>IAZJBCLD</td>
</tr>
<tr>
<td>“Convert Device ID Service” on page 116</td>
<td>SSJICVDV</td>
<td>IAZCVDEV</td>
</tr>
<tr>
<td>“Checkpoint Version Information Service” on page 121</td>
<td>SSJIFOBT/SSJIFREL</td>
<td>IAZDSERV</td>
</tr>
</tbody>
</table>

- The JES job information interface is designed to be a general purpose interface to obtain access to JES internal data areas.
- Callers set the field SSJIFREQ in IAZSSJI to the function they want to have performed.
- SSJIUSER points to a data area that contains the data area needed to complete the request.

### SPOOL Read Service

The SPOOL read service provides access to JES2 SPOOL data records. When requesting the SPOOL read service, the SSJIUSER field must point to a parameter area mapped by IAZSPLIO. Any SPOOL record can be read using SPOOL read including JOB or data set control blocks (mapped by $JCT, $IOT, $PDDB, etc.) or SYSIN and SYSOUT data records (mapped by $HDB and $LRC). The SPOOL read service can perform validation of the data read or just read the data at a particular location. The primary input to this function is the SPOOL address of the record to be read (MTTR). The MTTR for the JES2 $JCT data area can be obtained using the extended status SSI (SSI 80 field STJ2SPOL). MTTRs can also be obtained from other JES2 data areas.

Storage for the SPOOL record read is managed by the SPOOL read service. The SPOOL read service is composed of 2 function calls (set in SSJIFREQ). SSJIOM requests that data be read. SSJISIRS releases any storage associated with the request. Multiple reads can be issued without a corresponding release request. Multiple read requests will use the same data buffer to store the data read. If an application needs to access multiple buffers at the same time, it should use multiple IAZSPLIO parameter areas (one per buffer).

Security authorization checking will insure that the user has authority to read the data on spool. For unauthorized (problem state) callers, a security authorization
check will always be performed. For authorized (supervisor state) callers, a security
authorization check will only be performed if requested. The actual security
authorization check performed is dependent on the data (control block) that the
caller is attempting to read. If identifying information about that control block can be
located, an authorization check will be performed in the JESSPOOL class for a
profile with a name of node.userid.jobname.jobid.SPOOLIO.cbname. If this
identifying information cannot be located, then a more restrictive authorization check
will be performed in the JESJOBS class, for a profile with a name of
SPOOLIO.node.jobname.jobid.cbname, where:
• UNKNOWN can be used if JOBNAME is not available
• JOBID always starts with ‘J’ or ‘JOB’ and could be JOB00000
• UNKN can be used for cbname if the control block is not known.

JESJOBS profile checks will be used, for example, when the control block was
created by JES2 that was running at a pre-z/OS V1R9 release level.

See the following sections for more information on SPOOL Read Service:
• “Type of Request”
• “Use Information”
• “Issued to”
• “Related SSI Codes”
• “Related Concepts”
• “Environment”
• “Input Register Information” on page 89
• “Input Parameters” on page 90
• “Output Register Information” on page 92
• “Return Code Information” on page 92
• “Output Parameters” on page 93

Type of Request: Directed SSI Call.

Use Information: To use the JES job information services SSI, a caller must first
decide the function they wish to perform. The appropriate parameter list must be
obtained and pointed to by SSJIUSER.

Issued to: A JES2 subsystem (either primary or secondary). The subsystem does
not have to be associated with the requesting address space.

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:
• IEFSSOBH
• IEFJSSIB
• IAZSSJI
• IAZSPLIO (SPOOL read service)
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, IAZSSJI, and IAZSPLIO, control blocks can reside in 24- or 31-bit virtual storage
- **Recovery**: The caller should provide an ESTAE-type recovery environment. See [z/OS MVS Programming: Assembler Services Guide](https://www.ibm.com/support/docview.wss?uid=swg21385658) for more information on an ESTAE-type recovery environment

Figure 11 shows the environment at the time of the call for SSI function code 71, Spool Read Subfunction.

**Input Register Information**: Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:
### SSI Function Code 71

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

**Input Parameters:** Input parameters for the function routine are:
- SSOB
- SSIB
- IAZSSJI
- IAZSPILO (SPOOL read service)

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 71 (SSOBSSJI)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of the SSIB control block or zero (if this field is zero, the life-of-job SSIB is used). See &quot;Subsystem Identification Block (SSIB)&quot; on page 8 for more information on the life-of-job SSIB</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function-dependent area (IAZSSJI control block)</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem to which this Job Information Services request is directed</td>
</tr>
</tbody>
</table>

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

**IAZSSJI Contents:** The caller must set the following fields in the IAZSSJI control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIID</td>
<td>Eyecatcher for the control block (set to 'SSJI')</td>
</tr>
<tr>
<td>SSJILEN</td>
<td>Length of the IAZSSJI (SSJISIZE) control block</td>
</tr>
<tr>
<td>SSJISVRN</td>
<td>Input version of the IAZSSJI control block. Set to SSJISVR# for version 1 of the control block</td>
</tr>
<tr>
<td>SSJIFREQ</td>
<td>Function to be performed on this request. Valid functions and their related SSJIUSER area are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Value</th>
<th>SSJIUSER Description</th>
</tr>
</thead>
</table>
SSJISIOM  IAZSPLIO SPOOL read service, read record from SPOOL
SSJISIRS  IAZSPLIO SPOOL read service, release storage
SSJIUSER  Pointer to service specific data area '(IAZSPLIO)'

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

SPOOL read service, IAZSPLIO contents: For the SPOOL read service (function code SSJISIOM) the caller must set the following fields in the IAZSPLIO control block:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPIOSSID</td>
<td>Eyecatcher of the control block (set to 'SPIO')</td>
</tr>
<tr>
<td>SPIOLEN</td>
<td>Length of the IAZSPLIO (SPIOSZE) control block</td>
</tr>
<tr>
<td>SPLIOVRN</td>
<td>Input version of the IAZSPLIO control block. Set to SPLIOVR1 for version 1 of the control block. Set to SPLIOVR# for the current (latest) version</td>
</tr>
<tr>
<td>SPIOSPAD</td>
<td>SPOOL address of the record to be read. For JES2 the MTTR is placed in the first four bytes and the next four bytes are set to zero</td>
</tr>
<tr>
<td>SPIOCTYP</td>
<td>Optional input which specifies the type of control block that the caller expects to find at location SPIOSPAD after the spool read operation is complete. Validation of the data area will be performed if this field is specified. The possible values for SPIOCTYP, and the corresponding control block mapping macros used for validation are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CHK</td>
<td>$CHK</td>
<td>Printer check record</td>
</tr>
<tr>
<td></td>
<td>HDB</td>
<td>$BUFFER</td>
<td>SYSIN/SYSOUT data buffer</td>
</tr>
<tr>
<td></td>
<td>IOT</td>
<td>$IOT</td>
<td>Data set information blocks. Contains the PDDBs</td>
</tr>
<tr>
<td></td>
<td>JCT</td>
<td>$JCT</td>
<td>JES2 job control table. Main job control block</td>
</tr>
<tr>
<td></td>
<td>NHSB</td>
<td>$NHSB</td>
<td>NJE headers and trailers</td>
</tr>
<tr>
<td></td>
<td>OCT</td>
<td>$OCT</td>
<td>/* OUTPUT JECL card descriptors</td>
</tr>
<tr>
<td></td>
<td>SIG</td>
<td>none</td>
<td>Signature record (record 0)</td>
</tr>
<tr>
<td></td>
<td>SWBI</td>
<td>$SWBIT</td>
<td>SYSOUT SWB information</td>
</tr>
</tbody>
</table>

| SPIOJNAM  | Optional input that requests that the job name in the data area read matches the value specified. Ignored if SPIOCTYP is not specified or set to 'SIG'. |
| SPIOJID   | Optional input that requests that the JOBID in the data area read matches the value specified. Only |
SSI Function Code 71

the number portion of the JOBID is verified. Ignored if SPIOCTYP is not specified.

**SPIOJKEY**
Optional input that requests that the job key in the data area read matches the value specified. Ignored if SPIOCTYP is not specified.

**SPIODSKY**
Optional input that requests that the data set key in the data area read matches the value specified. Ignored if SPIOCTYP is not specified as 'HDB'.

**SPIOSSNM**
Optional input. When combined with SPIOASID, it requests that SYSOUT data buffers that have not been written to spool be obtained from the specified address space. SPIOSSNM is the name of the system on which the job is currently running and must be specified as either blanks or the name of the local system. For this type of request, SPIOCTYP must be set to 'HDB'.

**SPIOASID**
Optional input. When combined with SPIOSSNM, it requests that SYSOUT data buffers that have not been written to spool be obtained from the address space specified by SPIOASID.

**SPIOOPT**
Processing options flag byte.

**SPIORACF**
Perform authorization checking even if the caller is authorized.

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

For the SPOOL read service function codes SSJISIRS (release storage), the caller should not alter any fields in the IAZSPLIO control block returned on the last SSJISIOM function call.

**Output Register Information:** When control returns to the caller, the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Return Code (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
</table>

SSRTOK (0)  The job information services request completed. Check the SSOBRETN field for specific function information.

SSRTNSUP (4)  The subsystem specified in the SSIBSSNM field does not support the job information services 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
• SSJIRETN
• IAZSPLIO (SPOOL read service)

SSOBRETN Contents:  When control returns to the caller and register 15 contains a zero, the job information services function places one of the following decimal values in the SSOBRETN field:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIOK (0)</td>
<td>Request successful.</td>
</tr>
<tr>
<td>SSJIERRVR (4)</td>
<td>Request completed with possible errors, see SSJIRETN for reason code.</td>
</tr>
<tr>
<td>SSJIERRRU (8)</td>
<td>Request cannot be completed because of user error, see SSJIRETN for reason code.</td>
</tr>
<tr>
<td>SSJIERRRJ (12)</td>
<td>Request cannot be completed, SSJIRETN contains internal reason code.</td>
</tr>
<tr>
<td>SSJIPARM (16)</td>
<td>Error in the parameter list, i.e., the SSJI extension is an invalid format - it is not an SSJI, the service version number is not supported, or the SSJI is not large enough.</td>
</tr>
</tbody>
</table>

SSJIRETN Contents:  In addition to the return code in SSOBRETN, the field SSJIRETN contains the service related error or more specific information about the error. SSJIRETN can be set to the following values if SSOBRETN = SSJIERRRU (8):

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIUNSF (4)</td>
<td>Unsupported subfunction requested.</td>
</tr>
<tr>
<td>SSJIUSER (24)</td>
<td>SSJIUSER does not point to the correct control block.</td>
</tr>
</tbody>
</table>
SSI Function Code 71

**SSJIUNSD (28)**  
Version number in the control block pointed to by SSJIUSER is not correct.

**SSJISMDS (32)**  
Length field in the control block pointed to by SSJIUSER is too small.

*Return codes in SSJIRETN specific to the SPOOL read service:* The following return codes are set if the SPOOL read service was requested and SSOBRETN is zero:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPIOOK (0)</td>
<td>Success</td>
</tr>
<tr>
<td>SPIONTVF (4)</td>
<td>Control block verification failed</td>
</tr>
<tr>
<td>SPOCBIO (8)</td>
<td>SPOOL control block I/O error</td>
</tr>
<tr>
<td>SPOCBTK (12)</td>
<td>SPOOL control block track address</td>
</tr>
<tr>
<td>SPOCBNG (16)</td>
<td>General control block problem</td>
</tr>
<tr>
<td>SPIOSTRG (20)</td>
<td>Error obtaining 31-bit storage</td>
</tr>
<tr>
<td>SPIOJSER (24)</td>
<td>Error obtaining 24-bit storage</td>
</tr>
<tr>
<td>SPIOILOG (28)</td>
<td>A logic error has occurred</td>
</tr>
<tr>
<td>SPIONSPL (32)</td>
<td>SPIOSTRP not initialized correctly</td>
</tr>
<tr>
<td>SPIONBUF (36)</td>
<td>Could not locate instorage buffer. Most likely, the buffer has been written to SPOOL and is no longer in memory.</td>
</tr>
<tr>
<td>SPIONSAL (40)</td>
<td>Authorization (SAF) failure accessing data.</td>
</tr>
</tbody>
</table>

*SPOOL read service, IAZSPLIO contents:* For the SPOOL read service (function code SSJISIOM) the following is returned in IAZSPLIO:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPIOVERO</td>
<td>Subsystem version number (currently 1)</td>
</tr>
<tr>
<td>SPIOOUTA</td>
<td>Address of buffer obtained. This is a pointer directly to the SPOOLed data area (the $SPID). Normally, pointers to the data areas point to a prefix area; this does not.</td>
</tr>
<tr>
<td>SPIOOLEN</td>
<td>Length of the data area returned (not including the prefix area).</td>
</tr>
<tr>
<td>SPIOND1</td>
<td>Indicator field.</td>
</tr>
<tr>
<td>Value (Decimal)</td>
<td>Meaning</td>
</tr>
<tr>
<td>SPIONSTG</td>
<td>The control block was retrieved from an instorage buffer.</td>
</tr>
</tbody>
</table>

**JES2 Health Monitor Information**

The JES2 Health Monitor information service provides diagnostic information about JES2. The information returned is the same data returned by $J commands. For more information about the JES2 Health Monitor see the appendix on Miscellaneous JES2 Facilities in the *z/OS JES2 Initialization and Tuning Guide*.

See the following sections for more information on JES2 Health Monitor Information:
Type of Request: Directed SSI Call.

Use Information: To use the JES job information services SSI, a caller must first decide the function they wish to perform. The appropriate parameter list must be obtained and pointed to by SSJIUSER.

Issued to: A JES2 subsystem (either primary or secondary). The subsystem does not have to be associated with the requesting address space.

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:
- IEFSSOBH
- IEFJSSIB
- IAZSSJI
- IAZMOND (JES2 Health Monitor Information)

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, IAZSSJI, and IAZMOND control blocks can reside in 24- or 31-bit virtual storage

Recovery
- The caller should provide an ESTAE-type recovery environment. See z/OS MVS Programming: Assembler Services Guide for more information on an ESTAE-type recovery environment.
Figure 12 shows the environment at the time of the call for SSI function code 71, JES2 Health Monitor Information Subfunction.

![Diagram of SSOB and SSIB registers](image)

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

**Input Parameters:** Input parameters for the function routine are:
- SSOB
- SSIB
- IAZSSJI
- IAZMOND (JES2 Health Monitor Information)

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
</tbody>
</table>
SSOBLLEN | Length of the SSOB (SSOBHSIZ) control block
SSOBFUNC | SSI function code 71(SSOBSJII)
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 (IAZSSJI 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem to which this Job Information Services request is directed</td>
</tr>
</tbody>
</table>

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

**IAZSSJI Contents:** The caller must set the following fields in the IAZSSJI control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIID</td>
<td>Eyecatcher for the control block (set to 'SSJI')</td>
</tr>
<tr>
<td>SSJILEN</td>
<td>Length of the IAZSSJI (SSJISIZE) control block</td>
</tr>
<tr>
<td>SSJISVRN</td>
<td>Input version of the IAZSSJI control block. Set to SSJISVR# for version 1 of the control block</td>
</tr>
<tr>
<td>SSJIFREQ</td>
<td>Function to be performed on this request. Valid functions and their related SSJIUSER area are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Value</th>
<th>SSJIUSER Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIMNOD</td>
<td>IAZMOND JES2 Health Monitor information, obtain data</td>
</tr>
<tr>
<td>SSJIMNRS</td>
<td>IAZMOND JES2 Health Monitor information, release storage</td>
</tr>
</tbody>
</table>

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

**JES2 Health Monitor Information, IAZMOND contents:** For the JES2 Health Monitor information service (function code SSJIMNOD) the caller must set the following fields in the IAZMOND control block on input to a SSJIMNOD function call:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MONDSSID</td>
<td>Eyecatcher of the control block (set to 'MOND')</td>
</tr>
<tr>
<td>MONDLEN</td>
<td>Length of the IAZMOND (MONDSZE) control block</td>
</tr>
</tbody>
</table>
SSI Function Code 71

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MONDVER</td>
<td>Input version and modifier of the IAZMOND control block. Set to MONDV020 for version 2 of the control block. Set to MONDCVRL and MNDCVRM for the current (latest) version and modifier.</td>
</tr>
<tr>
<td>MONDVERL</td>
<td>Version level</td>
</tr>
<tr>
<td>MONDVERM</td>
<td>Version modifier</td>
</tr>
<tr>
<td>MONDSTRP</td>
<td>Storage management anchor for use by the subsystem that responds to this request. It is expected that the caller will set this field to zero the first time IAZMOND is used and from that point on the field will be managed by the subsystem.</td>
</tr>
<tr>
<td>MONDSEL1</td>
<td>Information selection flag byte 1</td>
</tr>
<tr>
<td>MONDSRES</td>
<td>Resource usage statistics</td>
</tr>
<tr>
<td>MONDSMTS</td>
<td>Main task CPU statistics</td>
</tr>
<tr>
<td>MONDSERR</td>
<td>JES2 ERROR statistics</td>
</tr>
<tr>
<td>MONDSWTS</td>
<td>Main task WAIT statistics</td>
</tr>
<tr>
<td>MONDSJSA</td>
<td>JES2 Alerts®</td>
</tr>
<tr>
<td>MONSJSN</td>
<td>JES2 Notices</td>
</tr>
<tr>
<td>MONDSJST</td>
<td>JES2 Tracks</td>
</tr>
<tr>
<td>MONDSSTO</td>
<td>JES2 storage usage statistics</td>
</tr>
<tr>
<td>MONDSEL2</td>
<td>Information selection flag byte 2</td>
</tr>
<tr>
<td>MONDSMNS</td>
<td>Monitor status information</td>
</tr>
<tr>
<td>MONDHSTC</td>
<td>History count limit.</td>
</tr>
<tr>
<td>MONDRSNM</td>
<td>Resource name filter.</td>
</tr>
</tbody>
</table>

The JES2 Health Monitor maintains a history for some statistics (resource usage, CPU statistics, and error statistics). In general, these statistics are reset at the beginning of each hour and hourly statistics are maintained for as long as the JES2 Health Monitor is running. The amount of history returned can be limited by setting MONDHSTC to the number of history elements to return. Setting MONDHSTC to 0 or 1 will return only the current data. Setting it to 5 will return the 5 most recent history elements. Setting it to X'FFFF' will return all history elements.

MONDHSTC only applies when setting MONDSRES, MONDSMTS, or MONDSERR in MONDSEL1.

MONDRSNM

Resource name filter.

If MONDSEL1 is set to MONDSRES, then MONDRSNM can be set to the resource name for which information is to be returned (left justified and padded with blanks). Generics ("*" and ?) are allowed. Setting the first byte of MONDRSNM to zero or blanks is the same as setting MONDRSNM to "*". All resources are returned.
Set all other fields in the IAZMOND control block to binary zeros before issuing the IEFSSREQ macro.

For the JES2 Health Monitor Information service function codes SSJIMNRS (release storage), the caller should set MONDSTRP in the IAZMOND control block to indicate the storage to be released.

**Output Register Information:** When control returns to the caller, the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Return Code (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>The job information services request completed. Check the SSOBRETN field for specific function information.</td>
</tr>
<tr>
<td>SSRTNSUP (4)</td>
<td>The subsystem specified in the SSIBSSNM field does not support the job information services function call.</td>
</tr>
<tr>
<td>SSRTNTUP (8)</td>
<td>The subsystem specified in the SSIBSSNM field exists but is not active.</td>
</tr>
<tr>
<td>SSRTNOSS (12)</td>
<td>The subsystem specified in the SSIBSSNM field is not defined to MVS.</td>
</tr>
<tr>
<td>SSRTDIST (16)</td>
<td>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.</td>
</tr>
<tr>
<td>SSRTLERR (20)</td>
<td>Either the SSIB control block or the SSOB control block has incorrect lengths or formats.</td>
</tr>
<tr>
<td>SSRTNSSI(24)</td>
<td>The SSI has not been initialized.</td>
</tr>
</tbody>
</table>

**Output Parameters:** Output parameters for the function routine are:
- SSOBRETN
- SSJIRETN
- IAZMOND (JES2 Health Monitor information)

**SSOBRETN Contents:** When control returns to the caller and register 15 contains a zero, the job information services function places one of the following decimal values in the SSOBRETN field:
### SSI Function Code 71

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIOK (0)</td>
<td>Request successful.</td>
</tr>
<tr>
<td>SSJIERVR (4)</td>
<td>Request completed with possible errors, see SSJIRETN for reason code.</td>
</tr>
<tr>
<td>SSJIERRU (8)</td>
<td>Request cannot be completed because of user error, see SSJIRETN for reason code.</td>
</tr>
<tr>
<td>SSJIERRJ (12)</td>
<td>Request cannot be completed, SSJIRETN contains internal reason code.</td>
</tr>
<tr>
<td>SSJIPARM (16)</td>
<td>The parameter list, i.e., the SSJI extension is an invalid format - it is not an SSJI, the service version number is not supported, or the SSJI is not large enough.</td>
</tr>
</tbody>
</table>

**SSJIRETN Contents:** In addition to the return code in SSOBRETN, the field SSJIRETN contains the service related error or more specific information about the error. SSJIRETN will be set to one of the following decimal values:

When *SSOBRETN* is *SSJIERVR (4)*:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MONDNMON (4)</td>
<td>JES2 Health Monitor address space is down</td>
</tr>
</tbody>
</table>

When *SSOBRETN* is *SSJIERRU (8)*:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIUNSF (4)</td>
<td>Unsupported subfunction requested</td>
</tr>
<tr>
<td>MONDIERR (12)</td>
<td>Input error (no information selection flags set)</td>
</tr>
<tr>
<td>MONDSTRE (16)</td>
<td>MONDSTRP not set correctly</td>
</tr>
<tr>
<td>SSJINTDS (24)</td>
<td>SSJIUSER does not point to the correct data area</td>
</tr>
<tr>
<td>SSJIUNSD (28)</td>
<td>SSJIUSER CB version number is not correct</td>
</tr>
<tr>
<td>SSJISMDS (32)</td>
<td>SSJIUSER CB length is too small</td>
</tr>
</tbody>
</table>

*Return codes in SSJIRETN specific to the JES2 Health Monitor Information service:* The following return codes are set if the JES2 Health Monitor Information service was request and SSOBRETN is zero:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MONDOK (0)</td>
<td>Success</td>
</tr>
</tbody>
</table>

### JES2 Health Monitor Information service, IAZMOND contents
For the JES2 Health Monitor Information service (function code SSJIMNOD), two types of data are returned in the IAZMOND. The fixed data section and the section which contains elements for each type of usage statistic that matched the filters specified:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MONDVERO</td>
<td>Subsystem version/modifier number.</td>
</tr>
<tr>
<td>MONDPERF</td>
<td>Performance index for last performed request.</td>
</tr>
<tr>
<td>MONDSTAT</td>
<td>Status indicator for JES2 and the JES2 Health Monitor.</td>
</tr>
</tbody>
</table>
MONDJDWN
  JES2 is down

MONDMONI
  Pointer to resource usage statistics (MDRSDATA)

MONDCPUS
  Pointer to main task CPU statistics (MDCPDATA)

MONDERRC
  Pointer to JES2 ERROR statistics (MDERDATA)

MONDMSGS
  Pointer to JES2 Alert/Track/Notice messages (MDMSDATA)

MONDMONI
  Pointer to JES2 Health Monitor status information (MDMIDATA)

MONDSTRU
  Pointer to JES2 storage usage statistics (MDSTDATA)

MONDSZE1
  Version 1 size

MONDSZE2
  Version 2 size

Resource Usage Statistics Elements: For each resource, an information element is added to the chain pointed to by MONDRESQ. Each element contains a fixed size prefix (mapped by the MDRSDATA DSECT) and one or more fixed size time entries (mapped by the MDRSNTRY DSECT).

The fields in the MDRSDATA prefix are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDRSEYE</td>
<td>Eyecatcher 'MDRS'</td>
</tr>
<tr>
<td>MDRSNEXT</td>
<td>Next MDRS entry</td>
</tr>
<tr>
<td>MDRSNAME</td>
<td>Resource name</td>
</tr>
<tr>
<td>MDRSENTO</td>
<td>Offset to first time entry</td>
</tr>
<tr>
<td>MDRSCNT</td>
<td>Number of time entries</td>
</tr>
<tr>
<td>MDRSENTL</td>
<td>Length of a time entry</td>
</tr>
<tr>
<td>MDRSFLG1</td>
<td>General flag byte</td>
</tr>
<tr>
<td>MDRS1OVR</td>
<td>Resource currently over warning level</td>
</tr>
</tbody>
</table>

The fields in the MDRSNTRY time entry are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDRSTIME</td>
<td>Time interval started (STCKE)</td>
</tr>
<tr>
<td>MDRSLIMT</td>
<td>Current upper limit</td>
</tr>
<tr>
<td>MDRSINUS</td>
<td>Current number in use</td>
</tr>
<tr>
<td>MDRSLOW</td>
<td>Low usage value</td>
</tr>
<tr>
<td>MDRSHIGH</td>
<td>High usage value</td>
</tr>
<tr>
<td>MDRSAVRG</td>
<td>Average in use value</td>
</tr>
</tbody>
</table>
**SSI Function Code 71**

**MDRSWARN**  Warn level (%)
**MDROVER**  Usage over warn level (% * 100)

**Main Task CPU Statistics Elements:** Each element contains a fixed size prefix (mapped by MDCPDATA DSECT) and one or more fixed size time entries (mapped by MDCPNTRY DSECT).

The fields in the MDCPDATA prefix are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDCPEYE</td>
<td>Eyecatcher ‘MDCP’</td>
</tr>
<tr>
<td>MDCPENTO</td>
<td>Offset to first time entry</td>
</tr>
<tr>
<td>MDCPCNT</td>
<td>Number of time entries</td>
</tr>
<tr>
<td>MDCPENTL</td>
<td>Length of a time entry</td>
</tr>
</tbody>
</table>

The fields in the MDCPNTRY time entry are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDCPTIME</td>
<td>Time interval started (STCKE)</td>
</tr>
<tr>
<td>MDCPACT</td>
<td>Active sample count</td>
</tr>
<tr>
<td>MDCPIDLE</td>
<td>Idle sample count</td>
</tr>
<tr>
<td>MDCPWAIT</td>
<td>Wait sample count</td>
</tr>
<tr>
<td>MDCPLLOK</td>
<td>Local lock sample count</td>
</tr>
<tr>
<td>MDCPNDSP</td>
<td>Non Dispatchable count</td>
</tr>
<tr>
<td>MDCPPAGE</td>
<td>Page wait sample count</td>
</tr>
<tr>
<td>MDCPDMVS</td>
<td>Awaiting MVS dispatch count</td>
</tr>
</tbody>
</table>

**JES2 ERROR Statistics Elements:** For each error, an information element is added to the chain pointed to by MONDERRC. Each element contains a fixed size prefix (mapped by the MDERDATA DSECT) and one or more fixed size time entries (mapped by the MDERNTRY DSECT).

The fields in the MDERDATA prefix are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIDEREYE</td>
<td>Eyecatcher ‘MDER’</td>
</tr>
<tr>
<td>MDERNEXT</td>
<td>Next MDER entry</td>
</tr>
<tr>
<td>MDERNAME</td>
<td>Error name</td>
</tr>
<tr>
<td>MDERENTO</td>
<td>Offset to first time entry</td>
</tr>
<tr>
<td>MDERCNT</td>
<td>Number of time entries</td>
</tr>
<tr>
<td>MDERENTL</td>
<td>Length of a time entry</td>
</tr>
</tbody>
</table>

The fields in the MDERNTRY time entry are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDERTIME</td>
<td>Time interval started (STCKE)</td>
</tr>
<tr>
<td>MDERCOUN</td>
<td>Error count</td>
</tr>
</tbody>
</table>
Main Task WAIT Statistics Elements: Each element contains a fixed size prefix (mapped by MDWTDATA DSECT) and one or more fixed size time entries (mapped by MDWTNTRY DSECT).

The fields in the MDWTDATA prefix are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDWTEYE</td>
<td>Eyecatcher 'MDWT'</td>
</tr>
<tr>
<td>MDWTENTO</td>
<td>Offset to first time entry</td>
</tr>
<tr>
<td>MDWTCNT</td>
<td>Number of wait entries</td>
</tr>
<tr>
<td>MDWTENTL</td>
<td>Length of a wait entry</td>
</tr>
</tbody>
</table>

The fields in the MDWTNTRY time entry are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDWTSTCK</td>
<td>Time of most recent wait (STCKE)</td>
</tr>
<tr>
<td>MDWTADDR</td>
<td>Address of wait (from RB)</td>
</tr>
<tr>
<td>MDWTWCNT</td>
<td>Count of waits detected</td>
</tr>
<tr>
<td>MDWTSCNT</td>
<td>Count of matching samples</td>
</tr>
<tr>
<td>MDWTNAME</td>
<td>Module name from wait</td>
</tr>
<tr>
<td>MDWTOFFS</td>
<td>Offset of wait in module</td>
</tr>
<tr>
<td>MDWTPCE</td>
<td>Name of PCE in control (or MULTIPLE)</td>
</tr>
<tr>
<td>MDWTEXIT</td>
<td>Exit number in control</td>
</tr>
<tr>
<td>MDWTFINI</td>
<td>JES2 was initializing</td>
</tr>
<tr>
<td>MDWTFRM</td>
<td>JES2 was terminating</td>
</tr>
</tbody>
</table>

JES2 Alert/Track/Notice Messages Elements: Each element contains one or more fixed size entries (mapped by MDMSDATA).
The fields in the MDMSDATA entry are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDMSEYE</td>
<td>Eyecatcher 'MDMS'</td>
</tr>
<tr>
<td>MDMSNEXT</td>
<td>Next MDMS entry</td>
</tr>
<tr>
<td>MDMSLEN</td>
<td>Length of entry</td>
</tr>
<tr>
<td>MDMSTIME</td>
<td>Time condition started (STCKE). Only for Alerts and Tracks.</td>
</tr>
<tr>
<td>MDMSTYPE</td>
<td>Message type</td>
</tr>
<tr>
<td>MDMSTALR</td>
<td>Alert message</td>
</tr>
<tr>
<td>MDMSTTRK</td>
<td>Track message</td>
</tr>
<tr>
<td>MDMSTNOT</td>
<td>Notice message</td>
</tr>
<tr>
<td>MDMDL1LN</td>
<td>Length of first line of message</td>
</tr>
<tr>
<td>MDMDL1TX</td>
<td>Text of first line of message</td>
</tr>
<tr>
<td>MDMDL2LN</td>
<td>Length of second line of message</td>
</tr>
<tr>
<td>MDMDL2TX</td>
<td>Text of second line of message</td>
</tr>
<tr>
<td>MDMDL3LN</td>
<td>Length of third line of message</td>
</tr>
<tr>
<td>MDMDL3TX</td>
<td>Text of third line of message</td>
</tr>
<tr>
<td>MDMDL4LN</td>
<td>Length of fourth line of message</td>
</tr>
<tr>
<td>MDMDL4TX</td>
<td>Text of fourth line of message</td>
</tr>
</tbody>
</table>

JES2 Health Monitor status information elements: Each element contains a fixed size prefix (mapped by MDMIDATA DSECT) and one or more fixed size subtask entries (mapped by MDMINTRY DSECT).

The fields in the MDMIDATA entry are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDMIEYE</td>
<td>Eyecatcher 'MDM'</td>
</tr>
<tr>
<td>MDMIENTO</td>
<td>Offset to first subtask entry</td>
</tr>
<tr>
<td>MDMICNT</td>
<td>Number of subtask entries</td>
</tr>
<tr>
<td>MDMIENTL</td>
<td>Length of a subtask entry</td>
</tr>
</tbody>
</table>

The fields in the MDMINTRY subtask entry are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDMINAME</td>
<td>Name of monitor task</td>
</tr>
<tr>
<td>MDMISTAT</td>
<td>Current task status</td>
</tr>
<tr>
<td>MDMINFO</td>
<td>Status information for subtask</td>
</tr>
</tbody>
</table>

JES2 Storage Usage Statistics Elements: Each element contains a fixed size prefix (mapped by MDSTDATA DSECT) and one or more fixed size time entries (mapped by MDSTNTRY DSECT).
The fields in the MDSTDATA entry are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDSTEYE</td>
<td>Eyecatcher 'MDST'</td>
</tr>
<tr>
<td>MDSTNEXT</td>
<td>Next MDST entry</td>
</tr>
<tr>
<td>MDSTNAME</td>
<td>Storage area description</td>
</tr>
<tr>
<td>MDSTENTO</td>
<td>Offset to first time entry</td>
</tr>
<tr>
<td>MDSTCNT</td>
<td>Number of time entries</td>
</tr>
<tr>
<td>MDSTENTL</td>
<td>Length of a time entry</td>
</tr>
</tbody>
</table>

The fields in the MDSTNTRY time entry are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDSTTIME</td>
<td>Interval start time (STCKE)</td>
</tr>
<tr>
<td>MDSTREGN</td>
<td>Current region size (bytes)</td>
</tr>
<tr>
<td>MDSTUSE</td>
<td>Current bytes in use</td>
</tr>
<tr>
<td>MDSTLOW</td>
<td>Low usage value (bytes)</td>
</tr>
<tr>
<td>MDSTHIGH</td>
<td>High usage value (bytes)</td>
</tr>
<tr>
<td>MDSTAVRG</td>
<td>Average usage value (bytes)</td>
</tr>
</tbody>
</table>

**Job Class Information**

The job class information service provides the attributes of a job class.

See the following sections for more information on Job Class Information:
- "Type of Request"
- "Use Information"
- "Issued to"
- "Related SSI Codes"
- "Related Concepts"
- "Environment" on page 106
- "Input Register Information" on page 107
- "Input Parameters" on page 107
- "Output Register Information" on page 109
- "Return Code Information" on page 109
- "Output Parameters" on page 110

**Type of Request:** Directed SSI Call.

**Use Information:** To use the JES job information services SSI, a caller must first decide the function they wish to perform. The appropriate parameter list must be obtained and pointed to by SSJIUUSER.

**Issued to:** A JES2 subsystem (either primary or secondary). The subsystem does not have to be associated with the requesting address space.

**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:
- IEFSSOBH
- IEFJSSIB
- IAZSSJI
- IAZJBCLD (Job Class Information)

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, IAZSSJI, and IAZJBCLD control blocks can reside in 24- or 31-bit virtual storage |
| Recovery | The caller should provide an ESTAE-type recovery environment. See [z/OS MVS Programming: Assembler Services Guide](https://pubs.opengroup.org/onlinepubs/009695399/) for more information on an ESTAE-type recovery environment |

Figure 13 on page 107 shows the environment at the time of the call for SSI function code 71, Job Class Information Subfunction.
**Input Register Information:** Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

**Input Parameters:** Input parameters for the function routine are:
- SSOB
- SSIB
- IAZSSJI
- IAZJBCLD (Job Class Information)

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 71(SSOBSSJI)</td>
</tr>
</tbody>
</table>
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 (IAZSSJI 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier ‘SSIB’</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem to which this Job Information Services request is directed</td>
</tr>
</tbody>
</table>

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

**IAZSSJI Contents:** The caller must set the following fields in the IAZSSJI control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIID</td>
<td>Eyecatcher for the control block (set to ‘SSJI’)</td>
</tr>
<tr>
<td>SSJILEN</td>
<td>Length of the IAZSSJI (SSJISIZE) control block</td>
</tr>
<tr>
<td>SSJISVRN</td>
<td>Input version of the IAZSSJI control block. Set to SSJISVR# for version 1 of the control block</td>
</tr>
<tr>
<td>SSJIFREQ</td>
<td>Function to be performed on this request. Valid functions and their related SSJIUSER area are:</td>
</tr>
<tr>
<td></td>
<td>Field Value</td>
</tr>
<tr>
<td></td>
<td>SSJIFJCO</td>
</tr>
<tr>
<td></td>
<td>SSJIFJCR</td>
</tr>
<tr>
<td>SSJIUSER</td>
<td>Pointer to service specific data area (IAZJBCLD)</td>
</tr>
</tbody>
</table>

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

**Job Class Information service, IAZJBCLD contents:** For the job class information service (function code SSJIFJCO) the caller must set the following fields in the IAZJBCLD control block:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBCLSSID</td>
<td>Eyecatcher of the control block (set to ‘JBCL’)</td>
</tr>
<tr>
<td>JBCLLEN</td>
<td>Length of the IAZJBCLD (JBCLSZE) control block</td>
</tr>
<tr>
<td>JBCLSVRN</td>
<td>Input version of the IAZJBCLD control block. Set to JBCLSVR# for the current (latest) version.</td>
</tr>
</tbody>
</table>
JBCLSTRP  Storage management anchor for use by the subsystem that responds to this request. It is expected that the caller will set this field to zero the first time IAZJBCLD is used and from that point on the field will be managed by the subsystem.

JBCLFLAG  Flag byte

JBCL1JOB  Return a particular job class indicated by JBCLNAM

JBCLJNAM  Single job class to be returned.

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

For the Job Class information service function codes SSJIFJCR (return storage), the caller should set JBCLSTRP in the IAZJBCLD control block to indicate the storage to be released.

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

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 job information services request completed. Check the SSOBRETN field for specific function information.

SSRTNSUP (4)  The subsystem specified in the SSIBSSNM field does not support the job information services 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.
SSI Function Code 71

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
- SSJIRETN
- IAZJBCLD (Job Class Information service)

**SSOBRETN Contents:** When control returns to the caller and register 15 contains a zero, the job information services function places one of the following decimal values in the SSOBRETN field:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIOK (0)</td>
<td>Request successful.</td>
</tr>
<tr>
<td>SSJIERRVR (4)</td>
<td>Request completed with possible errors, see SSJIRETN for reason code.</td>
</tr>
<tr>
<td>SSJIERRU (8)</td>
<td>Request cannot be completed because of user error, see SSJIRETN for reason code.</td>
</tr>
<tr>
<td>SSJIERRJ (12)</td>
<td>Request cannot be completed, SSJIRETN contains internal reason code.</td>
</tr>
<tr>
<td>SSJIPARM (16)</td>
<td>The parameter list, ie, the SSJI extension is an invalid format - it is not an SSJI, the service version number is not supported, or the SSJI is not large enough.</td>
</tr>
</tbody>
</table>

**SSJIRETN Contents:** In addition to the return code in SSOBRETN, the field SSJIRETN contains the service related error or more specific information about the error. SSJIRETN will be set to one of the following decimal values when SSOBRETN is not zero:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIUNSF (4)</td>
<td>Unsupported subfunction requested</td>
</tr>
<tr>
<td>SSJINTDS (24)</td>
<td>SSJIUSER does not point to the correct data area</td>
</tr>
<tr>
<td>SSJIUNSD (28)</td>
<td>SSJIUSER control block version number is not correct</td>
</tr>
<tr>
<td>SSJISMD (32)</td>
<td>SSJIUSER control block length is too small</td>
</tr>
</tbody>
</table>

**Return codes in SSJIRETN specific to the Job Class Information service:** The following return codes are set if the Job Class Information service was requested and SSOBRETN is zero:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIOK (0)</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Job Class Information service, IAZJBCLD contents:** For the Job Class Information service (function code SSJIFJCO), two types of data, fixed data in the IAZJBCLD and elements for each job class that match the filters specified. The following describes the fixed data fields returned in the IAZJBCLD:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBCLVERO</td>
<td>Subsystem version number (currently 4)</td>
</tr>
</tbody>
</table>
JBCLSMCL STC message class
JBCLTMCL TSU message class
JBCLDPTR Pointer to first job class data buffer
JBCLNJC Number of job classes returned

Job Class Information Elements: Each element contains a header and two sections, class attribute table and member specific attributes. The class attribute table and member specific sections contain a prefix and data section.

Job Class Information Element header (mapped by JBCLDHDR DSECT):

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBTCTEYE</td>
<td>Eyecatcher ‘DCAT’</td>
</tr>
<tr>
<td>JBCTOHDR</td>
<td>Offset to first section</td>
</tr>
<tr>
<td>JBCTNEXT</td>
<td>Address of next CAT</td>
</tr>
<tr>
<td>JBCLDHSZ</td>
<td>Size of header</td>
</tr>
</tbody>
</table>

Class Attribute Table Prefix section (mapped by JBCTPREF DSECT):

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBCTHDLN</td>
<td>Length of entire job class entry (maximum value is 65535)</td>
</tr>
<tr>
<td>JBCTHDTGP</td>
<td>Type of this section</td>
</tr>
<tr>
<td>JBCTHDMD</td>
<td>Modifier</td>
</tr>
<tr>
<td>JBCTHDSZ</td>
<td>Size of prefix section</td>
</tr>
</tbody>
</table>

Class Attribute Table Data section (mapped by JBCLDCAT DSECT):

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBCTLEN</td>
<td>Length of job class data section</td>
</tr>
<tr>
<td>JBCTTYPE</td>
<td>Type of this section (JBCLTCAT)</td>
</tr>
<tr>
<td>JBCTMOD</td>
<td>Modifier</td>
</tr>
<tr>
<td>JBCJOBFL</td>
<td>Job flags</td>
</tr>
<tr>
<td>JBCBATCH</td>
<td>Batch job</td>
</tr>
<tr>
<td>JBCTSUJB</td>
<td>Time sharing user</td>
</tr>
<tr>
<td>JBCTCJB</td>
<td>System task</td>
</tr>
<tr>
<td>JBCVALJB</td>
<td>Valid job types</td>
</tr>
<tr>
<td>JBCNOJNL</td>
<td>No journal option</td>
</tr>
<tr>
<td>JBCNOUPT</td>
<td>No output option</td>
</tr>
</tbody>
</table>

Chapter 3. SSI Function Codes Your Program Can Request 111
SSI Function Code 71

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBCTSCAN</td>
<td>TYPRUN=SCAN was specified</td>
</tr>
<tr>
<td>JBCTCOPY</td>
<td>TYPRUN=COPY was specified</td>
</tr>
<tr>
<td>JBCRSTRT</td>
<td>Allow warm start to re-que for execution</td>
</tr>
<tr>
<td>JBCJBOPT</td>
<td>Job options flag</td>
</tr>
<tr>
<td>JBCTHOLD</td>
<td>TYPRUN=HOLD</td>
</tr>
<tr>
<td>JBCNOLOG</td>
<td>No job log option</td>
</tr>
<tr>
<td>JBCXBMII</td>
<td>XBM II job</td>
</tr>
<tr>
<td>JBCQHELD</td>
<td>Class queue is held</td>
</tr>
<tr>
<td>JBCPROCN</td>
<td>Procedure library number</td>
</tr>
<tr>
<td>JBCSMFLG</td>
<td>SMF Flag</td>
</tr>
<tr>
<td>JBCNOUSO</td>
<td>Do not take IEFUSO exit</td>
</tr>
<tr>
<td>JBCNOTY6</td>
<td>Do not produce Type 6 SMF record</td>
</tr>
<tr>
<td>JBCNOUJP</td>
<td>Do not take IEFUJP exit</td>
</tr>
<tr>
<td>JBCNOT26</td>
<td>Do not produce Type 26 SMF record</td>
</tr>
<tr>
<td>JBCPERFM</td>
<td>Default performance group</td>
</tr>
<tr>
<td>JBCCPBGN</td>
<td>Beginning of converter parameters</td>
</tr>
<tr>
<td>JBCCACCT</td>
<td>Accounting information flag</td>
</tr>
<tr>
<td>JBCCNONE</td>
<td>No information is required</td>
</tr>
<tr>
<td>JBCNAME</td>
<td>Programmer required</td>
</tr>
<tr>
<td>JBCCNUMB</td>
<td>Account number required</td>
</tr>
<tr>
<td>JBCCALL</td>
<td>Programmer and account number required</td>
</tr>
<tr>
<td>JBCCSWAL</td>
<td>SWA above 16M line</td>
</tr>
<tr>
<td>JBCCTIME</td>
<td>Default job step interval time</td>
</tr>
<tr>
<td>JBCCMNTE</td>
<td>Maximum minutes</td>
</tr>
<tr>
<td>JBCSECS</td>
<td>Maximum seconds</td>
</tr>
</tbody>
</table>
JBCCREGN  Default job step region
JBCCRGN   Numeric specification
JBCCRGGA  Kilobytes or megabytes specification
JBCCMND   Command disposition
JBCCEXEC  Pass the command through
JBCCDSPL  Display and then pass command
JBCCVER   Ask operator disposition
JBCCIGN   Ignore the command
JBCCBLP   Bypass label processing option
JBCCBLPY  Process bypass label parm
JBCCOCG   Operator command group
JBCCGSYS  Group 1 commands (SYS)
JBCCGIO   Group 2 commands (I/O)
JBCCGCON  Group 3 commands (CONS)
JBCCGALL  All command groups
JBCCCLJCL  Default MSGLEVEL, JCL listed if not MSGLEVEL
JBCCCTMSG  Allocation termination messages
JBCCONVP   End of converter parameters
JBCCOPSWT  Converter option switches
JBCCFLAG1  Normal OUTDISP for JESDS
JBCC1CDP  Conditionally purge output for jobs in this class
JBCC1NODP  NORMAL OUTDISP=PURGE
JBCC1NODW  NORMAL OUTDISP=WRITE
JBCC1NODH  NORMAL OUTDISP=HOLD
JBCC1NODK  NORMAL OUTDISP=KEEP
SSI Function Code 71

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBC1NODL</td>
<td>NORMAL OUTDISP=LEAVE</td>
</tr>
<tr>
<td>JBC1AODP</td>
<td>ABNORMAL OUTDISP=PURGE</td>
</tr>
<tr>
<td>JBC1AODW</td>
<td>ABNORMAL OUTDISP=WRITE</td>
</tr>
<tr>
<td>JBC1AODH</td>
<td>ABNORMAL OUTDISP=HOLD</td>
</tr>
<tr>
<td>JBC1AODK</td>
<td>ABNORMAL OUTDISP=KEEP</td>
</tr>
<tr>
<td>JBC1AODL</td>
<td>ABNORMAL OUTDISP=LEAVE</td>
</tr>
<tr>
<td>JBCFLAG2</td>
<td>Abnormal OUTDISP for JESDS</td>
</tr>
<tr>
<td>JBC3WLM</td>
<td>WLM managed class</td>
</tr>
<tr>
<td>JBC3SPEC</td>
<td>Special class (STC/TSU)</td>
</tr>
<tr>
<td>JBC3PSEU</td>
<td>Pseudo class (Only class name and counts valid)</td>
</tr>
<tr>
<td>JBC3SINV</td>
<td>Default SCHENV (CATSCHED) no longer defined</td>
</tr>
<tr>
<td>JBC3DUOK</td>
<td>Duplicate job names OK for this job class</td>
</tr>
<tr>
<td>JBCXBM</td>
<td>PROCNAME for XBM II job</td>
</tr>
<tr>
<td>JBCCLASS</td>
<td>Job class</td>
</tr>
<tr>
<td>JBCMAXJ</td>
<td>Maximum executing jobs in this class in the JESplex</td>
</tr>
<tr>
<td>JBCCURJ</td>
<td>Current executing job in this class in the JESplex</td>
</tr>
<tr>
<td>JBCQSIZE</td>
<td>Jobs eligible for execution (including executing jobs)</td>
</tr>
<tr>
<td>JBCCHLDCT</td>
<td>Jobs held in class (not including executing jobs)</td>
</tr>
<tr>
<td>JBCTSIZ1</td>
<td>Version 1 job class length</td>
</tr>
<tr>
<td>JBCDSCHE</td>
<td>Default SCHENV, Job classes only</td>
</tr>
<tr>
<td>JBCDMCLS</td>
<td>Default MSGCLASS, TSU and STC classes only</td>
</tr>
<tr>
<td>JBCSIZ2</td>
<td>Version 2 job class length</td>
</tr>
<tr>
<td>JBCJLOG</td>
<td>JESLOG control information</td>
</tr>
<tr>
<td>JBCFLFLG</td>
<td>Flags</td>
</tr>
<tr>
<td>JBJLELIG</td>
<td>Spin eligible</td>
</tr>
<tr>
<td>JBJLTIMI</td>
<td>Spin on time interval</td>
</tr>
</tbody>
</table>
JetStream Internal Module Definitions

- **JBJLTIMD**: Spin on time of day
- **JBJLLINE**: Spin upon line delta
- **JBJLSUP**: Suppress
- **JBJLNOSP**: No spin
- **JBJSOURC**: Source of JESLOG info
- **JBJSEXIT**: JESLOG from Exit
- **JBJSJCL**: JESLOG from JCL
- **JBJSJCL**: JESLOG from CAT
- **JBJSJCL**: JESLOG from IEFSSRR
- **JBCJLVAL**: Spin value
- **JBCTSZ3**: Version 3 job class length
- **JBCTSZ4**: Version 4 job class length
- **JBCTSIZE**: Job class data length

Member specific attribute prefix section (mapped by JBCLMEMD DSECT):

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBCMLEN</td>
<td>Length of member specific section (prefix + data)</td>
</tr>
<tr>
<td>JBCMCTYPE</td>
<td>Type of this section (JBCLTFMEM)</td>
</tr>
<tr>
<td>JBCMMOD</td>
<td>Modifier</td>
</tr>
<tr>
<td>JBCMFIRST</td>
<td>First member section offset</td>
</tr>
<tr>
<td>JBCMCONT</td>
<td>Count of member entries</td>
</tr>
<tr>
<td>JBCMMLEN</td>
<td>Length of a member entry</td>
</tr>
<tr>
<td>JBCM1ST</td>
<td>Beginning of first member entry</td>
</tr>
<tr>
<td>JBCMSIZE</td>
<td>Size of fixed length prefix</td>
</tr>
</tbody>
</table>

Member specific attribute data section (mapped by JBCLMEME DSECT):

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBCMNAME</td>
<td>JES2 member name</td>
</tr>
<tr>
<td>JBCMSYS</td>
<td>MVS system name</td>
</tr>
<tr>
<td>JBCMFLG1</td>
<td>Member flags</td>
</tr>
<tr>
<td>JBCM1JBA</td>
<td>Jobclass active on member</td>
</tr>
<tr>
<td><strong>JBC1ACT</strong></td>
<td>Member is active</td>
</tr>
</tbody>
</table>
SSI Function Code 71

Convert Device ID Service
The Convert Device ID service translates a binary device ID into its EBCDIC device name.

See the following sections for more information on Convert Device ID Service:
- "Type of Request"
- "Use Information"
- "Issued to"
- "Related SSI Codes"
- "Related Concepts"
- "Environment"
- "Input Register Information" on page 117
- "Input Parameters" on page 118
- "Output Register Information" on page 119
- "Return Code Information" on page 119
- "Output Parameters" on page 120

Type of Request: Directed SSI Call.

Use Information: To use the JES job information services SSI, a caller must first decide the function they wish to perform. The appropriate parameter list must be obtained and pointed to by SSJIUSER.

Issued to: A JES2 subsystem (either primary or secondary). The subsystem does not have to be associated with the requesting address space.

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:
- IEFSSOBH
- IEFJSSIB
- IAZSSJI
- IAZCVDEV (Convert Device ID service)

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, IAZSSJI, and IAZCVDEV control blocks can reside in 24- or 31-bit virtual storage
Recovery: The caller should provide an ESTAE-type recovery environment. See [z/OS MVS Programming: Assembler Services Guide] for more information on an ESTAE-type recovery environment.

Figure 14 shows the environment at the time of the call for SSI function code 71, Convert Device ID Service Subfunction.

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 14. Environment at Time of Call for SSI Function Code 71, Convert Device ID Service Subfunction
### SSI Function Code 71

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
- IAZSSJI
- IAZCVDEV (Convert Device ID service)

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 71(SSOBSSJI)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of the SSIB control block or zero (if this field is zero, the life-of-job SSIB is used). See <a href="#">“Subsystem Identification Block (SSIB)” on page 8</a> for more information on the life-of-job SSIB</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function-dependent area (IAZSSJI control block)</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier ‘SSIB’</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem to which this Job Information Services request is directed</td>
</tr>
</tbody>
</table>

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

**IAZSSJI Contents:** The caller must set the following fields in the IAZSSJI control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIID</td>
<td>Eyecatcher for the control block (set to ‘SSJI’)</td>
</tr>
<tr>
<td>SSJILEN</td>
<td>Length of the IAZSSJI (SSJISIZE) control block</td>
</tr>
<tr>
<td>SSJISVRN</td>
<td>Input version of the IAZSSJI control block. Set to SSJISVR# for version 1 of the control block</td>
</tr>
<tr>
<td>SSJIFREQ</td>
<td>Function to be performed on this request. Valid functions and their related SSJIUSER area are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJICVDV</td>
<td>IAZCVDEV, Convert device ID service</td>
</tr>
<tr>
<td>SSJIUSER</td>
<td>Pointer to service specific data area (IAZCVDEV)</td>
</tr>
</tbody>
</table>
Set all other fields in the IAZSSJI control block to binary zeros before issuing the IEFSSREQ macro.

**Convert Device ID service, IAZCVDEV contents:** For the Convert device ID service (function code SSJICVDV) the caller must set the following fields in the IAZCVDEV control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVDVSSID</td>
<td>Eyecatcher of the control block (set to 'CVDV')</td>
</tr>
<tr>
<td>CVDVLEN</td>
<td>Length of the IAZCVDEV (CVDSZE) control block</td>
</tr>
<tr>
<td>CVDEVVRN</td>
<td>Input version of the IAZCVDEV control block. Set to CVDVVER1 for version 1 of the control block. Set to CVDVVER# for the current (latest) version</td>
</tr>
<tr>
<td>CVDVID</td>
<td>Device ID in binary</td>
</tr>
</tbody>
</table>

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

**Output Register Information:** When control returns to the caller, the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>The job information services request completed. Check the SSORETN field for specific function information.</td>
</tr>
<tr>
<td>SSRTNSUP (4)</td>
<td>The subsystem specified in the SSIBSSNM field does not support the job information services function call.</td>
</tr>
<tr>
<td>SSRTNTUP (8)</td>
<td>The subsystem specified in the SSIBSSNM field exists but is not active.</td>
</tr>
<tr>
<td>SSRTNOSS (12)</td>
<td>The subsystem specified in the SSIBSSNM field is not defined to MVS.</td>
</tr>
<tr>
<td>SSRTDIST (16)</td>
<td>The pointer to the SSOB control block or the SSIB control block is not valid, or the function code specified in the SSOFUNC field is greater than the maximum number of functions supported by the subsystem specified in the SSIBSSNM field.</td>
</tr>
</tbody>
</table>
SSI Function Code 71

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
- SSJIRETN
- IAZCVDEV (Convert Device ID service)

SSOBRETN Contents:  When control returns to the caller and register 15 contains a zero, the job information services function places one of the following decimal values in the SSOBRETN field:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIOK (0)</td>
<td>Request successful</td>
</tr>
<tr>
<td>SSJIERRVR (4)</td>
<td>Request completed with possible errors, see SSJIRETN for reason code.</td>
</tr>
<tr>
<td>SSJIERRU (8)</td>
<td>Request cannot be completed because of user error, see SSJIRETN for reason code.</td>
</tr>
<tr>
<td>SSJIERRJ (12)</td>
<td>Request cannot be completed, SSJIRETN contains internal reason code.</td>
</tr>
<tr>
<td>SSJIPARM (16)</td>
<td>The parameter list, ie, the SSJI extension is an invalid format - it is not an SSJI, the service version number is not supported, or the SSJI is not large enough.</td>
</tr>
</tbody>
</table>

SSJIRETN Contents:  In addition to the return code in SSOBRETN, the field SSJIRETN contains the service related error or more specific information about the error. SSJIRETN will be set to one of the following decimal values:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIUNSF (4)</td>
<td>Unsupported subfunction requested</td>
</tr>
<tr>
<td>SSJINTDS (24)</td>
<td>SSJIUSER does not point to the correct data area</td>
</tr>
<tr>
<td>SSJIUNSD (28)</td>
<td>SSJIUSER CB version number is not correct</td>
</tr>
<tr>
<td>SSJISMDS (32)</td>
<td>SSJIUSER CB length is too small</td>
</tr>
</tbody>
</table>

Return codes in SSJIRETN specific to the Convert Device ID service: The following return codes are set if the Convert Device ID service was requested and SSOBRETN is zero:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVDVK (0)</td>
<td>Success</td>
</tr>
</tbody>
</table>

Convert Device ID service, IAZCVDEV contents:  For the Convert Device ID service (function code SSJICVDV) the following is returned in IAZCVDEV:

Field Name  Description

CVDVVERO  Subsystem version number (currently 1)

CVDVNAME  Convert device name in EBCDIC. If the device type is not known, then this will be set to ‘UNKNOWN’

CVDVSZE  Size of IAZCVDEV
Checkpoint Version Information Service
The Checkpoint Versions information service gets or releases control to a version of
the JES2 checkpoint data space.

See the following sections for more information on Checkpoint Version Information
Service:
- "Type of Request"
- "Use Information"
- "Issued to"
- "Related SSI Codes"
- "Related Concepts"
- "Environment"
- "Input Register Information" on page 122
- "Input Parameters" on page 122
- "Output Register Information" on page 124
- "Return Code Information" on page 124
- "Output Parameters" on page 125

Type of Request: Directed SSI Call.

Use Information: To use the JES job information services SSI, a caller must first
decide the function they wish to perform. The appropriate parameter list must be
obtained and pointed to by SSJIUSER.

Issued to: A JES2 subsystem (either primary or secondary). The subsystem does
not have to be associated with the requesting address space.

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:
- IEFSSOBH
- IEFJSSIB
- IAZSSJI
- IAZDSERV (Checkpoint Versions Information service)

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, IAZSSJI, and IAZDSERV control blocks
can reside in 24- or 31-bit virtual storage
Recovery

The caller should provide an ESTAE-type recovery environment. See z/OS MVS Programming: Assembler Services Guide for more information on an ESTAE-type recovery environment.

Figure 15 shows the environment at the time of the call for SSI function code 71, Checkpoint Versions Subfunction.

Figure 15. Environment at Time of Call for SSI Function Code 71, Checkpoint Versions Subfunction

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

**Input Parameters:** Input parameters for the function routine are:
- SSOB
- SSIB
- IAZSSJI
- IAZDSERV (Checkpoint Versions Information service)
**SSOB Contents:** The caller sets the following fields in the SSOB control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB’</td>
</tr>
<tr>
<td>SSOBLLEN</td>
<td>Length of the SSOB (SSOBSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 71(SSOBSJ)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>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</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function-dependent area (IAZSSJI control block)</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB’</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem to which this Job Information Services request is directed</td>
</tr>
</tbody>
</table>

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

**IAZSSJI Contents:** The caller must set the following fields in the IAZSSJI control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIID</td>
<td>Eyecatcher for the control block (set to 'SSJI’)</td>
</tr>
<tr>
<td>SSJILEN</td>
<td>Length of the IAZSSJI (SSJISIZE) control block</td>
</tr>
<tr>
<td>SSJISVRN</td>
<td>Input version of the IAZSSJI control block. Set to SSJISVR# for version 1 of the control block</td>
</tr>
<tr>
<td>SSJIFREQ</td>
<td>Function to be performed on this request. Valid functions and their related SSJIUSER area are:</td>
</tr>
<tr>
<td>Field Value</td>
<td>SSJIUSER Description</td>
</tr>
<tr>
<td>SSJIJOBT</td>
<td>IAZDSERV, Obtain Checkpoint versions information</td>
</tr>
<tr>
<td>SSJIFREL</td>
<td>IAZDSERV, Release Checkpoint versions information</td>
</tr>
<tr>
<td>SSJIUSER</td>
<td>Pointer to service specific data area (IAZDSERV)</td>
</tr>
</tbody>
</table>

Set all other fields in the IAZSSJI control block to binary zeros before issuing the IEFSSREQ macro.
Checkpoint versions information service, IAZDSERV contents: For the Checkpoint versions information service (function codes SSJIFOBT and SSJIFREL) the caller must set the following fields in the IAZDSERV control block on input to a SSJIFOBT function call:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSRVSSID</td>
<td>Eyecatcher of the control block (set to 'DSRV')</td>
</tr>
<tr>
<td>DSRVLEN</td>
<td>Length of the IAZDSERV (DSRVSZE) control block</td>
</tr>
<tr>
<td>DSRVSVRN</td>
<td>Input version of the IAZDSERV control block. Set to DSRVSVR# for the current (latest) version.</td>
</tr>
<tr>
<td>DSRVFLG1</td>
<td>DSERV flags</td>
</tr>
</tbody>
</table>

**DSRVF1LI**  
Use live version

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

For the Checkpoint Versions information service function codes SSJIFREL (return version), the caller should set DSRVCVPT in the IAZDSERV control block to indicate the version to be released.

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>The job information services request completed. Check the SSOBRETN field for specific function information.</td>
</tr>
<tr>
<td>SSRTNSUP (4)</td>
<td>The subsystem specified in the SSIBSSNM field does not support the job information services function call.</td>
</tr>
<tr>
<td>SSRTNTUP (8)</td>
<td>The subsystem specified in the SSIBSSNM field exists but is not active.</td>
</tr>
<tr>
<td>SSRTNOSS (12)</td>
<td>The subsystem specified in the SSIBSSNM field is not defined to MVS.</td>
</tr>
<tr>
<td>SSRTDIST (16)</td>
<td>The pointer to the SSOB control block or the SSIB control block is not valid, or the function code</td>
</tr>
</tbody>
</table>
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:
- **SSOBTRETN**
- **SSJIRETN**
- **IAZDSERV** (Checkpoint Versions Information service)

**SSOBTRETN 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 SSOBTRETN field:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIOK (0)</td>
<td>Request successful</td>
</tr>
<tr>
<td>SSJIERVERVR (4)</td>
<td>Request completed with possible errors, see SSJIRETN for reason code.</td>
</tr>
<tr>
<td>SSJIERRU (8)</td>
<td>Request cannot be completed because of user error, see SSJIRETN for reason code.</td>
</tr>
<tr>
<td>SSJIERRJ (12)</td>
<td>Request cannot be completed, SSJIRETN contains internal reason code.</td>
</tr>
<tr>
<td>SSJIPARM (16)</td>
<td>The parameter list, ie, the SSJI extension is an invalid format - it is not an SSJI, the service version number is not supported, or the SSJI is not large enough.</td>
</tr>
</tbody>
</table>

**SSJIRETN Contents:** In addition to the return code in SSOBTRETN, the field SSJIRETN contains the service related error or more specific information about the error. SSJIRETN will be set to one of the following decimal values:

**When SSOBTRETN is SSJIERVER (4):**

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIOLDD (20)</td>
<td>The data may be obsolete</td>
</tr>
</tbody>
</table>

**When SSOBTRETN is SSJIERRU (8):**

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIUNSF (4)</td>
<td>Unsupported subfunction requested</td>
</tr>
<tr>
<td>SSJI2OBT (8)</td>
<td>Successive obtains without an intervening release requested</td>
</tr>
<tr>
<td>SSJIDISA (12)</td>
<td>Subtask disabled, try again later</td>
</tr>
<tr>
<td>SSJIVINA (16)</td>
<td>Versioning inactive, activate it</td>
</tr>
<tr>
<td>SSJINTDS (24)</td>
<td>SSJIUSER does not point to the correct data area</td>
</tr>
<tr>
<td>SSJIUNSD (28)</td>
<td>SSJIUSER control block version number is not correct</td>
</tr>
<tr>
<td>SSJISMDS (32)</td>
<td>SSJIUSER control block length is too small</td>
</tr>
</tbody>
</table>
SSI Function Code 71

SSJIINVR (36) Invalid input data to release. Could be successive releases without an intervening obtain or release without an intervening obtain or release without having done an obtain.

When SSOBRETN is SSJIERRJ (12):

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>Data space unavailable</td>
</tr>
<tr>
<td>132</td>
<td>Subtask in PJES2</td>
</tr>
<tr>
<td>136</td>
<td>ALESERV error</td>
</tr>
</tbody>
</table>

Return codes in SSJIRETN specific to the Checkpoint versions information service: The following return codes are set if the Checkpoint versions information service was requested and SSOBRETN is zero:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJIOK (0)</td>
<td>Success</td>
</tr>
</tbody>
</table>

Checkpoint version information service, IAZDSERV contents: For the Checkpoint versions information service (function codes SSJIFOBT and SSJIFREL) the following is returned in IAZDSERV:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSRVCVPT</td>
<td>Pointer to Checkpoint version</td>
</tr>
<tr>
<td>DSRVCNUM</td>
<td>Version number</td>
</tr>
<tr>
<td>DSRVJOTK</td>
<td>JOT token</td>
</tr>
<tr>
<td>DSRVJOPT</td>
<td>Pointer to JOT</td>
</tr>
<tr>
<td>DSRVJOAL</td>
<td>ALET of JOT</td>
</tr>
<tr>
<td>DSRVJQTK</td>
<td>JQE token</td>
</tr>
<tr>
<td>DSRVJQPT</td>
<td>Pointer to JQE</td>
</tr>
<tr>
<td>DSRVJQAL</td>
<td>ALET of JQE</td>
</tr>
<tr>
<td>DSRVQSTK</td>
<td>QSE token</td>
</tr>
<tr>
<td>DSRVQSPT</td>
<td>Pointer to QSE</td>
</tr>
<tr>
<td>DSRVQSAI</td>
<td>ALET of QSE</td>
</tr>
<tr>
<td>DSRVHCTK</td>
<td>HCT token</td>
</tr>
<tr>
<td>DSRVHCPT</td>
<td>Pointer to HCT</td>
</tr>
<tr>
<td>DSRVHCAL</td>
<td>ALET of HCT</td>
</tr>
<tr>
<td>DSRVTIME</td>
<td>Time stamp</td>
</tr>
<tr>
<td>SSI Function Code 71</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td><strong>DSRVSZE1</strong></td>
<td>Version 1 length</td>
</tr>
<tr>
<td><strong>DSRVJNTK</strong></td>
<td>JNT token</td>
</tr>
<tr>
<td><strong>DSRVJNPT</strong></td>
<td>Pointer to JNT</td>
</tr>
<tr>
<td><strong>DSRVJNAL</strong></td>
<td>ALET of JNT</td>
</tr>
<tr>
<td><strong>DSRVSZE2</strong></td>
<td>Version 2 length</td>
</tr>
<tr>
<td><strong>DSRVJQXK</strong></td>
<td>JQX token</td>
</tr>
<tr>
<td><strong>DSRVJZPT</strong></td>
<td>Pointer to JQX</td>
</tr>
<tr>
<td><strong>DSRVJXLAL</strong></td>
<td>ALET of JQX</td>
</tr>
<tr>
<td><strong>DSRVJTTK</strong></td>
<td>JQE trackgroup extension token</td>
</tr>
<tr>
<td><strong>DSRVJTPT</strong></td>
<td>Pointer to JQE trackgroup extension</td>
</tr>
<tr>
<td><strong>DSRVJTAL</strong></td>
<td>ALET of JQE trackgroup extension</td>
</tr>
<tr>
<td><strong>DSRVDASK</strong></td>
<td>DAS token</td>
</tr>
<tr>
<td><strong>DSRVDAPT</strong></td>
<td>Pointer to DAS</td>
</tr>
<tr>
<td><strong>DSRVDAAL</strong></td>
<td>ALET of DAS</td>
</tr>
<tr>
<td><strong>DSRVSZE3</strong></td>
<td>Version 3 length</td>
</tr>
<tr>
<td><strong>DSRVFLG1</strong></td>
<td>DSERV flags</td>
</tr>
<tr>
<td><strong>DSRVF1LI</strong></td>
<td>Use live version</td>
</tr>
<tr>
<td><strong>DSRVRQSV1</strong></td>
<td>Reserved for future use</td>
</tr>
<tr>
<td><strong>DSRVSZE4</strong></td>
<td>Version 4 length</td>
</tr>
<tr>
<td><strong>DSRVWQSK</strong></td>
<td>WLM Q position token</td>
</tr>
<tr>
<td><strong>DSRVWQST</strong></td>
<td>Pointer to WQPOS</td>
</tr>
<tr>
<td><strong>DSRVWQL</strong></td>
<td>ALET of WQPOS</td>
</tr>
<tr>
<td><strong>DSRVSZE5</strong></td>
<td>DSERV Version 5 fixed parameter length</td>
</tr>
<tr>
<td><strong>DSRVJOXK</strong></td>
<td>JOX token</td>
</tr>
<tr>
<td><strong>DSRVOXPT</strong></td>
<td>Pointer to JOX</td>
</tr>
<tr>
<td><strong>DSRVXAL</strong></td>
<td>ALET of JOX</td>
</tr>
<tr>
<td><strong>DSRVCNPT</strong></td>
<td>Reserved for subsystem use</td>
</tr>
<tr>
<td><strong>DSRVSZE6</strong></td>
<td>DSERV Version 6 fixed parameter length</td>
</tr>
<tr>
<td><strong>DSRVSZE</strong></td>
<td>Current version length</td>
</tr>
</tbody>
</table>
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
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 SSNU control blocks can reside in storage above 16 megabytes.

Recovery
The caller should provide an ESTAE-type recovery environment. See z/OS MVS Programming: Authorized Assembler Services Guide for more information on an ESTAE-type recovery environment.
**Input Register Information**

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>
### SSI Function Code 75

#### 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 75 (SSOBSSNU)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>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.</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function dependent area (SSNU control block)</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>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.</td>
</tr>
<tr>
<td>SSIBSUSE</td>
<td>(JES3 only) Subsystem use — the SSIBSUSE value that was returned upon completion of the Request Job ID call (SSI function code 20).</td>
</tr>
</tbody>
</table>

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.

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSNUID</td>
<td>Identifier 'SSNU'</td>
</tr>
<tr>
<td>SSNULEN</td>
<td>Length of the SSNU (SSNUSIZE) control block</td>
</tr>
<tr>
<td>SSNUVER</td>
<td>Version of mapping for the caller — Set this field to SSNUCVER (an IBM-defined integer constant within the SSNU control block).</td>
</tr>
</tbody>
</table>
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. It is only valid for authorized callers. For unauthorized callers, the value will be ignored and a SSNEURCD of 40 will be set.

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 home node 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.

SSNUMEMB  JES2 member name on which send should be issued if the user is not logged on and if a shared broadcast is not used (OUTDEF BRODCAST=NO). Not used by JES3 as the send is always issued from the global.

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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>The Notify User Message Service request was processed. Check the SSOBRETN field for specific function information.</td>
</tr>
<tr>
<td>SSRTNSUP (4)</td>
<td>The subsystem specified in the SSIBSSNM field does not support this function.</td>
</tr>
<tr>
<td>SSRTNTUP (8)</td>
<td>The subsystem specified in the SSIBSSNM field exists, but is not active.</td>
</tr>
</tbody>
</table>
### SSI Function Code 75

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSNUOK</td>
<td>The message was issued successfully. The SSNUERCD field contains a zero (SSNUROQOK).</td>
</tr>
<tr>
<td>SSNUOKB</td>
<td>The message was issued successfully but had a minor error. See the SSNUERCD field in the SSNU control block for the specific reason code.</td>
</tr>
<tr>
<td>SSNUERR</td>
<td>The message was not issued. See the SSNUERCD field in the SSNU control block for the specific reason code.</td>
</tr>
<tr>
<td>SSNUNEX</td>
<td>The value for SSOBRETN means that SSOBINDV does not point to a valid SSNU control block. For example, SSOBINDV can be zero or in JES2 the eyecatcher pointed to by SSOBINDV is not 'SSNU'.</td>
</tr>
</tbody>
</table>

#### 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSNUROQOK</td>
<td>The request was successful.</td>
</tr>
<tr>
<td>SSNUMSGT</td>
<td>The request was successful, but the message text was truncated because it was too long.</td>
</tr>
<tr>
<td>SSNUEXC</td>
<td>A user exit cancelled the request (JES2 only). In JES2, exit 42 may have requested the cancellation of the message.</td>
</tr>
<tr>
<td>SSNUNUSR</td>
<td>An invalid userid was specified (blanks or zeros).</td>
</tr>
<tr>
<td>SSNUINVD</td>
<td>An invalid nodename was supplied. The message was not issued.</td>
</tr>
<tr>
<td>SSNUVID</td>
<td>An invalid identifier (SSNUID) was supplied. The message was not issued.</td>
</tr>
<tr>
<td>SSNUIVER</td>
<td>An invalid version of the SSNU control block was</td>
</tr>
</tbody>
</table>
supplied. The message was not issued. The value supplied in the SSNUVER field is not valid. Both JES2 and JES3 will issue this return value if SSNUVER is zero. JES3 will also issue this return value if SSNUVER is at a higher level than 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.

SSNUUUTK (40) The request was successful, but the user token is not allowed for an unauthorized caller.

SSNUINVE (44) Indicates an invalid extension (incorrect length) has been provided.

SSNUMEME (48) An incorrect member name was specified in SSNUMEMB. The message was issued with a default member specification.
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 136.
• 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 142.

• BULK MODIFY
  Modifies selected attributes of one or more data sets. The BULK MODIFY call is described on "BULK MODIFY Requests" on page 143.

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:
  – SSS2CREA (Owning Userid)
  – SSS2DEST (Destination)
  – SSS2FORM (Form Numbers)
  – SSS2JBIL (Job ID)
SSI Function Code 79

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

If any of the preceding validity check fails, the application thread is not terminated.

If the validity check for SSS2 passes, JES will set appropriate SSOBRETN code and will terminate the thread during the final call processing (SSS2CTRL set by application).

If the datasets within the clone JOE are not disposed of uniformly (even though application has indicated that its thread is terminating), JES2 will set SSS2CLON return code in SSOBRETN.

- 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 17 on page 138.
Figure 17. Protocol for the SAPI PUT/GET Call (Part 1 of 2)
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 SJRETSTUP can be set with the contents of SSS2SWTU when issuing this request. Set field SJRSTSWBN 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 occurred 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 z/OS MVS Programming: Authorized Assembler Services Reference 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.

Note: In JES3 you can override the default number of buffers to be used when reading from the data set by specifying the text unit for BUFNO. The default is 2 spool tracks of buffers. Specifying 0 or 1 will cause the default to be used.

The subsequent dynamic allocation call is depicted in Figure 18 on page 142.
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 19 on page 143. If the application does provide for multi-tasking, each thread in the application address space must follow the protocol shown in Figure 19 on page 143.
**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.

In certain situations the BULK MODIFY request may not be successful. This is normal, and can occur if the output being released/returned is BUSY. There is no ECB posted and no error code returned; JES2 will always return to the caller, but the actual request may be partially or completely bypassed.

To assure that the desired changes are processed, it may be necessary to query the status of a job or specific job output, and if it appears that a request did not complete, then the BULK MODIFY request can be reissued.

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 20 on page 145](#). If the application does provide for multi-tasking, each thread in the application address space must follow the protocol shown in [Figure 20 on page 145](#).
**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.

The CTOKEN maps the JES dependent portion of the client (SYSOUT) token (mapped by IAZCTKN). The client (SYSOUT) token has a length defined by the field, CTKNSIZE. The CTOKEN specifically maps the field, CTKNJESD, in IAZCTKN. The JES dependent portion of the client (SYSOUT) token contains the information that JES needs to uniquely identify and locate the data set represented by the client (SYSOUT) token. Also, a bit map from the CTOKEN maps the field, CTKNBMAP, in IAZCTKN. The bit map provides information as to which parts of the client (SYSOUT) token are valid for comparison between client (SYSOUT) tokens.

**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.
- Selection by token is requested.

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
• IAZSS2

Your application thread must meet the following requirements:

Minimum Authorization
Problem state, any PSW key.

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 z/OS MVS Programming: Authorized Assembler Services Guide for more information on an ESTAE-type recovery environment.

Figure 21 on page 148 shows the environment at the time of the call for SSI function code 79.
Before issuing the IEFSSREQ macro, your application thread must ensure that the following general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier SSOB</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 79 (SSOBSSOU2)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of an SSIB control block or zero. (If this field is zero, the life-of-job SSIB is used.) See &quot;Subsystem Identification Block (SSIB)&quot; on page 8 for more about the life-of-job SSIB.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier SSIB</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>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.</td>
</tr>
<tr>
<td>SSIBJBID</td>
<td>Job identifier — the job ID that was returned upon completion of the Request Job ID call (SSI function code 20).</td>
</tr>
<tr>
<td>SSIBSUSE</td>
<td>(JES3 only) Subsystem use — the SSIBSUSE value that was returned upon completion of the Request Job ID call (SSI function code 20).</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS2LEN</td>
<td>Input field</td>
</tr>
<tr>
<td></td>
<td>The length of the SSS2, set with the value SSS2SIZE</td>
</tr>
<tr>
<td>SSS2VER</td>
<td>Input field</td>
</tr>
<tr>
<td></td>
<td>Set with the current value of SSS2CVER</td>
</tr>
<tr>
<td>SSS2EYE</td>
<td>Input field</td>
</tr>
<tr>
<td></td>
<td>Eye catcher</td>
</tr>
<tr>
<td></td>
<td>Set with the character string SSS2</td>
</tr>
<tr>
<td>SSS2TYPE</td>
<td>Input field</td>
</tr>
<tr>
<td></td>
<td>Type of call. Set with either SSS2PUGE, SSS2COUN, or SSS2BULK below.</td>
</tr>
<tr>
<td>SSS2PUGE</td>
<td>Request type of PUT/GET.</td>
</tr>
<tr>
<td></td>
<td>Find a data set matching the selection criteria.</td>
</tr>
<tr>
<td>SSS2COUN</td>
<td>Request type of Count.</td>
</tr>
</tbody>
</table>
|            | Find the number of schedulable elements (OSEs/JOEs) 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 SSS2DES2 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
SSS2SJBH  Use SSS2JOBH 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.
SSS2SJBH  Use SSS2JBL and SSS2JBHI 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
SSI Function Code 79

**SSI2SU**S  Use SSI2UCS as a filter

**SSI2SEL3**  Input field

-used for selection of new data sets.

**SSI2STC**  Select Started Tasks (STCs) (see note in SSI2STYP)

**SSI2STSU**  Select Time Sharing Users (TSUs) (see note in SSI2STYP)

**SSI2SJOB**  Select batch jobs (JOBs) (see note in SSI2STYP)

**SSI2SAPC**  Select APPC output (see note in SSI2STYP)

**SSI2STYP**  If none of these bits is on, then selection is as if all of the bits are on.

**SSI2SEL4**  Input field

-used for selection of new data sets.

**SSI2SMOD**  Use SSI2MOD as a filter

**SSI2SFLS**  Use SSI2FLSH as a filter

**SSI2SAGE**  Data sets selected must be at least as old as the value in SSI2AGE.

**SSI2SLIN**  Use minimum and maximum line counts specified in SSI2LMIN and SSI2LMAX as a data set group filter

**SSI2SPAG**  Use minimum and maximum page counts specified in SSI2PMIN and SSI2PMAX as a data set group filter

**SSI2SPRI**  Select output based on priority

**SSI2SVOL**  Select output based on the volume serial list in SSI2VOL (SSI2NVOL in SSI2RET2 on if the JES does not support)

**SSI2SCHR**  Use Printer translation tables in SSI2CHAR as a filter (SSI2NCHR in SSI2RET2 on if the JES does not support)

**SSI2SEL5**  Input field

-used for selection of new data sets.

**SSI2SCPN**  Select data set having no CPDS.

**SSI2SCTK**  Select by client token. Mutually exclusive with SSI2SJB1. 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.

**SSI2SBRO**  Use SAPI as a “browse” facility rather than a “processing” facility.

**SSI2SODS**  Use SSI2ODST as a filter.

**SSI2SRON**  SSI2SRON indicates that this is a non-update type selection and denotes that the application intent is read-only.
If this bit is on, JES will perform READ access requests for the data sets selected and give an error return code if an attempt is made to update the status of a data set that was obtained with read access. The error will cause the current thread to be terminated.

The indicators SSS2RNPT (do not show to thread again) and SSS2RNPR (do not show to address space again) will be honored for non-update type calls.

The default SAF access for SAPI requests is UPDATE in the JESSPOOL class. Specifying SSS2SRON means that the application guarantees that no modification of the data sets will be attempted and thus READ access to the JESSPOOL class is all that is required. If the application attempts to modify the data sets in any way other than setting SSS2RNPT (do not show to thread again) or setting SSS2RNPR (do not show to this address space again), the thread will be terminated with return code of SSS2BDIS in SSOBRETN and a reason code of SSS2RRON in SSS2REAS. SSS2SRON applies to PUTGET requests only.

**SSS2SENL**

SSS2SENL is used with SSS2SLIN.

If this bit is on, JES2 will enforce the use of minimum and maximum line counts specified in SSS2LMIN and SSS2LMAX as a data set group filter for data sets with a line count of zero. The default logic for JES2 work selection is to ignore line limits when the data sets have a line count of zero.

**SSS2SENP**

SSS2SENP is used with SSS2SPAG.

If this bit is on, JES2 will enforce the use of minimum and maximum page counts specified in SSS2PMIN and SSS2PMAX as a data set group filter for data sets with a page count of zero. The default logic for JES2 work selection is to ignore page limits when the data sets have a page count of zero.

**SSS2SEL6**

Input field

Used for selection of new data sets.

**SSS2STPN**

Transaction job name filtering.

If this bit is on, SYSOUT associated with a transaction job name that matches SSS2JOBN is selected. In addition, SYSOUT that is owned by a job that matches SSS2JOBN is also returned.

The SSS2STPN bit is ignored if one of the following situations occurs:
SSI Function Code 79

- SSS2SJBN or SSS2SDUP is not set.
- JES2 is used but JES2 is not running with checkpoint mode z11.

**SSS2STPI**
Transaction ID filtering.

If this bit is on, SYSOUT associated with a transaction job id in the range specified by SSS2JBIL and SSS2JBIH are selected. In addition, SYSOUT that is owned by a job that has a job id in this range is also returned.

The SSS2STPI bit is ignored if one of the following situations occurs:

- SSS2SJBI is not set.
- JES2 is used but JES2 is not running with checkpoint mode z11.

**SSS2MSC1**
Processing flags

**SSS2CTRL**

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

- **Off**
  Normal processing is to occur depending on the value of the SSS2TYPE field (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 is on).

When SSS2JBIL is 2-8 characters and starts with one of the prefixes 'J', 'JO', 'JOB', or '*', then the suffix is converted to a binary value. Job IDs with a suffix matching the SSS2JBIL suffix are
returned. Embedded and trailing blanks are acceptable. The maximum length of the jobid is eight characters.

When SSS2JBIL contains 1-8 character with one or more generic characters '* ' and '?', and EBCDIC characters A-Z; 0-9; or national characters @, #, $, then job IDs, as returned in SSS2JBIR, that match a 1-8 character EBCDIC comparison with SSS2JBIL are returned. A single character SSS2JBIL with '*' or '?' is not allowed. SSS2JBIH must be blank.

When SSS2STPI is set and SSS2JBIL is 2-8 characters starting with the prefix '*', then the suffix is converted to a binary value. Transaction job IDs with a suffix matching the SSS2JBIL suffix are also returned.

When SSS2STPI is set and SSS2JBIL contains 1-8 EBCDIC character A-Z; 0-9; national characters @, #, $; or generic characters '*' and '?', then transaction job IDs that match a 1-8 character EBCDIC comparison with SSS2JBIL are also returned. A single character SSS2JBIL with '*' or '?' is not allowed. When generic characters are used, SSS2JBIH must be blank.

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

In a JES2 environment, for STCs and TSUs, the jobid must be passed in the format of xxxnnnnn where xxx is JOB, JO, or J, or the format of xxxnnnnn where xxx is JO, J0, or J.

SSS2JBIH

Input field

High jobid used for selection (if SSS2JBI on). This value must be null or at least as high as SSS2JBIL.

When SSS2JBIH is 2-8 characters and starts with one of the prefixes 'J', 'JO', 'JOB', then the suffix is converted to a binary value. Job IDs with a suffix within the range from the SSS2JBIL suffix through the SSS2JBIH suffix are returned. Generics characters '*' or '?' are not allowed. Embedded and trailing blanks are acceptable. The maximum length of the jobid is eight characters.

When SSS2STPI is set, EBCDIC characters A-Z; 0-9; and national characters @, #, $ are allowed. Job IDs, as returned in STTRJID, and transaction job IDs within the 1-8 character EBCDIC range from SSS2JBIL through SSS2JBIH, are returned. Generics characters '*' or '?' are not allowed.

The following table describes examples of jobs returned for SSS2JBIL when SSS2JBIH is blank. Numeric matches are in normal font, EBCDIC matches are in italicized font.

<table>
<thead>
<tr>
<th>SSS2JBIL</th>
<th>Examples of standard job ID matches</th>
<th>Examples of transaction job ID matches if SSS2STPI is on</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOB00100</td>
<td>Numeric match(es): JOB00100 or TSU00100, and so on. EBCDIC match(es): not applicable.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): JOB00100</td>
</tr>
</tbody>
</table>
Table 3. Examples of jobs returned for SSS2JBIL when SSS2JBIH is blank (continued)

<table>
<thead>
<tr>
<th>SSS2JBIL</th>
<th>Examples of standard job ID matches</th>
<th>Examples of transaction job ID matches if SSS2STPI is on</th>
</tr>
</thead>
<tbody>
<tr>
<td>J100</td>
<td>Numeric match(es): JOB00100 or TSU00100, and so on. EBCDIC match(es): not applicable.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): J100.</td>
</tr>
<tr>
<td>A100</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): not applicable. Error if SSS2STPI is not on.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): A100</td>
</tr>
<tr>
<td>*0000100</td>
<td>Numeric match(es): JOB00100 or TSU00100, and so on. EBCDIC match(es): no additional matches.</td>
<td>Numeric match(es): JOB00100, A0000100, Z0000100, ZZZZZ100, etc. EBCDIC match(es): no additional matches.</td>
</tr>
<tr>
<td>*100</td>
<td>Numeric match(es): JOB00100 or INT00100, and so on. EBCDIC match(es): JOB09100, T9999100, and so on.</td>
<td>Numeric match(es): JOB00100, A0000100, Z0000100, Z999100, and so on. EBCDIC match(es): no additional matches.</td>
</tr>
<tr>
<td>*5555555</td>
<td>Numeric match(es): JO5555555 or ST5555555 etc. EBCDIC match(es), T9555555, J8555555, and so on.</td>
<td>Numeric match(es): JO5555555, ZZ5555555, and so on. EBCDIC match(es): ZZ5555555, 55555555, and so on.</td>
</tr>
<tr>
<td>J*</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): JOB00100, JO123456, J7654321.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): JOB00100, JO123456, J7654321, J9, JAMES, and so on.</td>
</tr>
<tr>
<td>*0001?0</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): not applicable.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): A0000110, Z00001A0, K0001J0, KT001P0 and so on.</td>
</tr>
<tr>
<td>ZZ#00100</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): not applicable. Error if SSS2STPI is not on.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): ZZ#00100</td>
</tr>
</tbody>
</table>

**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.  
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, local, anylocal, or NJE. Also, wildcards are not supported for destinations defined by DESTID initialization statements. For more information, see the topic on Controlling JES2 Processes in [z/OS JES2 Initialization and Tuning Reference](https://www.ibm.com/support/knowledgecenter/SSS2SCW_2.3.0/com.ibm.zos.r23.jes2initref.doc/sg246943.pdf).

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...
SSI Function Code 79

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.

The minimum number of lines (records) generated by an output group must not fall below this value if SSS2SLIN is set on, if JES2 is to consider the output group selectable. JES2 checks record limits if the data set is in line mode.

**SSS2LMAX**

Input field

Used for selection of new data sets.

The maximum number of lines (records) generated by an output group must not exceed this value if SSS2SLIN is on, if JES2 is to consider the output group selectable. JES2 checks record limits if the data set is in line mode.

**SSS2PMIN**

Input field

Used for selection of new data sets.

The minimum number of pages generated by an output group must not fall below this value if SSS2SPAG is set on, if JES2 is to consider the output group selectable. JES2 checks the page limits (SSS2PMIN and SSS2PMAX) if the data set is in page mode. If the page data set contains some line mode data, then JES2 checks both page limits and record limits (SSS2LMIN and SSS2LMAX).

**SSS2PMAX**

Input field

Used for selection of new data sets.

The maximum number of pages generated by an output group must not exceed this value if SSS2SPAG is set on, if JES2 is to consider the output group selectable. JES2 checks the page limits (SSS2PMIN and SSS2PMAX) if the data set is in page mode. If the page data set contains some line mode data, then JES2 checks both page limits and record limits (SSS2LMIN and SSS2LMAX).

**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. In JES3, the absence of the keep bit implies that the data set will be deleted.

In JES2, if SSS2DKPE is off and the data set has OUTDISP=KEEP, the data set will have OUTDISP=LEAVE after processing. If SSS2DKPE is off and the data set does not have OUTDISP=KEEP, the data set is deleted regardless of other disposition settings in this section.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS2DKPE</td>
<td>Keep the data set</td>
</tr>
<tr>
<td>SSS2RHLD</td>
<td>Keep the data set and make it non-selectable (system hold)</td>
</tr>
<tr>
<td>SSS2RNPR</td>
<td>Keep the data set and leave it selectable, but never return to this SAPI address space</td>
</tr>
<tr>
<td></td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>• The JES is restarted</td>
</tr>
<tr>
<td></td>
<td>• The application is restarted</td>
</tr>
<tr>
<td></td>
<td>• Some characteristic is changed by the operator or another application</td>
</tr>
<tr>
<td></td>
<td>• Selection by token is requested</td>
</tr>
<tr>
<td>SSS2DHLD</td>
<td>Hold the data set</td>
</tr>
<tr>
<td></td>
<td>This bit is mutually exclusive with SSS2DRLS.</td>
</tr>
<tr>
<td>SSS2DRLS</td>
<td>Release the data set</td>
</tr>
<tr>
<td></td>
<td>This bit is mutually exclusive with SSS2DHLD.</td>
</tr>
<tr>
<td>SSS2CHKP</td>
<td>Use SSS2RBA to checkpoint the data set position. Next data set returned will have SSS2DSF on.</td>
</tr>
<tr>
<td>SSS2DNWR</td>
<td>Set writer name to a null value (all X'40's).</td>
</tr>
<tr>
<td>SSS2RNPT</td>
<td>Leave the data set selectable, but never return to this SYSOUT API thread again. The data set could be seen by the thread if:</td>
</tr>
<tr>
<td></td>
<td>• The JES is restarted</td>
</tr>
<tr>
<td></td>
<td>• Some characteristic is changed by the operator to another application</td>
</tr>
<tr>
<td></td>
<td>• Selection by token is requested</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSS2DSP2</th>
<th>Input field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Flags describing the disposition for the data set whose name is currently in SSS2DSN.</td>
</tr>
<tr>
<td>SSS2RPRI</td>
<td>Use the priority in SSS2DPRI.</td>
</tr>
<tr>
<td>SSS2DNFO</td>
<td>Set forms code to the installation default</td>
</tr>
</tbody>
</table>
The following fields (SSS2DCLS, SSS2DFOR, SSS2DPGM, SSS2DDES, SSS2CLFT, and SSS2DPRI) 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.

SSS2DPRI  Input field
           New data set priority. A value of 0 through 255 is accepted. This field is meaningful only if SSS2RPRI is on in SSS2DSP2. This specification is supported by JES2 only.

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

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.
SSI Function Code 79

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 SSORETN and SSS2.

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

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS2RTOK (0)</td>
<td>Successful completion; for SAPI PUT/GET calls, a data set was returned whose name is in SSS2DSN</td>
</tr>
<tr>
<td>SSS2EODS (4)</td>
<td>No more data sets to select</td>
</tr>
</tbody>
</table>

See the reason codes defined for SSS2REAS at “Reason Codes for SSORETN being SSS2EODS” on page 164.

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS2INVA (8)</td>
<td>Invalid search arguments</td>
</tr>
<tr>
<td>SSS2UNAV (12)</td>
<td>Unable to process now</td>
</tr>
<tr>
<td>SSS2DUPJ (16)</td>
<td>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.</td>
</tr>
<tr>
<td>SSS2IDST (20)</td>
<td>Invalid destination specified</td>
</tr>
<tr>
<td>SSS2TKNM (28)</td>
<td>Token map failed. Application will not be allowed to allocate to data set and DISP=(,KEEP) will be forced</td>
</tr>
<tr>
<td>SSS2LERR (32)</td>
<td>Logic error</td>
</tr>
</tbody>
</table>

See the reason codes defined for SSS2REAS at “Reason Codes for SSORETN being SSS2LERR” on page 163.

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS2ICLS (36)</td>
<td>SSS2CLAS not A-Z and not 0-9</td>
</tr>
<tr>
<td>SSS2BDIS (40)</td>
<td>Disposition settings incorrect</td>
</tr>
</tbody>
</table>

See the reason codes defined for SSS2REAS at “Reason Codes for SSORETN being SSS2BDIS” on page 164.

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS2CLON (44)</td>
<td>Disposition for data set group not uniform (See SSS2DSH). DISP=(,KEEP) is forced with no override disposition information honored</td>
</tr>
</tbody>
</table>
**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:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS2RENI (4)</td>
<td>SSS2JEST zero, but SSS2DSN not null</td>
</tr>
<tr>
<td>SSS2REIP (8)</td>
<td>SSS2SIPA and SSS2SIPN are mutually exclusive</td>
</tr>
<tr>
<td>SSS2RALO (12)</td>
<td>Prior data set still allocated</td>
</tr>
<tr>
<td>SSS2RDUP (16)</td>
<td>SSS2SDUP on in SSS2SEL1 and wildcards used in SSS2JOBN</td>
</tr>
<tr>
<td>SSS2RJBI (20)</td>
<td>SSS2JBIH &lt; SSS2JBIL and SSS2SJBIs on</td>
</tr>
<tr>
<td>SSS2RCRE (24)</td>
<td>SSS2CREA has error and SSS2SCRE on</td>
</tr>
<tr>
<td>SSS2RLEN (28)</td>
<td>SSS2LEN is less than SSS2SIZE</td>
</tr>
<tr>
<td>SSS2RTYP (32)</td>
<td>SSS2TYPE is not valid</td>
</tr>
<tr>
<td>SSS2RDES (36)</td>
<td>SSS2DEST has error and SSS2SDST on</td>
</tr>
<tr>
<td>SSS2RJNM (40)</td>
<td>SSS2JOBN has error and SSS2SJBIs on</td>
</tr>
<tr>
<td>SSS2RFRM (44)</td>
<td>SSS2FORM has error and SSS2SRFM on</td>
</tr>
<tr>
<td>SSS2RPGM (48)</td>
<td>SSS2PGMN has error and SSS2SPGM on</td>
</tr>
<tr>
<td>SSS2RPRM (52)</td>
<td>SSS2PRMO has error and SSS2SPRM on</td>
</tr>
<tr>
<td>SSS2RCLS (56)</td>
<td>SSS2CLSL has error and SSS2SCLS on</td>
</tr>
<tr>
<td>SSS2RFCB (60)</td>
<td>SSS2FCB has error and SSS2SFCB on</td>
</tr>
<tr>
<td>SSS2RUCS (64)</td>
<td>SSS2UCS has error and SSS2SUCS on</td>
</tr>
<tr>
<td>SSS2RCHR (68)</td>
<td>SSS2CHAR has error and SSS2SCHR on</td>
</tr>
<tr>
<td>SSS2RMO (72)</td>
<td>SSS2MOD has error and SSS2SMOD on</td>
</tr>
<tr>
<td>SSS2RF (76)</td>
<td>SSS2FLS has error and SSS2SFLS on</td>
</tr>
<tr>
<td>SSS2RLPM (80)</td>
<td>SSS2LMIN or SSS2LMAX is negative and SSS2SLIN is on -- or -- SSS2PMIN or SSS2PMAX is negative and SSS2SPAG is on</td>
</tr>
<tr>
<td>SSS2RLPG (84)</td>
<td>SSS2LMIN &gt; SSS2LMAX and SSS2SLIN on -- or -- SSS2PMIN &gt; SSS2PMAX and SSS2SPAG on</td>
</tr>
<tr>
<td>SSS2RDE2 (88)</td>
<td>SSS2DES2 has error and SSS2TYPE is SSS2BULK and SSS2ROUT on</td>
</tr>
<tr>
<td>SSS2RVOL (92)</td>
<td>SSS2VOL has error and SSS2SVOL on</td>
</tr>
<tr>
<td>SSS2REYE (96)</td>
<td>SSS2EYE does not have SSS2</td>
</tr>
<tr>
<td>SSS2RCTK (100)</td>
<td>SSS2SCTK is on but SSS2CTKN is not specified or not valid.</td>
</tr>
<tr>
<td>SSS2RBRO (104)</td>
<td>SSS2SBRO is on but Bulk Modify or Count was requested.</td>
</tr>
<tr>
<td>SSS2RECJ (108)</td>
<td>SSS2SCTK and SSS2SJBIs are mutually exclusive.</td>
</tr>
<tr>
<td>SSS2RODS (112)</td>
<td>SSS2ODST has error and SSS2SODS on</td>
</tr>
</tbody>
</table>
SSI Function Code 79

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:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS2RDCL (184)</td>
<td>SSS2DCLS has error</td>
</tr>
<tr>
<td>SSS2RDFR (188)</td>
<td>SSS2DFOR has error</td>
</tr>
<tr>
<td>SSS2RDPG (192)</td>
<td>SSS2DPGM has error</td>
</tr>
<tr>
<td>SSS2RDDS (196)</td>
<td>SSS2DDES has error</td>
</tr>
<tr>
<td>SSS2RDHR (200)</td>
<td>Both SSS2DHLD and SSS2DRLS specified</td>
</tr>
<tr>
<td>SSS2RRON (204)</td>
<td>SSS2SRON was set (application requested READ only access), but an attempt was made to alter a characteristic of the data set or to delete the data set.</td>
</tr>
</tbody>
</table>

Reason codes 208 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:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS2RENM (240)</td>
<td>No matching output</td>
</tr>
<tr>
<td>SSS2RENS (244)</td>
<td>Matching output not selectable</td>
</tr>
</tbody>
</table>

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 163 and "Reason Codes for SSOBRETN being SSS2BDIS."

**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.
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
Transaction Program Jobname that created this data set

SSS2TJID Output field
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
SSI Function Code 79

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. The value is derived from the value specified as the DSID keyword on a DD statement, which is only used for 3540 Diskette data sets.

**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
<table>
<thead>
<tr>
<th>SSI2NVOL</th>
<th>Selecting output based on a volume serial list not supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSI2NNHD</td>
<td>Returning addresses of NJE headers not supported</td>
</tr>
<tr>
<td>SSI2NMOD</td>
<td>Selecting output based on a modification is not supported</td>
</tr>
<tr>
<td>SSI2NPRI</td>
<td>Selecting output in priority order is not supported</td>
</tr>
<tr>
<td>SSI2NIPA</td>
<td>IP address selection not supported. Turned on if JES does not support and SSI2SIPA or SSI2SIPN is on</td>
</tr>
</tbody>
</table>

### SSI2RET3

**Output field**
- **SSI2RSTC**: Data set created by started task
- **SSI2RTSU**: Data set created by time sharing user
- **SSI2JOB**: Data set created by batch job

### SSI2RET4

**Output field**
- **SSI2CPDS**: Data set has page mode data
- **SSI2SPUN**: Data set was spun at close
- **SSI2SDSH**: All data sets in group must be unallocated identically.

### SSI2RET5

**Output field**
- **SSI2HLV**: Data set on “HOLD/LEAVE” queue (JES2) or “Hold for TSO” queue (JES3)
- **SSI2RXWH**: Data set on “Hold for XWTR” queue. This will never be true in a JES2 environment.
- **SSI2RHOL**: Data set on one of the held queues.
- **SSI2RWTR**: Data set on “Write/Keep” queue (JES2) or “Writer” queue (JES3).
- **SSI2RFOR**: Record format

The following count fields (SSI2LNCT, SSI2PGCT, SSI2BYCT, and SSI2RCCT) are valid only if SSI2DSCL is on in SSI2RET1.

The fields represent counts for the single data set returned if SSI2TYPE is SSI2PUGE. The fields represent the total for all data sets selected if SSI2TYPE is SSI2COUN.

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.

### SSI2LNCT

**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.

### SSI2PGCT

**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.
SSI Function Code 79

**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, JES2 does not return this value. Meanwhile, for JES3 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 page 139 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 page 139 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. SSS2DSTR is filled in by JES after the token is constructed. After processing the data set, JES constructs the data set token in storage controlled by JES. JES then moves the address of the token into the field, SSS2DSTR. JES then returns the IAZSSSS2 parameter list to the application. To have the token available for use on a refetch, the user must save the entire token in storage controlled by the user. When refetching the data set, SSS2CTKN must point to the entire...
SSI Function Code 79

saved token and SSS2SCTK bit must be turned on in flag, SSS2SEL5. Note that the token has a length defined by the field, CTKNSIZE, in IAZCTKN.

**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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS2PNAM</td>
<td>Output field</td>
</tr>
<tr>
<td></td>
<td>Programmer name from the JOB statement</td>
</tr>
<tr>
<td>SSS2ROOM</td>
<td>Output field</td>
</tr>
<tr>
<td></td>
<td>Job level room number</td>
</tr>
<tr>
<td>SSS2NOTN</td>
<td>Output field</td>
</tr>
<tr>
<td></td>
<td>Job notify node</td>
</tr>
<tr>
<td>SSS2NOTU</td>
<td>Output field</td>
</tr>
<tr>
<td></td>
<td>Job notify userid</td>
</tr>
<tr>
<td>SSS2ACCT</td>
<td>Output field</td>
</tr>
<tr>
<td></td>
<td>Address of encoded accounting information</td>
</tr>
<tr>
<td></td>
<td>Accounting information is provided in ‘SMF’ format, just as if it is in type 5 and type 30 SMF records.</td>
</tr>
<tr>
<td></td>
<td><strong>AL1(number-of-pairs-that-follow)</strong></td>
</tr>
<tr>
<td></td>
<td>followed by 0 or more pairs of the form:</td>
</tr>
<tr>
<td></td>
<td><strong>AL1(length),CLlength’string’</strong></td>
</tr>
<tr>
<td></td>
<td>A length of 0 indicates an omitted field</td>
</tr>
<tr>
<td></td>
<td>Example: Accounting information of (X3600,42,,ANDY):</td>
</tr>
<tr>
<td></td>
<td>DC AL1(4) Nr of fields</td>
</tr>
<tr>
<td></td>
<td>DC AL1(5),CL5’X3600’ field 1</td>
</tr>
<tr>
<td></td>
<td>DC AL1(2),CL2’42’ field 2</td>
</tr>
<tr>
<td></td>
<td>DC AL1(0) field 3 (null)</td>
</tr>
<tr>
<td></td>
<td>DC AL1(4),CL4’ANDY’ field 4</td>
</tr>
<tr>
<td>SSS2XEQ</td>
<td>Output field</td>
</tr>
<tr>
<td></td>
<td>Node where job executed</td>
</tr>
<tr>
<td>SSS2ORG</td>
<td>Output field</td>
</tr>
<tr>
<td></td>
<td>Node where job entered system</td>
</tr>
<tr>
<td>SSS2TIME</td>
<td>Output field</td>
</tr>
<tr>
<td></td>
<td>Time on input processor for the selected job. This is in hundredths of seconds since midnight. The time field is local, not UCT/GMT.</td>
</tr>
<tr>
<td>SSS2DATE</td>
<td>Output field</td>
</tr>
<tr>
<td></td>
<td>Date on input processor for the selected job. This is in the form 0cyydddF. The date field is local, not UCT/GMT.</td>
</tr>
<tr>
<td>SSS2SYS</td>
<td>Output field</td>
</tr>
<tr>
<td></td>
<td>System name of the MVS image where the job output was created.</td>
</tr>
</tbody>
</table>
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.

**SSS2USID** Output field

Network account number.

Copy of JMRUSEID for the job being returned. This field should be used in SMF records associated with the printing of the job.

**SSS2MXRC** Output field

Maximum RC (Return Code) for the job.

Supplied by JES2 only.

**SSS2LSAB** Output field

Last ABEND code for the job.

Supplied by JES2 only.
SSI Function Code 80

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.

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 returns job information and SYSOUT status information. Callers can request either terse or verbose information. Terse requests return less data but have lower overhead because no I/O is required. Verbose requests return more detailed data, but involve multiple I/O requests. For this reason, verbose requests are limited in how much data can be obtained in a single SSI invocation. See “Use Information for Verbose requests” on page 174 for information about verbose requests.

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. It is recommended that verbose calls be directed (not broadcast) to a specific subsystem.

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. 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. 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 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.

The status request does not provide a way to freeze the job and data set status in the system. Other SAPI applications, JES writers, networking writers, and operators may change the state of any job or data set received in the status response. In general, the bigger the time lag between the status request and the use of the information, the bigger the chance that either some other function may have processed the data set or that a new output may have arrived.

The response to an extended status request will include data elements for jobs and sysout which match the original request, chained from the STATJOBF field in IAZSSST (STAT).

There are four types of data elements (control blocks) returned, depending on the type of request made:
- SJQE - Job Queue Element (chained from IAZSSST)
- SJVE - Job Queue Verbose Element (chained from SJQE)
- SOUT - SYSOUT Element (chained from SJQE)
- SSVE - SYSOUT Verbose Element (chained from SOUT and SJQE)

Based on the input type you requested, the following table describes the output data elements that are returned.
Use Information for Verbose requests

With version 4 of the IAZSSST macro, you can request verbose information for both jobs and SYSOUT.

**Note:** If you are running on JES3, the JES3 release of the global must be z/OS V1R7 or higher.

In general, verbose JOB or SYSOUT data can be obtained for a single job in three ways;
1. jobid
2. token
3. or to expand data obtained previously by a terse address (STATTRSA, with no intervening STATMEM call).

- **Obtaining verbose job level data**
  Set STATTYPE to STATVRBO and also set one of the following input fields:
  - Set STATTRSA to zeros and ensure that the job ID filters specified by STATSJBI refer to the same job ID in STATJBIL and STATJBIH (or that STATJBIH is set to zero). Both terse and verbose job data are returned.
  - Set STATTRSA to zeros and STATSCTK has STATCTKN set to the SYSOUT token you want verbose data for. Both terse and verbose data are returned.
  - Set STATTRSA to a STATJQ or STATSE (obtained previously with no intervening memory management call). The related STATJQ will chain to a verbose element (STATVE).

- **Obtaining verbose SYSOUT level data**
  Set STATTYPE to STATOUTV and also set one of the following input fields:
  - Set STATTRSA to zero and ensure that the job ID filters specified by STATSJBI refer to the same job ID in STATJBIL and STATJBIH (or that STATJBIH is set to zero). Both terse and verbose data are returned. Verbose data is also returned for all valid SYSOUT data sets (chained into the STATJQ). If the job is still executing, STATVOs for data sets that are still open may also be returned. Lastly, terse SYSOUT data is returned. The STATVOs are chained into the STATSEs with which they are associated.
  - Set STATTRSA to zero and STATSCTK has STATCTKN set to the SYSOUT token of the SYSOUT group for which you want verbose data. Both terse and verbose job and SYSOUT data are returned (only for the data sets represented by the token passed).
  - Set STATTRSA to a STATJQ (obtained previously with no intervening memory management call). Similar to the case in which STATSJBI is set, verbose job
data will be chained into the STATJQ, STATVOs will be obtained for all valid
SYSOUT data sets, and STATSEs will be obtained for all SYSOUT groups for
the job.

– Set STATTRSA to a STATSE (obtained previously with no intervening memory
management call). Similar to the case in which STATSCTK is set, verbose job
data will be obtained for the job, and all the STATVOs related to the STATSE.

Notes:
1. Additional SYSOUT filters can be set (bits in STATSSLx) when STATTRSA is
set to STATJQ or STATSBTI. The STATSE are built that match the SYSOUT
filters and then STATVOs are built that correspond to each of the STATSEs.
2. Setting STATTRSA to a STATJQ or STATSE is available in JES2 only.

Use Information for data set list requests
A list of all data sets associated with a single job can be requested by setting
STATTYPE to STATDLST. Since this is considered a verbose type call (I/O is
required to obtain the needed information), only information about a single job can
be requested (STATTRSA is supported).

Note the following about STATDLST calls:

• Information on all data sets is returned including instream (SYSIN) data sets,
internal data sets, data set that will not print, and data sets that might have been
already processed and “deleted”. You can determine the type of data set being
returned by examining bits in the STVSFLG1 byte.

• One SYSOUT verbose element (STATVO) is returned per data set instance.
Each STATVO will have a single SYSOUT terse section (STATSE). This includes
instream (SYSIN) data sets. Data set grouping does not affect how data is
returned. If JES3 is the subsystem returning information, and the data set has
not been processed by output processing, the STATSE and STATSO will be
mostly null (except for the data set name and token). This is because output
processing is where JES3 resolves the various sources of output characteristics.

• SYSOUT and JOB filters can be used to limit the amount of data that is returned.

• Values for data returned will NOT always reflect attribute changes made after the
data set was created (including changes made via operator command, SWB
modify services, and exits).

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:
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 IAZSSST control blocks can reside above or below 16 megabytes in virtual storage.
- **Recovery**: The caller should provide an ESTAE-type recovery environment. See [z/OS MVS Programming: Assembler Services Guide](https://www.ibm.com) for more information about an ESTAE-type recovery environment.

Figure 22 shows the environment at the time of the 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register 1</td>
<td>SSOB</td>
</tr>
<tr>
<td></td>
<td>'SSOB' (SSOBID)</td>
</tr>
<tr>
<td></td>
<td>Length (SSOBLEN)</td>
</tr>
<tr>
<td></td>
<td>Function ID (SSOBFUNC)</td>
</tr>
<tr>
<td></td>
<td>SSIB (SSOBSSIB)</td>
</tr>
<tr>
<td></td>
<td>Function Dependent Area (SSOBINDV)</td>
</tr>
<tr>
<td></td>
<td>SSIB</td>
</tr>
<tr>
<td></td>
<td>Length (SSIBLEN)</td>
</tr>
<tr>
<td></td>
<td>Subsystem Name (SSIBSSNM)</td>
</tr>
<tr>
<td></td>
<td>SSST</td>
</tr>
<tr>
<td></td>
<td>Length (STATLEN)</td>
</tr>
<tr>
<td></td>
<td>Eyecatcher (STATEYE)</td>
</tr>
<tr>
<td></td>
<td>Version (STATVER)</td>
</tr>
<tr>
<td></td>
<td>Output Version (STATVER0)</td>
</tr>
<tr>
<td></td>
<td>Reason Code (STATREAS)</td>
</tr>
<tr>
<td></td>
<td>Reason Code 2 (STATREAS2)</td>
</tr>
<tr>
<td></td>
<td>Type (STATTYPE)</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Figure 22. Environment at Time of Call for SSI Function Code 80
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.

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLNEN</td>
<td>Length of the SSOB (SSOBHLSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 80 (SSOBEKST)</td>
</tr>
<tr>
<td>SSOBSIB</td>
<td>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.</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function-dependent area (IAZSSST control block).</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier ‘SSIB’</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem to which this extended status function call is directed (or MSTR if it is to be broadcast).</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATLEN</td>
<td>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.</td>
</tr>
<tr>
<td>STATEYE</td>
<td>Eyecatcher for the control block (set to C’STAT’)</td>
</tr>
<tr>
<td>STATVER</td>
<td>Input version of the IAZSSST control block (Set to STATV010 for the initial version of the control block, STATV020 for OS/390</td>
</tr>
</tbody>
</table>
### STATTYPE

Function to be performed on this request. Valid functions are:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATTERS</td>
<td></td>
<td>Requests obtaining terse job level data. The data returned on this call does not require large amounts of system overhead.</td>
</tr>
<tr>
<td>STATVRBO</td>
<td></td>
<td>Requests obtaining verbose job level data. The data returned on this call includes that returned for the terse job level call. STATVER must be set to at least STATV040 for this request to be valid. The request requires system overhead for I/O to obtain the data.</td>
</tr>
<tr>
<td>STATMEM</td>
<td></td>
<td>Return memory from a previous request. After one or more requests for data, the memory obtained must be returned using this function.</td>
</tr>
<tr>
<td>STATOUTT</td>
<td></td>
<td>Requests obtaining terse SYSOUT level data (including job level information). Data returned on this call does not require large amounts of system overhead. STATVER must be set to at least STATV030 for this request to be valid.</td>
</tr>
<tr>
<td>STATOUTV</td>
<td></td>
<td>Requests obtaining verbose SYSOUT level data. The data returned on this call includes that returned for the terse SYSOUT level call. STATVER must be set to at least STATV040 for this request to be valid. The request requires system overhead for I/O to obtain the data. Terse and verbose output for running jobs is also returned.</td>
</tr>
<tr>
<td>STATDLST</td>
<td></td>
<td>Requests data set list for a job. This request obtains verbose type information for all data sets associated with a job. It includes information on SYSIN and other internal data sets. It is only valid for STATV060 and above callers.</td>
</tr>
</tbody>
</table>

### STVBMLRC

Max LRCECL of JCLIN stream.

The caller can also set the following fields in the IAZSST control block on input to limit (or select) the jobs for which data will be returned:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSEL1</td>
<td>Flag byte which describes the filters to use to select jobs. Each bit corresponds to a filter field which must match any job returned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSCSLS</td>
<td>Apply job class filter in STATCLSL or STATCLSP. Only one class needs to match. If only specifying one class it must be specified in STATCLSL.</td>
</tr>
<tr>
<td>STATSDST</td>
<td>Apply default destination filter in STATDEST or STATDSTP. Only one destination needs to match. If only specifying one destination it must be specified in STATDEST.</td>
</tr>
</tbody>
</table>
STATSJBN  Apply job name filter in STATJOBN or STATJBNP. Only one job name needs to match. If only specifying one job name it must be specified in STATJOBN.

STATSJBII  Apply job ID filters in STATJBIL and STATJBIH. STATSJBII cannot be specified with STATSCRTK.

STATSOJI  Apply original job ID filter inSTATOJBI. Not supported in JES3.

STATSOWN  Apply current owner filter in STATOWNR.

STATSSEC  Apply current SECLABEL filter in STATSECL.

STATSSUB  Apply submitter filter in STATSUBR (only supported by JES3).

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 STATPRI.
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.

STATSYS  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. STATVER must be set to STATV020 or greater to use this filter.

**Notes:**
1. In a JES2 environment, WLM service class queue information is always returned regardless of the setting for STATPOS.
2. In a JES3 environment, WLM service class queue information is returned only if STATPOS is specified.
3. Be aware that there is a performance penalty in a JES3 environment when STATPOS is set and there is a large number of jobs in a service class queue along with a request that selects a large number of those jobs.

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.

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSCLX</td>
<td>Apply job class filter in STATCLSL and STATCLSP only to jobs in STAT-SELECT or STAT-ONMAIN phases.</td>
</tr>
<tr>
<td>STATSOJD</td>
<td>Do not apply job name filter in STATJOBN and STATJOBNP and job id filters in STATJBIIL and STATJBIH to jobs that created OUTPUT with STST1APC on. (Only supported by JES2.)</td>
</tr>
<tr>
<td>STATSQPS</td>
<td>Always return current job position in the queue (even if a special queue scan is necessary).</td>
</tr>
<tr>
<td>STATSORG</td>
<td>Apply origin node filter in STATORGN. Only supported by JES2.</td>
</tr>
<tr>
<td>STATSEQ</td>
<td>Apply execution node filter in STATXEQN. Only supported by JES2.</td>
</tr>
<tr>
<td>STATSRV</td>
<td>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.</td>
</tr>
<tr>
<td>STATSEN</td>
<td>Apply scheduling environment filter in STATSENV.</td>
</tr>
</tbody>
</table>

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. STATVER must be set to STATV030 or greater to use these filters.

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSCTK</td>
<td>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.</td>
</tr>
<tr>
<td>STATSSOW</td>
<td>Apply the SYSOUT owner filter in STATSCRE.</td>
</tr>
<tr>
<td>STATSSDS</td>
<td>Apply the SYSOUT destination filter in STATSDES. STATSDSP also contains additional SYSOUT destination filters. Mutually exclusive with STATSSLC or STATSSNT.</td>
</tr>
<tr>
<td>STATSSCL</td>
<td>Apply the SYSOUT class filter in STATSCLA. STATSCLP also contains additional SYSOUT class filters.</td>
</tr>
<tr>
<td>STATSSWR</td>
<td>Apply the SYSOUT external writer filter in STATSWTR.</td>
</tr>
<tr>
<td>STATSSHL</td>
<td>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. It also includes SYSOUT that is held by an operator command or by the system due to a processing error. Setting both STATSSHL and STATSSNH on is the same as setting both bits off.</td>
</tr>
<tr>
<td>STATSSNH</td>
<td>Select SYSOUT that is not currently held. Setting both STATSSHL and STATSSNH on is the same as setting both bits off.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSSL2</td>
<td>Flag byte that describes the SYSOUT filters to use to select verbose data to return. If JOB data is requested (STATVRBO), then only jobs with SYSOUT that match the specified filters are returned. If SYSOUT data is requested (STATOUTV), then data for SYSOUT that matches these filters is returned along with the corresponding job level data. STATVER must be set to STATV040 or higher and the JES processing the request must be at the z/OS V1R7 level or higher to use these filters. On JES3, the global must be at the z/OS V1R7 level or higher.</td>
</tr>
<tr>
<td>STATSSFR</td>
<td>Apply the SYSOUT forms name filter in STATSFOR. STATVER must be set to STATV040 or higher and the JES processing the request must be at the z/OS V1R7 level or higher to use these filters. On JES3, the global must be at the z/OS V1R7 level or higher.</td>
</tr>
</tbody>
</table>
| STATSSPR  | Apply the SYSOUT PRMODE filter in STATSPRM. STATVER must be set to STATV040 or higher and the JES processing the request must be at the z/OS V1R7 level or higher.
SSI Function Code 80

z/OS V1R7 level or higher to use these filters. On JES3, the global must be at the z/OS V1R7 level or higher.

**STATSSSP**
Apply the Select SPIN output only filter in STATSSSP. STATVER must be set to STATV040 or higher and the JES processing the request must be at the z/OS V1R7 level or higher to use these filters. On JES3, the global must be at the z/OS V1R7 level or higher.

*Note:* STATSSSP and STATSSNS are mutually exclusive. If STATSSSP and STATSSNS are both ON or both OFF, then the spin state of the output will not be considered.

**STATSSNS**
Apply the non-SPIN output only filter in STATSSNS. STATVER must be set to STATV040 or higher and the JES processing the request must be at the z/OS V1R7 level or higher to use these filters. On JES3, the global must be at the z/OS V1R7 level or higher.

*Note:* STATSSSP and STATSSNS are mutually exclusive. If STATSSSP and STATSSNS are both ON or both OFF, then the spin state of the output will not be considered.

**STATSSIP**
Select SYSOUT elements that are routed to an IP address. STATVER must be set to STATV050 or higher and the JES processing the request must be at the z/OS V1R9 level or higher to use these filters.

**STATSSNP**
Select SYSOUT elements that are not routed to an IP address. STATVER must be set to STATV050 or higher and the JES processing the request must be at the z/OS V1R9 level or higher to use these filters.

**STATSSOD**
When on with STATSSOW, it indicates to match if SYSOUT is destined to STATSCRE on the local node. STATVER must be set to STATV050 or higher and the JES processing the request must be at the z/OS V1R9 level or higher to use these filters.

**STATSSJD**
This is a JES2 only bit.

- If JES2 is running with checkpoint mode z2 in R11 and STATSJBN is on, it indicates to match if SYSOUT is destined to STATJOBN or STATJBNP on the local node (ignored if STATSJBN is off).
- If JES2 is running with checkpoint mode z11 in R11 and STATSJBN and STATSTPN are on, it indicates to match if SYSOUT is destined to STATJOBN, STATJBNP or transaction job name on the local node (ignored if STATSJBN is off).
STATJOBN Job name filter (used if STATSJB 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 left justified and padded on the right with blanks. When STATJBIL is 2-8 characters and starts with one of the prefixes 'J', 'JO', 'JOB', 'T', 'TS', 'TSU', 'S', 'ST', 'STC', 'I', 'IN', 'INT', or '*', then the suffix is converted to a binary value. Job IDs with a suffix matching the STATJBIL suffix are returned. The prefix character '*' is not allowed for verbose requests.

When STATJBIL contains 1-8 characters with one or more generic characters '*' and '?', and EBCDIC characters A-Z; 0-9; or national characters @, #, $, then job IDs, as returned in STTRJID, that match a 1-8 character EBCDIC comparison with STATJBIL are returned. A single character STATJBIL with '*' or '?' is not allowed. STATJBIIH must be blank. Generics characters '*' or '?' are not allowed for verbose requests.

When STATSTPI is set and STATJBIL is 2-8 characters starting with the prefix '*', then the suffix is converted to a binary value. Transaction job IDs with a suffix matching the STATJBIL suffix are also returned.

When STATSTPI is set and STATJBIL contains 1-8 EBCDIC characters A-Z; 0-9; national characters @, #, $; or generic characters '*' and '?', then transaction job IDs, as returned in STSAJID, that match a 1-8 character EBCDIC comparison with STATJBIL are also returned. A single character STATJBIL with '*' or '?' is not allowed. When generic characters are used, STATJBIIH must be blank.

Note: INT and IN are valid only in JES3.

STATJBIIH High job ID value (used if STATSJBI is set). If this field is not specified, then information is only returned using the filter specified in STATJBIL. When STATJBIIH is 2-8 characters and starts with one of the prefixes 'J', 'JO', 'JOB', 'T', 'TS', 'TSU', 'S', 'ST', 'STC', 'I', 'IN', or 'INT', then the suffix is converted to a binary value. Job IDs with a suffix within the range from the STATJBIL suffix through the STATJBIIH suffix are returned. Generics characters '*' or '?' are not allowed.

When STATSTPI is set, EBCDIC characters A-Z; 0-9; and national characters @, #, $ are allowed. Job IDs, as returned in STTRJID, and transaction job IDs, as returned in STSAJID, within the 1-8 character EBCDIC range from STATJBIL through STATJBIIH, are returned. Generics characters '*' or '?' are not allowed.

Note: INT and IN are valid only in JES3.

The following table describes examples of jobs returned for STATJBIL when STATJBIIH is blank. Numeric matches are in normal font, EBCDIC matches are in italicized font.
Table 5. Examples of jobs returned for STATJBIL when STATJBIH is blank.

<table>
<thead>
<tr>
<th>STATJBIL</th>
<th>Examples of standard job ID matches</th>
<th>Examples of transaction job ID matches if STATSTPI is on</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOB00100</td>
<td>Numeric match(es): JOB00100 or TSU00100, and so on. EBCDIC match(es): not applicable.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es):JOB00100</td>
</tr>
<tr>
<td>J100</td>
<td>Numeric match(es): JOB00100 or TSU00100, and so on. EBCDIC match(es): not applicable.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es):J100.</td>
</tr>
<tr>
<td>A100</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): not applicable. Error if STATSTPI is not on.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es):A100</td>
</tr>
<tr>
<td>*0000100</td>
<td>Numeric match(es): JOB00100 or TSU00100, and so on. EBCDIC match(es): no additional matches.</td>
<td>Numeric match(es): JOB00100, A0000100, Z100, ZZZZZ100, and so on. EBCDIC match(es): no additional matches.</td>
</tr>
<tr>
<td>*100</td>
<td>Numeric match(es): JOB00100 or INT00100, and so on. EBCDIC match(es): JOB09100, T9999100, and so on.</td>
<td>Numeric match(es): JOB00100, A0000100, Z100, and so on. EBCDIC match(es): JOB09100, T9999100, Z99100, and so on.</td>
</tr>
<tr>
<td>*555555</td>
<td>Numeric match(es): J5555555 or T5555555, and so on. EBCDIC match(es): no additional matches.</td>
<td>Numeric match(es): J5555555, A5555555, Z5555555, and so on. EBCDIC match(es): 55555555</td>
</tr>
<tr>
<td>J*</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): JOB00100, JO123456, J7654321.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es):JOB00100, JO123456, J7654321, J9, JAMES, and so on.</td>
</tr>
<tr>
<td>STATJBIL</td>
<td>Examples of standard job ID matches</td>
<td>Examples of transaction job ID matches if STATSTPI is on</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>*0001?0</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): not applicable.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): A0000110, Z00001A0, K0001J0, KT001P0, and so on.</td>
</tr>
<tr>
<td>ZZ#00100</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): not applicable. Error if STATSTPI is not on.</td>
<td>Numeric match(es): not applicable. EBCDIC match(es): ZZ#00100</td>
</tr>
</tbody>
</table>

**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. Not supported in JES3.

**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 STATSSSEC 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. Only supported by JES2.

**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. Only supported by JES2.

**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**
This keyword is supported when requesting information from a JES2 subsystem only. 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:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT_INPUT</td>
<td>Job is active in input processing</td>
<td></td>
</tr>
<tr>
<td>STAT_WTCONV</td>
<td>Job is queued for conversion</td>
<td></td>
</tr>
<tr>
<td>STAT_CONV</td>
<td>Job is actively converting</td>
<td></td>
</tr>
<tr>
<td>STAT_VOLWT</td>
<td>Job is queued for SETUP (not currently used by JES2 code)</td>
<td></td>
</tr>
<tr>
<td>STAT_SETUP</td>
<td>Job is active in SETUP (not currently used by JES2 code)</td>
<td></td>
</tr>
<tr>
<td>STAT_SELECT</td>
<td>Job is queued for execution</td>
<td></td>
</tr>
<tr>
<td>STAT_ONMAIN</td>
<td>Job is actively executing</td>
<td></td>
</tr>
<tr>
<td>STAT_SPIN</td>
<td>JES2 is processing SPIN data sets for the JOB</td>
<td></td>
</tr>
</tbody>
</table>
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 is 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:

<table>
<thead>
<tr>
<th>Phase Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAT_NOSUB</td>
<td>No subchain exists</td>
</tr>
<tr>
<td>STAT_FSSCI</td>
<td>Job is active in conversion/interpretation in an FSS address space</td>
</tr>
<tr>
<td>STAT_PSCBAT</td>
<td>Job is awaiting postscan (batch)</td>
</tr>
<tr>
<td>STAT_PSCDSL</td>
<td>Job is awaiting postscan (demand select)</td>
</tr>
<tr>
<td>STAT_FETCH</td>
<td>Job is awaiting volume fetch</td>
</tr>
<tr>
<td>STAT_VOLWT</td>
<td>Job is awaiting start setup</td>
</tr>
<tr>
<td>STAT_SYSSEL</td>
<td>Job is awaiting or active in MDS system select processing</td>
</tr>
<tr>
<td>STAT_ALLOC</td>
<td>Job is awaiting resource allocation</td>
</tr>
<tr>
<td>STAT_VOLUAV</td>
<td>Job is awaiting unavailable volume(s)</td>
</tr>
<tr>
<td>STAT_VERIFY</td>
<td>Job is awaiting volume mount(s)</td>
</tr>
<tr>
<td>STAT_SYSVER</td>
<td>Job is awaiting or active in MDS system verification processing</td>
</tr>
<tr>
<td>STAT_ERROR</td>
<td>Job encountered an error during MDS processing</td>
</tr>
<tr>
<td>STAT_SELECT</td>
<td>Job is awaiting selection on main</td>
</tr>
</tbody>
</table>
SSI Function Code 80

STAT_ONMAIN
Job is scheduled on main

STAT_BRKDOWN
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 for 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 STATSEN 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.

STATTOPT1
Option byte

STAT1RAC
If on, requests that the RACF authorization checks be made whether or not the caller of the SSI is APF authorized. This bit has no effect if the caller is not APF authorized. The RACF check that is made if the SECLABEL class is active is a dominance check of the seclabel of the job/SYSOUT compared to the seclabel of the requestor. This check is a JES2 only check.

STAT1LCL
Specifies that the destination information returned in fields STTRONOND, STTRXNOD, STTRPRRE/STTRPRND, STTRPURE/STTRPUND, and STSTDEST should suppress the local node name. If the destination is the local node, and there is no secondary routing information, the LOCAL
is returned (instead of the local node name). If the
destination is a secondary routing at the local node, then
only the secondary routing is returned (for example, R1 is
returned if the destination is remote at the local node). This
option does not affect destinations information returned for
the destinations other than the local node. The setting of
the JES2 parameter DESTDEF SHOWUSER= will influence
what is returned if this bit is on. This bit only applies to
JES2.

STATSSL3

More SYSOUT selection criteria. This is supported if STATVER is
STATV050 or higher and the corresponding JES2 is z/OS V1R9 or
higher.

STATSSLC

Select SYSOUT that is destined to the local node. If
STATSSLC and STATSSNT are both on or both off, then
the destination of the output will not be considered.
However, either bit being on is mutually exclusive with
STATSSDS being set.

STATSSNT

Select SYSOUT that is not destined to the local node. If
STATSSLC and STATSSNT are both on or both off, then
the destination of the output will not be considered.
However, either bit being on is mutually exclusive with
STATSSDS being set.

STATSSNJ

For selection purposes, treat SYSOUT destined to an NJE
node as OUTDISP of WRITE regardless of the actual
OUTDISP. This has no effect if STATSWRT, STATSHOL,
STATSKEP and STATSLVE are all on or all off.

STATSWRT

Select output hat has an OUTDISP of WRITE.

STATSHOL

Select output that has an OUTDISP of HOLD.

STATSKEP

Select output that has an OUTDISP of KEEP.

STATSLVE

Select output that has an OUTDISP of LEAVE.

Note: Setting STATSWRT, STATSHOL, STATSKEP and STATSLVE
all on has the same effect as setting them all off.

STATSSL4

STATSSL4 is used to support filtering of information returned based
on transaction name, transaction job id, or transaction owner.

STATSTPN

Transaction job name filtering.

If this bit is on, information about jobs and SYSOUT
associated with a transaction job name that matches
STATJOBN or STATJBNP is returned.

The STATSTPN bit is ignored if one of the following
situations occurs:

• STATSJBN is not set.
SSI Function Code 80

- Requesting verbose information
- STATSOJD is not set for JES2.
- JES2 is used but JES2 is not running with checkpoint mode z11.

**STATSTPI**
Transaction job ID filtering.
- If STATSTPI is not set, only jobs and SYSOUT that has a job id in the range specified by STATJBIL and STATJBIH are returned.
- If STATSTPI is set, jobs and SYSOUT associated with a SYSOUT data set with a transaction job id are also selected. The job id is in the range specified by STATJBIL and STATJBIH.

The STATSTPI bit is ignored if one of the following situations occurs:
- STATSJBI is not set.
- Requesting verbose information
- STATSOJD is not set for JES2.
- JES2 is used but JES2 is not running with checkpoint mode z11.

**STATSTPU**
SYSOUT owner filtering.
If this bit is on, jobs and SYSOUT that are associated with a SYSOUT data set whose transaction owner matches STATOWNR are returned.

The STATSTPU bit is ignored if one of the following situations occurs:
- STATSOWN is not set.
- Requesting verbose information
- JES2 is used but JES2 is not running with checkpoint mode z11.

**STATTRSA**
Pointer to a STATJQ or STATSE (or zero) for which verbose data is to be obtained. If non-zero, use this terse address to expand data obtained previously through a terse JOB or SYSOUT extended status call (with no intervening STATMEM call). Only valid when both STATVER and JES2 are at a STATV040 level or higher. On JES3, the calling system must be at the z/OS V1R10 level or higher and the global system must be at the z/OS V1R7 level or higher.

**STATCTKN**
Pointer to the SYSOUT token to be used for selection (used if STATSTCK 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.

**STATSFOR**
The SYSOUT forms name for selection (used if STATSSFR is set). The forms name is 1-8 characters, left justified, and padded on the right with blanks. The generic characters ‘*’ and ‘?’ are allowed. On JES3, the global must be at the z/OS V1R7 level or higher.

**STATSPRM**
The process mode name for selection (used if STATSSPR is set). The process mode name is 1-8 characters, left justified, and padded on the right with blanks. The generic characters ‘*’ and ‘?’ are allowed. On JES3, the global must be at the z/OS V1R7 level or higher.

**STATSUBR**
The submitting userid for selection (used if STATSSUB is set). The submitting userid is 1-8 characters, left justified, and padded on the right with blanks. The generic characters ‘*’ and ‘?’ are allowed. This filter is available in JES3 only. On JES3, the global must be at the z/OS V1R7 level or higher.

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.

There are fields that relate to the additional input filters. Each filter is a count followed by a pointer to a list of values. Any one value that matches is considered passing. You must place the first value in the base field. Failure to do so will result in an invalid parameter error. For example, to filter on the job classes A, B, C, or D you would set the following:

- **STATCLSL = C‘A’**
- **STATCLSN = F‘3’**
- **STATCLSP = A(CLASSLST)**
- **CLASSLST = CL8‘B’,CL8‘C’,CL8‘D’**

<table>
<thead>
<tr>
<th>Filter (base field)</th>
<th>Selection Bit</th>
<th>Flag Byte (contains selection bit)</th>
<th>Array Pointer</th>
<th>Array Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATCLSL</td>
<td>STATCLS</td>
<td>STATSEL1</td>
<td>STACLSP</td>
<td>STATCLSN</td>
<td>Job class filters</td>
</tr>
<tr>
<td>STATJOBN</td>
<td>STATJBN</td>
<td>STATSEL1</td>
<td>STATJBNP</td>
<td>STATJBNN</td>
<td>Job name filters</td>
</tr>
<tr>
<td>STATDEST</td>
<td>STATDEST</td>
<td>STATSEL1</td>
<td>STATDSTP</td>
<td>STATDSTN</td>
<td>Job destination filters</td>
</tr>
<tr>
<td>STATPHAZ</td>
<td>STATPHAZ</td>
<td>STATSEL3</td>
<td>STATPHZP</td>
<td>STATPHZN</td>
<td>Job phase filters</td>
</tr>
</tbody>
</table>

Table 6. SSI Function Code 80 Filters

Chapter 3. SSI Function Codes Your Program Can Request 191
Table 6. SSI Function Code 80 Filters (continued)

<table>
<thead>
<tr>
<th>Filter (base field)</th>
<th>Selection Bit</th>
<th>Flag Byte (contains selection bit)</th>
<th>Array Pointer</th>
<th>Array Count</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSCLA</td>
<td>STATSSCL</td>
<td>STATSSL1</td>
<td>STATSCLP</td>
<td>STATSCLN</td>
<td>SYSOUT class filters</td>
</tr>
<tr>
<td>STATSDES</td>
<td>STATSSDS</td>
<td>STATSSL1</td>
<td>STATSDSP</td>
<td>STATSDSN</td>
<td>SYSOUT destination filters</td>
</tr>
</tbody>
</table>

The new fields are as follows:

**Field Name**  **Description**

STATCLSN  Additional job class count
STATCLSP  Pointer to STATCLSL extension containing additional job class filters
STATJBNN  Additional job name count
STATJBNP  Pointer to STATJOBN extension containing additional job name filters
STATDSTN  Additional job destination count
STATDSTP  Pointer to STATDEST extension containing additional job destination filters
STATPHZN  Additional job phase count
STATPHZP  Pointer to STATPHAZ extension containing additional job phase filters
STATSCLN  Additional SYSOUT class count
STATSCLP  Pointer to STATSCLA extension containing additional SYSOUT class filters
STATSDSN  Additional SYSOUT destination count
STATSDSP  Pointer to STATSDES extension containing additional SYSOUT destination filters.

These new filters are only honored if STATVER is STATV050 or higher and JES2 is z/OS V1R9 or higher or JES3 is z/OS V1R10 or higher.

**Output Register Information**

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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**  **Meaning**

(Decimal)
### 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.

### STATRTRS (148)
STATTERS or STATOUTT requested with incorrect STATCTKN type.

### 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:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATRTOK (0)</td>
<td>Input parameters were valid, check STATJOBF for output.</td>
</tr>
<tr>
<td>STATINVA (4)</td>
<td>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).</td>
</tr>
<tr>
<td>STATLERR (8)</td>
<td>Logic error in one of the search arguments. See output parameter STATREAS (below) for details as to the exact error.</td>
</tr>
<tr>
<td>STATINVT (12)</td>
<td>The request type in STATTYPE is not valid.</td>
</tr>
</tbody>
</table>

**STATREAS Contents:** When SSOBRETN contains an 8 (STATLERR) indicating a logic error, the field STATREAS indicates the specific error detected. STATREAS will be set to one of the following decimal values:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATRDST (4)</td>
<td>Destination in STATDEST is not valid.</td>
</tr>
<tr>
<td>STATRJBL (8)</td>
<td>Low job ID in STATJBIL is not valid.</td>
</tr>
<tr>
<td>SSI Function Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>STATRJBH (12)</td>
<td>High job ID in STATJBIH is not valid.</td>
</tr>
<tr>
<td>STATRJLM (16)</td>
<td>The high job ID in STATJBIH is less than the low job ID in STATJBL.</td>
</tr>
<tr>
<td>STATRCLS (20)</td>
<td>Job class in STATCLSL is not valid.</td>
</tr>
<tr>
<td>STATRVOL (24)</td>
<td>The volume list in STATVOL is null or has characters that are not that are not allowed.</td>
</tr>
<tr>
<td>STATRJBH (28)</td>
<td>The phase specified in STATPHAZ is either not valid or not supported by this subsystem.</td>
</tr>
<tr>
<td>STATRQUE (32)</td>
<td>Unable to access job queue.</td>
</tr>
<tr>
<td>STATREYE (36)</td>
<td>The eyecatcher in STATEYE is not C'STAT'.</td>
</tr>
<tr>
<td>STATRLEN (40)</td>
<td>The length of the IAZSSST specified in STATLEN is too short.</td>
</tr>
<tr>
<td>STATRJBN (44)</td>
<td>The job name in STATJOBN is not valid.</td>
</tr>
<tr>
<td>STATROWN (48)</td>
<td>The owning userid in STATOWNR is not valid.</td>
</tr>
<tr>
<td>STATRSYS (52)</td>
<td>The system name in STATSYS is not a valid system name.</td>
</tr>
<tr>
<td>STATRMEM (56)</td>
<td>The member name in STATMEMB is not valid.</td>
</tr>
<tr>
<td>STATRCST (60)</td>
<td>STATSEL2 specifies to select only non-batch jobs and batch job class selection was specified in STATSCLS.</td>
</tr>
<tr>
<td>STATROJB (64)</td>
<td>Original job ID in STATOJBI is not valid.</td>
</tr>
<tr>
<td>STATRSEC (68)</td>
<td>The SECLABEL in STATSECL is not valid.</td>
</tr>
<tr>
<td>STATRORG (72)</td>
<td>The origin node in STATORGN is not defined.</td>
</tr>
<tr>
<td>STATRXEQ (76)</td>
<td>The execution node in STATXEQN is not defined.</td>
</tr>
<tr>
<td>STATRPRI (80)</td>
<td>The priority in STATPRIO is not valid for this JES.</td>
</tr>
<tr>
<td>STATRSVC (84)</td>
<td>The service class in STATSRVC is not valid.</td>
</tr>
<tr>
<td>STASSSEN (88)</td>
<td>The scheduling environment in STATTSEN is not valid.</td>
</tr>
<tr>
<td>STATRSCT (92)</td>
<td>The SYSOUT token pointed to by STATCTKN is not valid.</td>
</tr>
<tr>
<td>STATRSCE (96)</td>
<td>The SYSOUT owner in STATSCRE is not valid.</td>
</tr>
<tr>
<td>STATRSSD (100)</td>
<td>The SYSOUT destination in STASSDES is not valid.</td>
</tr>
<tr>
<td>STATRSCC (104)</td>
<td>The SYSOUT class in STASSCLA is not valid.</td>
</tr>
<tr>
<td>STATRSXW (108)</td>
<td>The SYSOUT external writer in STASSWTR is not valid.</td>
</tr>
<tr>
<td>STATRECJ (112)</td>
<td>STATJSB and STATSCTK are mutually exclusive.</td>
</tr>
<tr>
<td>STATRVM (116)</td>
<td>STATVRBO or STATOUTV requested with incorrect filters.</td>
</tr>
<tr>
<td>STATRBEA (120)</td>
<td>STATTRSA does not point to a valid STATJQ or STATSE.</td>
</tr>
<tr>
<td>STATRSFR (124)</td>
<td>STATSFOR is not valid.</td>
</tr>
</tbody>
</table>
### STATRE2 Contents
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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATVERO</td>
<td>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.</td>
</tr>
<tr>
<td>STATJOBF</td>
<td>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 <a href="#">Job Information Elements</a> 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 <a href="#">SYSOUT Information Elements</a> for a description of each SYSOUT level element.</td>
</tr>
<tr>
<td>STATNRJQ</td>
<td>The number of jobs that match the specified filter requirements.</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STJQEYE</td>
<td>Eycatcher ‘SJQE’.</td>
</tr>
<tr>
<td>STJQOHDR</td>
<td>Offset from the start of the STATJQ to the first job information data section.</td>
</tr>
<tr>
<td>STJQNEXT</td>
<td>Address of the next STATJQ area on the STATJOBF chain.</td>
</tr>
<tr>
<td>STJQSE</td>
<td>If SYSOUT data is requested, this is the head of the SYSOUT information elements (STATSE) for this job.</td>
</tr>
<tr>
<td>STJQOSS</td>
<td>Name of the subsystem that created this entry.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STHDLEN</td>
<td>Length of all sections which follow (including this section)</td>
</tr>
<tr>
<td>STHDTYPE</td>
<td>Section type identifier of STHD1HDR (0)</td>
</tr>
<tr>
<td>STHDMOD</td>
<td>Section type modifier of STHD1MOD (0)</td>
</tr>
<tr>
<td>STHDSIZE</td>
<td>Length of this section (4 bytes)</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STTRTYPE</td>
<td>Section type identifier of STTRTERS (1)</td>
</tr>
<tr>
<td>STTRMOD</td>
<td>Section type modifier of STTRTMOD (0)</td>
</tr>
<tr>
<td>STTRNAME</td>
<td>Job name</td>
</tr>
<tr>
<td>STTRJID</td>
<td>Job ID</td>
</tr>
</tbody>
</table>
**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. Whether or not the local node name appears in the destination depends on the setting of the STAT1LCL option bit.

**STTRXNOD**
Job’s execution node. Whether or not the local node name appears in the destination depends on the setting of the STAT1LCL option bit.

**STTRPRND**
The default print node for the job. Whether or not the local node name appears in the destination depends on the setting of the STAT1LCL option bit.

**STTRPRRE**
The default print remote or userid for the job. Whether or not the local node name appears in the destination depends on the setting of the STAT1LCL option bit.

**STTRPUND**
The default punch node for the job. Whether or not the local node name appears in the destination depends on the setting of the STAT1LCL option bit.

**STTRPURE**
The default punch remote for the job. Whether or not the local node name appears in the destination depends on the setting of the STAT1LCL option bit.

**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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRJNHL</td>
<td></td>
<td>Job is not held</td>
</tr>
<tr>
<td>STTRJHLD</td>
<td></td>
<td>Job is held</td>
</tr>
<tr>
<td>STTRJHLD</td>
<td></td>
<td>Job is held for duplicate job name</td>
</tr>
</tbody>
</table>

**STTRTYP**
Type of job

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRSTC</td>
<td></td>
<td>Started task</td>
</tr>
<tr>
<td>STTRTSU</td>
<td></td>
<td>Time sharing user</td>
</tr>
<tr>
<td>STTRJOB</td>
<td></td>
<td>Batch job</td>
</tr>
</tbody>
</table>
SSI Function Code 80

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRAPPCC</td>
<td>APPC initiator</td>
</tr>
<tr>
<td>STTRPRIO</td>
<td>Job’s priority</td>
</tr>
<tr>
<td>STTRARMS</td>
<td>Job’s automatic restart manager status</td>
</tr>
</tbody>
</table>

**Bit Value Description**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRARMR</td>
<td>Job is automatic restart manager registered</td>
<td></td>
</tr>
<tr>
<td>STTRARMW</td>
<td>Job is awaiting automatic restart manager restart</td>
<td></td>
</tr>
</tbody>
</table>

**Bit Value Description**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRMSPN</td>
<td>JESLOG for this job is spinable</td>
<td></td>
</tr>
<tr>
<td>STTRPEOM</td>
<td>Indicates job is being process for End of Memory</td>
<td></td>
</tr>
<tr>
<td>STTRJCLD</td>
<td>JESJCLIN dataset available</td>
<td></td>
</tr>
<tr>
<td>STTRSYSL</td>
<td>MVS SYSLOG job</td>
<td></td>
</tr>
</tbody>
</table>

**Field Name Description**

STTRMXIND
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.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STTRXAB</td>
<td>If this bit is on, STTRMXCC contains an ABEND code.</td>
</tr>
<tr>
<td>STTRXCDE</td>
<td>If this bit is on, STTRMXCC contains a completion code.</td>
</tr>
<tr>
<td>STTRXUNK</td>
<td>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.</td>
</tr>
<tr>
<td>STTRXNRM</td>
<td>Job ended normally</td>
</tr>
<tr>
<td>STTRXCC</td>
<td>Job ended by completion code</td>
</tr>
<tr>
<td>STTRXJCL</td>
<td>Job had a JCL error</td>
</tr>
<tr>
<td>STTRXCAN</td>
<td>Job was canceled</td>
</tr>
<tr>
<td>STTRXABN</td>
<td>Job ABENDed</td>
</tr>
<tr>
<td>STTRXCAB</td>
<td>Converter ABENDed while processing the job</td>
</tr>
<tr>
<td>STTRXSEC</td>
<td>Job failed security checks</td>
</tr>
<tr>
<td>STTRXEOEM</td>
<td>Job failed in end-of-memory</td>
</tr>
</tbody>
</table>

**Field Name Description**

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STJ2LEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STJ2TYPE</td>
<td>Section type identifier of STJ2TERS (2)</td>
</tr>
<tr>
<td>STJ2MOD</td>
<td>Section type modifier of STJ2TMOD (0)</td>
</tr>
<tr>
<td>STJ2FLG1</td>
<td>General flag byte</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STJ21PRO</td>
<td>Job is protected</td>
</tr>
<tr>
<td>STJ21IND</td>
<td>Job is set to independent mode</td>
</tr>
<tr>
<td>STJ21SYS</td>
<td>Job represents a system data set</td>
</tr>
<tr>
<td>STJ21CNW</td>
<td>Job can only be processed by a converter that can wait for OS resources</td>
</tr>
<tr>
<td>STJ21RBL</td>
<td>Job is on the JES2 rebuild queue</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STJ2JKEY</td>
<td>The JES2 job key for the JOB</td>
</tr>
<tr>
<td>STJ2SPOL</td>
<td>The SPOOL token associated with the job</td>
</tr>
<tr>
<td>STJ2SPAC</td>
<td>Number of track groups of SPOOL space used by the job (a value of -1 indicates that the count is not available).</td>
</tr>
<tr>
<td>STJ2DPNO</td>
<td>Binary default print node</td>
</tr>
<tr>
<td>STJ2DPRM</td>
<td>Binary default print remote</td>
</tr>
<tr>
<td>STJ2DPUS</td>
<td>Default print userid</td>
</tr>
<tr>
<td>STJ2INPN</td>
<td>Binary input node</td>
</tr>
<tr>
<td>STJ2XEQN</td>
<td>Binary execution node (if job has completed execution).</td>
</tr>
<tr>
<td>STJ2JQEI</td>
<td>Index of JQE</td>
</tr>
<tr>
<td>STJ2OFSL</td>
<td>Offload status mask</td>
</tr>
</tbody>
</table>
SSI Function Code 80

**STJ2BUSY** Binary busy byte

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAFLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STAFTYPE</td>
<td>Section type identifier of STAFFIN (3)</td>
</tr>
<tr>
<td>STAFMOD</td>
<td>Section type modifier of STAFTMOD (0)</td>
</tr>
<tr>
<td>STAFNUM</td>
<td>Number of members for which the job has affinity</td>
</tr>
<tr>
<td>STAFMEMB</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSCLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STSCTYPE</td>
<td>Section type identifier of STSCHED ('04'x)</td>
</tr>
<tr>
<td>STSCMOD</td>
<td>Section type modifier of STSCTMOD ('00'x)</td>
</tr>
<tr>
<td>STSCAHLD</td>
<td>Reasons why the job will not run</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSCJCLS</td>
<td>Job class is held</td>
</tr>
<tr>
<td>STSCJCLM</td>
<td>Job class limit has been reached</td>
</tr>
<tr>
<td>STSCJSCH</td>
<td>Scheduling environment is not available</td>
</tr>
<tr>
<td>STSCJAFF</td>
<td>Systems for which the job has affinity are not available</td>
</tr>
<tr>
<td>STSCJSPL</td>
<td>Spool volumes needed by the job are not available</td>
</tr>
<tr>
<td>STSCJBSY</td>
<td>Job is busy on a device</td>
</tr>
<tr>
<td>STSCJSCF</td>
<td>The RACF SECLABEL by system option is in effect. The SECLABEL associated with the job (STTRSECL) is not available on any active system</td>
</tr>
<tr>
<td>STSCNOSY</td>
<td>No system(s) with the correct combination of resources is available</td>
</tr>
</tbody>
</table>

**STSCFLG1** General flag byte

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
</table>
STSC1JCM  Mode of the JOBCLASS. Off is JES mode, on is WLM mode.

STCASID  ASID where job is executing (zero if not active).

STCSRVC  Service class associated with the job

STCSESTT 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, STCSESTT will be set to negative 1 (−1).

STCSENV  Scheduling environment required by the job.

STCQPOS  Position of this job on a WLM service class queue (if STATSPOS is on)

STCQNUM  Number of jobs on this WLM service class queue (if STATSPOS is on)

STCQACT  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 STCSESTT. If STCSESTT 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 STCSESTT. If STCSESTT 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 STSCHEXD (‘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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSSLEN</td>
<td>Length of this section.</td>
</tr>
<tr>
<td>STSSTTYPE</td>
<td>Section type identifier of STSCHEXD (‘04’x)</td>
</tr>
<tr>
<td>STSSMOD</td>
<td>Section type modifier of STSSTMOD (‘01’x)</td>
</tr>
<tr>
<td>STSSNUM</td>
<td>Number of systems that have the required scheduling environment.</td>
</tr>
<tr>
<td>STSSSYS</td>
<td>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.</td>
</tr>
</tbody>
</table>
Job Queue Element SECLABEL Availability Section: This section is mapped by the STATSCLF DSECT and is identified by a type of STSECLAF ('05'x) and a modifier of STSLTMOD ('00'x). This section is present if the the SECLABEL by system RACF option is enabled and the job is queued for conversion processing or execution. This section lists the MVS system names where the SECLABEL associated with the job (STTRSECL) is active (available). This section is not returned by JES3 subsystems.

The fields in the STATSCLF section are:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSLLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STSLTYPE</td>
<td>Section type identifier of STSECLAF ('05'x)</td>
</tr>
<tr>
<td>STSLMOD</td>
<td>Section type modifier of STSLTMOD ('00'x)</td>
</tr>
<tr>
<td>STSLNUM</td>
<td>Number of systems where the SECLABEL is active</td>
</tr>
<tr>
<td>STSLSYS</td>
<td>Name of first system where the SECLABEL is active. Other system names follow after this system name. The number of system names present is in field STSLNUM</td>
</tr>
</tbody>
</table>

Job Queue Element JES3 Terse Section: This section is mapped by the STATJ3TR DSECT and is identified by a type of STJ3TERS and a modifier of STJ3TMOD (X'0'). This section is present only if the job is owned by a JES3 subsystem.

The fields in the STATJ3TR section are:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STJ3LEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STJ3TYPE</td>
<td>Section type identifier of STJ3TERS</td>
</tr>
<tr>
<td>STJ3MOD</td>
<td>Section type modifier of STJ3TMOD (X'0')</td>
</tr>
<tr>
<td>STJ3SPOL</td>
<td>Spool data token or zero</td>
</tr>
<tr>
<td>STJ3JSTT</td>
<td>List of reasons, by system, why job is waiting to run (RQJSTAT)</td>
</tr>
<tr>
<td>STJ3JSTM</td>
<td>List of system names corresponding to STJ3JSTT, terminated by zero</td>
</tr>
</tbody>
</table>

Job Queue Element Verbose Prefix: This section is mapped by the STATVE DSECT.

The fields in the STATVE prefix are:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STVEEYE</td>
<td>Eye catcher C’SJVE’</td>
</tr>
<tr>
<td>STVEOHDR</td>
<td>Offset from the start of the STATVE to the first information section.</td>
</tr>
<tr>
<td>STVEJOB</td>
<td>Address of the associated job queue data element.</td>
</tr>
<tr>
<td>STVESIZE</td>
<td>Size of the prefix.</td>
</tr>
</tbody>
</table>

Job Verbose Element 1st Header Section: This section is mapped by the STATJVHD DSECT and is identified by a type of STJV1HDR and a modifier of STJV1MOD (X'0').
### Job Queue Element Verbose Section

This section is mapped by the STATJQVB DSECT and is identified by a type of STVBVRBO (X'') and a modifier of STVBVMOD (X'0'). Data in this section requires disk I/O and aSTATVER=STATV040.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STVBLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STVBTYPE</td>
<td>Section type identifier of STVBVRBO</td>
</tr>
<tr>
<td>STVBMOD</td>
<td>Section type modifier of STVBVMOD (X'0')</td>
</tr>
<tr>
<td>STVBFLG1</td>
<td>Section flag byte</td>
</tr>
</tbody>
</table>

#### Bit value | Description
---|---
STBB1ERR | Error returning verbose data (terse data section returned)

### Field name | Description
---|---
STVBJCPY | Job copy count

This value is meaningful in JES2 only. JES3 always returns 1.

STVBLNCT | Job line count

This value is meaningful in JES2 only. JES3 always returns 0.

STVBIDEV | Input device name

STVBISID | Input system/member

**Note:** In JES3, for a TSO or INTRDR submission job, STVBISID is set to the system name on which the submitting user or job is active. For all other submissions, STVBISID is set to the JES3 global. Also, for a TSO or INTRDR submission from a user or job that was active on a system running a JES3 release lower than z/OS V1R7, STVBISID is set to the JES3 global.

STVBJCIN | Job input count

STVBJLIN | Job line count

STVBJPAG | Job page count

STVBJPUN | Job card (output) count

STVBRTS | Input start time/date

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STVBRSTS</td>
<td>Input start time. This is in hundredths of seconds since midnight.</td>
</tr>
<tr>
<td>STVBRTSID</td>
<td>Input start date. This is in the form 0cyydddF.</td>
</tr>
</tbody>
</table>

STVBRTE | Input end time/date

### Field name | Description
STVBRTET  Input end time. This is in hundredths of seconds since midnight.

STVBRTED  Input end date. This is in the form 0cyydddF.

STVBSYS  Execution MVS system name

STVBMBR  Execution JES2 member name

STVBXTS  Execution start time/date

Field name  Description

STVBXTST  Execution start time. This is in hundredths of seconds since midnight.

STVBXTSD  Execution start date. This is in the form 0cyydddF.

STVBXTE  Execution end time/date

Field name  Description

STVBXTET  Execution end time. This is in hundredths of seconds since midnight.

STVBXTED  Execution end date. This is in the form 0cyydddF.

STVBJUSR  JMRUSEID field

STVBMCLS  Message class (Job card)

STVBNOTN  Notify Node

STVBNOTU  Notify Userid

STVPBNAM  Programmer’s name (from Job card)

STVBACCT  Account number (from Job card)

STVBDEPT  NJE department

STVBBLDG  NJE building

STVBROOM  Job card room number

In JES3, the values STVBACCT, STVBDEPT, STVBBLDG, and STVBROOM are filled in from the ACCT=, DEPT=, BLDG=, and ROOM= parameters on the /*NETACCT statement.

STVBJVDT  JDVT name for job

STVBSUBU  Submitting userid

STVBSUBG  Submitter’s security group name. In JES3, this field contains the owner’s security group name.

STVBMLRC  The maximum LRECL of the JCLIN stream

STVBSIZE  Size of verbose information

Job Queue Element Security Section (mapped by SAF token): This section is mapped by the STATJQSE DSECT and is identified by a type of STSESEC and a modifier of STSESMOD(X'0').

Field name  Description

STSELEN  Length of this section

STSETYPE  Section type identifier of STSESEC

STSEMOD  Section type modifier of STSESMOD (X'0')
STSEFLG1   Security Section flag byte

Bit value   Description
STSE1ERR    Error obtaining verbose data (terse data returned)
STSE1JB     Token represents a job

STSEOFFS   Offset to SAF token
STSETOKN   Mapped SAF token

Job Queue Element Accounting Section: This section is mapped by the
STATJQAC DSECT and is identified by a type of STACACCT and a modifier of
STACAMOD (X'0').

Note: If the job does not have job accounting string then in then in then:
* In the case of JES2, the Job Information Element section will be truncated
  after the Job Queue Element Security section. There will be no Job
  Queue Element Accounting section.
* In the case of JES3, the Job Queue Element Accounting section will have
  the accounting length of 145, number of sub string one and substring will
  contain all zeroes.

Field name   Description
STACLEN     Length of this section
STACTYPE    Section type identifier of STACACCT
STACMOD     Section type modifier of STACAMOD (X'0')
STACFLG1    Security Section flag byte

Bit value   Description
STAC1ERR    Error obtaining verbose data (terse data returned)
STAC1OVJB   Accounting string can be overlaid by other than
            originating node

STACOFFS   Offset to beginning of accounting information
STACFLEN   Length of fixed portion

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSEEYE</td>
<td>Eyecatcher C’SOUT’</td>
</tr>
<tr>
<td>STSEOHDR</td>
<td>Offset from the start of the STATSE to the first SYSOUT information data section.</td>
</tr>
<tr>
<td>STSEJNXT</td>
<td>Address of the next STATSE area for this job.</td>
</tr>
<tr>
<td>STSEJOB</td>
<td>Address of the STATJQ for the job that owns this SYSOUT.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSHLEN</td>
<td>Length of all sections which follow (including this section)</td>
</tr>
<tr>
<td>STSHTYPE</td>
<td>Section type identifier of STSH1HDR (‘40’x)</td>
</tr>
<tr>
<td>STSHMOD</td>
<td>Section type modifier of STSH1MOD (‘00’x)</td>
</tr>
<tr>
<td>STSHSIZE</td>
<td>Length of this section (4 bytes)</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STS2LEN</td>
<td>Length of this section.</td>
</tr>
<tr>
<td>STS2TYPE</td>
<td>Section type identifier of STS2TERS (‘42’x)</td>
</tr>
<tr>
<td>STS2MOD</td>
<td>Section type modifier of STS2TMOD (‘00’x)</td>
</tr>
<tr>
<td>STS2FLG1</td>
<td>General flag byte</td>
</tr>
</tbody>
</table>

**Bit Value** | **Description**                                                                 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>STS21DSH</td>
<td>JOE representing this SYSOUT data set has been cloned</td>
</tr>
<tr>
<td>STS21TSO</td>
<td>JOE is available for TSO OUTPUT processing</td>
</tr>
<tr>
<td>STS21USR</td>
<td>SYSOUT element is on the userid queue</td>
</tr>
</tbody>
</table>
STS2OGNM    JOE output group name
STS2CRTM    JOE create time (STCK format system clock time)
STS2RNOD    Binary destination node
STS2RRMT    Binary destination remote number
STS2RUSR    Destination user route code
STS2TSWB    JOE level SWB MTTR (8 byte field)
STS2CKPT    Checkpoint MTTR (8 byte) if checkpoint is valid (else zero)
STS2JOEI    Index of JOE
STS2OFSL    SPOOL offload selection mask
STS2BUSY    Binary busy byte

**SYSOUT Element JES3 Terse Section:** This section is mapped by the STATSJ3T DSECT. This section is meaningful only if the job is owned by a JES3 subsystem.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STS3LEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STS3TYPE</td>
<td>Section type identifier of</td>
</tr>
<tr>
<td>STS3MOD</td>
<td>Section type modifier of</td>
</tr>
<tr>
<td>STS3FLG1</td>
<td>flag byte</td>
</tr>
<tr>
<td></td>
<td><strong>Bit value</strong></td>
</tr>
<tr>
<td>STS31XSY</td>
<td>Extended keywords used</td>
</tr>
<tr>
<td>STS3SIZE</td>
<td>Length of section</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSTLEN</td>
<td>Length of this section.</td>
</tr>
<tr>
<td>STSTTYPE</td>
<td>Section type identifier of STSTTERS (‘41’x)</td>
</tr>
<tr>
<td>STSTMOD</td>
<td>Section type modifier of STSTTMOD (‘00’x)</td>
</tr>
<tr>
<td>STSTOUID</td>
<td>Userid that owns the SYSOUT</td>
</tr>
<tr>
<td>STSTSECL</td>
<td>SECLABEL assigned to the SYSOUT</td>
</tr>
<tr>
<td>STSTDEST</td>
<td>Destination of SYSOUT. Whether or not the local node name appears in the destination depends on the setting of the STAT1LCL option bit.</td>
</tr>
<tr>
<td>STSTCLAS</td>
<td>Class assigned to the SYSOUT</td>
</tr>
<tr>
<td>STSTNREC</td>
<td>Number of records in the SYSOUT element</td>
</tr>
<tr>
<td>STSTPAGE</td>
<td>Number of pages in the SYSOUT element</td>
</tr>
<tr>
<td>STSTLNCT</td>
<td>Number of lines in the SYSOUT element (JES3 only)</td>
</tr>
</tbody>
</table>
SSI Function Code 80

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSTBYCT</td>
<td>Number of bytes in the SYSOUT element (JES3 only)</td>
</tr>
<tr>
<td>STSTFORM</td>
<td>Form assigned to the SYSOUT</td>
</tr>
<tr>
<td>STSTFCB</td>
<td>Forms control buffer (FCB)</td>
</tr>
<tr>
<td>STSTUCS</td>
<td>Universal character set (UCS)</td>
</tr>
<tr>
<td>STSTXWTR</td>
<td>External writer name</td>
</tr>
<tr>
<td>STSTPMDE</td>
<td>Processing mode (PRMODE)</td>
</tr>
<tr>
<td>STSTFLSH</td>
<td>Flash</td>
</tr>
<tr>
<td>STSTCHAR</td>
<td>Character sets assigned to the SYSOUT (JES3 only)</td>
</tr>
<tr>
<td>STSTMODF</td>
<td>MODIFY=(modname) value (JES3 only)</td>
</tr>
<tr>
<td>STSTMODC</td>
<td>MODIFY=(,trc) value (JES3 only)</td>
</tr>
<tr>
<td>STSTFLG2</td>
<td>General flag byte</td>
</tr>
<tr>
<td>STSTSYS</td>
<td>MVS system name where output currently being processed (blank if not currently active)</td>
</tr>
<tr>
<td>STSTMEM</td>
<td>JES member name where output currently being processed (blank if not currently active)</td>
</tr>
<tr>
<td>STSTDEVN</td>
<td>Device name on which output currently being processed (blank if not currently active)</td>
</tr>
<tr>
<td>STSTHSTA</td>
<td>Current hold status of the SYSOUT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STST2CIV</td>
<td></td>
<td>The token in STSTCTKN cannot be used. It is not valid.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSTDHLD</td>
<td>OUTDISP=HOLD</td>
</tr>
<tr>
<td>STSTDLVE</td>
<td>OUTDISP=LEAVE</td>
</tr>
<tr>
<td>STSTDWRT</td>
<td>OUTDISP=WRITE</td>
</tr>
<tr>
<td>STSTDKEP</td>
<td>OUTDISP=KEEP</td>
</tr>
</tbody>
</table>
**STSTFLG1**

General flag byte

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STST1BRT</td>
<td>BURST=YES requested</td>
</tr>
<tr>
<td>STST1DSI</td>
<td>3540 held SYSOUT element</td>
</tr>
<tr>
<td>STST1IPA</td>
<td>SYSOUT destination includes an IP address</td>
</tr>
<tr>
<td>STST1CPD</td>
<td>SYSOUT element includes page mode data</td>
</tr>
<tr>
<td>STST1SPN</td>
<td>SYSOUT element was spun</td>
</tr>
<tr>
<td>STST1NSL</td>
<td>SYSOUT not selectable</td>
</tr>
<tr>
<td>STST1APC</td>
<td>SYSOUT has job level information (has a STOTAPPC type section)</td>
</tr>
<tr>
<td>STST1CTK</td>
<td>When SYSOUT was allocated, the DALRTCTK key was specified (client token returned)</td>
</tr>
</tbody>
</table>

**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.

**Note:** Ensure that the token is valid by verifying that the STST2CIV bit is not on.

**Using STSTCTKN**

You may receive multiple tokens for a set of data sets meeting your status selection criteria. This is based on how the JES groups data sets into schedulable elements and may be different for each JES.

For example, if your status request specifies FORMS as the only selection criterion, you may still receive multiple tokens for a single job because other characteristics may vary or because of the way JES decided to group the data sets under a single schedulable element.

The Extended Status token will return the same group of data sets on a subsequent SAPI call unless:

- The JES was restarted
- Some of the output was modified such that a new schedulable element was created in place of an existing one
- The schedulable element was either deleted by the operator or it was processed by another application or writer

Therefore, it is possible that you will receive SSS2EODS for what otherwise would be a valid token request. To make sure there are no data sets left in JES that meet your selection criteria, you should repeat a status request, examine the results, and issue another SAPI request until you get an output group for a different job. You can then continue with that job or issue a PUT for the received group with the KEEP disposition to return it back to the queue for some other output function to process.
**SSI Function Code 80**

**STS1APC**: SYSOUT data set has APPC JOB information associated with it

**SYSOUT Verbose Element Prefix:**

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STVOEYE</td>
<td>Eye catcher (C'SSVE')</td>
</tr>
<tr>
<td>STVOOHDR</td>
<td>Offset to first section</td>
</tr>
<tr>
<td>STVOJOB</td>
<td>Address of associated job queue data element - STATJQ</td>
</tr>
<tr>
<td>STVOJNXT</td>
<td>Address of next verbose SYSOUT element for JOB</td>
</tr>
<tr>
<td>STVOSOUT</td>
<td>Address of associated SYSOUT data element - STATSE</td>
</tr>
<tr>
<td>STVOSNXT</td>
<td>Address of next verbose SYSOUT element for STATSE</td>
</tr>
<tr>
<td>STVOSIZE</td>
<td>Size of prefix</td>
</tr>
</tbody>
</table>

**SYSOUT Verbose Element 1st Header Section**: This section is mapped by the STATSVHD DSECT and is identified by a type of STSV1HDR and a modifier of STSV1MOD (X'0').

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSVLEN</td>
<td>Length of entire SYSOUT verbose element (Maximum value is 65535)</td>
</tr>
<tr>
<td>STSVYPE</td>
<td>Section type identifier of STSV1HDR</td>
</tr>
<tr>
<td>STSVMOD</td>
<td>Section type modifier of STSV1MOD (X'0')</td>
</tr>
<tr>
<td>STSVSIZE</td>
<td>Size of 1st Header Section</td>
</tr>
</tbody>
</table>

**SYSOUT Element Verbose Section**: This section is mapped by the STATSEVB DSECT and is identified by a type of STVSVRBO and a modifier of STVSVMOD (X'0').

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STVSLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STVSYPE</td>
<td>Section type identifier of STVSVRBO</td>
</tr>
<tr>
<td>STVSMOD</td>
<td>Section type modifier of STVSVMOD (X'0')</td>
</tr>
<tr>
<td>STVSFLG1</td>
<td>Section flag byte</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STVS1ERR</td>
<td>Error obtaining verbose data (terse section returned).</td>
</tr>
<tr>
<td>STVSDSCL</td>
<td>Line count, page count, byte count, and record count (STVSLMCT, STVSPGCT, STVSBUCT, and STVSRCCCT) are accurate. This bit will not be on if there was an abnormal termination or the data was created on a different node.</td>
</tr>
<tr>
<td>STVS1SPN</td>
<td>SPIN data set</td>
</tr>
<tr>
<td>STVS1JSL</td>
<td>Spun JESLOG data set</td>
</tr>
<tr>
<td>STVS1SYS</td>
<td>System data set</td>
</tr>
<tr>
<td>STVS1SIN</td>
<td>Instream data set (SYSIN)</td>
</tr>
</tbody>
</table>
STVS1DUM  Dummy data set (SYSOUT data set which will not print)

STVSRECF  Record format
STVSPRCD  Procname for the step creating this data set
STVSSTPD  Stepname for the step creating this data set
STVSDDND  DDNAME for the data set creation
STVSTJN   APPC Transaction Program Jobname that created this data set. This field has been deprecated. Applications should use STOTJOBN in the STATSEOT section.
STVSTJID   APPC Transaction Program Job ID that created this data set. This field has been deprecated. Applications should use STOTJID in the STATSEOT section.
STVSTOD   Date and time of data set availability in TOD format (that is, this value is the high-order word of the TOD clock obtained with a STCK instruction)
STVSSEGM   Segment ID (zero if data set not segmented)
STVSDSKY   Data set number (key)
STVSMLRL   Maximum logical record length (LRECL)
STVSLNCT   Line count (valid only if STVSDSCL is ON in STVSRET1)
STVVPGCT   Page count (valid only if STVSDSCL is ON in STVSRET1)
STVSBYCT   Byte count after blank truncation, 63 bit right justified (valid only if STVSDSCL is ON in STVSRET1)
STVVRCCCT   Record count (JES3 only) (valid only if STVSDSCL is ON in STVSRET1)
STVSDDSN   SYSOUT data set name (valid only if STVSDSCL is ON in STVSRET1)
STVSCOPY   Data set copy count
STVSFLSC   Number of flash copies
STVSCTKN   SYSOUT data set token
STVSCHAR   Printer translate table
STVSMODF   MODIFY=(modname)
STVSMODC   MODIFY=(,trc)
STVSIZE    Length of section

**SYSOUT Element JES2 Verbose Section:** This section is mapped by the STATSEO2 DSECT and is identified by a type of STO2VRBO and a modifier of STO2TMOD. This section contains general information that is meaningful only if the job is owned by a JES2 subsystem.

<table>
<thead>
<tr>
<th><strong>Field name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>STO2LEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STO2TYPE</td>
<td>Section type identifier of STO2VRBO</td>
</tr>
<tr>
<td>STO2MOD</td>
<td>Section type modifier of STO2TMOD (X'0')</td>
</tr>
</tbody>
</table>
SSI Function Code 80

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STO2FLG1</td>
<td>General flags</td>
</tr>
<tr>
<td>Bit value</td>
<td>Description</td>
</tr>
<tr>
<td>STO21ERR</td>
<td>Error obtaining verbose data</td>
</tr>
<tr>
<td>STO2SPST</td>
<td>Data set SPOOL data token</td>
</tr>
<tr>
<td>STO2SIZE</td>
<td>Length of section</td>
</tr>
</tbody>
</table>

**SYSOUT Element JES3 Verbose Section:** This section is mapped by the STATSE03 DSECT and is identified by a type of STO3VRBO and a modifier of STO3TMOD. This section contains general information that is meaningful only if the job is owned by a JES3 subsystem.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STO3LEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STO3TYPE</td>
<td>Section type identifier of STO3VRBO</td>
</tr>
<tr>
<td>STO3MOD</td>
<td>Section type modifier of STO3TMOD (X'0')</td>
</tr>
<tr>
<td>STO3FLG1</td>
<td>General flags</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit value</td>
<td>Description</td>
</tr>
<tr>
<td>STO31ERR</td>
<td>Error obtaining verbose data</td>
</tr>
<tr>
<td>STO3CMTK</td>
<td>Modify token which can be included on a *MODIFY,U operator command to uniquely identify the data set</td>
</tr>
<tr>
<td>STO3SIZE</td>
<td>Length of section</td>
</tr>
</tbody>
</table>

**SYSOUT Element Security Section (mapped by SAF token):** This section is mapped by the STATSE03 DSECT and is identified by a type of STSOSEC and a modifier of STSOSMOD (X'0').

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STSOLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>STSOTYPE</td>
<td>Section type identifier of STSOSEC</td>
</tr>
<tr>
<td>STSOMOD</td>
<td>Section type modifier of STSOSMOD (X'0')</td>
</tr>
<tr>
<td>STSOFLG1</td>
<td>Flag byte</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit value</td>
<td>Description</td>
</tr>
<tr>
<td>STSO1ERR</td>
<td>Error obtaining verbose data</td>
</tr>
<tr>
<td>STSOOFFS</td>
<td>Offset to SAF token</td>
</tr>
<tr>
<td>STSOTOKN</td>
<td>Mapped SAF token</td>
</tr>
</tbody>
</table>

**SYSOUT APPC transaction output section, (mapped by STATEOT DSECT):**
This section is mapped by the STATSEOT DSECT and is identified by a type of STOTAPPC and a modifier of STOTSMOD.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOTJOBN</td>
<td>APPC transaction program job name that created this data set</td>
</tr>
<tr>
<td>STOTJID</td>
<td>APPC transaction program job id that created this data set</td>
</tr>
</tbody>
</table>
JES3 Unsupported Flags and Fields

Table 7 summarizes which flags and fields are not supported by JES3.

<table>
<thead>
<tr>
<th>Flagname</th>
<th>Fieldname</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATSOJI</td>
<td>STATOJBI</td>
<td>Original job ID</td>
</tr>
<tr>
<td>STATSVOL</td>
<td>STATVOL</td>
<td>List of SPOOL volume serial numbers</td>
</tr>
<tr>
<td>STATSMEM</td>
<td>STATMEMB</td>
<td>JES member name where job is active</td>
</tr>
<tr>
<td>STATSORG</td>
<td>STATORGN</td>
<td>Origin node name for selection</td>
</tr>
<tr>
<td>STATSXEQ</td>
<td>STSTXEQN</td>
<td>Execution node name for selection</td>
</tr>
</tbody>
</table>

**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 SS0B                                *
***********************************************************************
```
SSI Function Code 80

USING SSOB, STSSOB
LA R0, STSSOB
LA R1, 'STSSOB'
SLR R15, R15
MVCL R0, R14

MVC SSOBID = 'SSOB'
MVC SSOBLEN = Y(SSOBH8IZ)
MVC SSOBFUNC = Y(SSOBE1TA)
MVC SSOBSSIB = 'F' O'
LA R0, SSOB + SSOBH8IZ
ST R0, SSOBINDV

USING STAT, SSOB + SSOBH8IZ

MVC STATEYE = 'STAT'
MVC STATLEN = Y(STATSIZE)
MVC STATVER = AL1(STATCVRL, STATCVRM)
MV1 STATTYPE, STATTERS

***********************************************************************
* Make only filter this userid *
***********************************************************************

OI STATSEL1, STATSOWN
LA R0, STATOWNR
LA R1, 'STATOWNR'

LA R14, THISUSER
LA R15, 'THISUSER'
IMC R15, B'1000', 'C'

MVCL R0, R14

***********************************************************************
* Call the subsystem *
***********************************************************************

MODESET MODE=SUP

LA R1, STSSOB
O R1, X'80000000'
ST R1, PARMPTTR
LA R1, PARMPTTR

IFEFSREQ

LTR R15, R15
BNZ SSREQERX

MODESET MODE=PROB

***********************************************************************
* Process results for IEFSSREQ here *
***********************************************************************

USING STATJQ, R4

LA R4, STATJOBF -(STJQNEXT - STATJQ)
LOOPSTJQ

ICM R4, B'1111', STJQNEXT

BZ DONESTJQ

LA R3, STJQOHDR
LA R5, STATJQ(R3)
SLR R2, R2

ICM R2, B'0011', STHDLEN - STATJQHD(R5)

LA R5, STHDSIZE(R5)
SL R2, A(STHDSIZE)

LOOPSECT

CLC 2(2, R5), AL1(STTRTERS, STTRTMOD)
BNE NOTTERSE
Chapter 3. SSI Function Codes Your Program Can Request 215
SSI Function Code 80

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
JES Properties—SSI Function Code 82

The JES property information services (SSI function code 82) allow a user-supplied program to obtain information about JES managed structures such as NJE nodes, SPOOL volumes, initiators, members in the JESPLEX, and job classes.

JES Property Information Services Request Types

Table 8. JES Properties Request Types

<table>
<thead>
<tr>
<th>Request Type</th>
<th>Function (SSJPFREQ)</th>
<th>Request Data Area Pointer (SSJPUSER)</th>
</tr>
</thead>
<tbody>
<tr>
<td>“NJE Node Information”</td>
<td>SSJPNJOD/SSJPNJRS</td>
<td>IAZJPNJN</td>
</tr>
<tr>
<td>“SPOOL Volume Information”</td>
<td>SSJPSPOD/SSJPSPRS</td>
<td>IAZJPSPL</td>
</tr>
<tr>
<td>“Initiator Information” on page 247</td>
<td>SSJPITOD/SSJPITRS</td>
<td>IAZJPITD</td>
</tr>
<tr>
<td>“JESplex Information” on page 262</td>
<td>SSJPJXOD/SSJPJXRS</td>
<td>IAZJPLEX</td>
</tr>
<tr>
<td>“Job Class Information” on page 274</td>
<td>SSJPJCOD/SSJPJCRS</td>
<td>IAZJPCLS</td>
</tr>
</tbody>
</table>

NJE Node Information

The NJE Node Information service provides information about JES Network Job Entry (NJE) nodes. Information can be obtained on all NJE nodes or filters can be supplied to limit which nodes are returned. Information is returned as a chained list of data areas and each data area represents an NJE node.

See the following sections for more information about NJE Node Information:

- “Type of Request” on page 218
- “Use Information” on page 218
- “Issued to” on page 218
- “Related SSI Codes” on page 218
- “Related Concepts” on page 218
- “Environment” on page 218
- “Input Register Information” on page 219
- “Input Parameters” on page 219
SSI Function Code 82

- “Output Register Information” on page 224
- “Return Code Information” on page 224
- “Output Parameters” on page 225

**Type of Request:** Directed SSI Call.

**Use Information:** To use the JES property information services SSI, callers must first decide the function they want to perform. The appropriate parameter list must be obtained and pointed to by SSJUSER.

**Issued to:** A JES subsystem (either primary or secondary). The subsystem does not have to be associated with the requesting address space.

**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 that are commonly referenced are mapped by the following mapping macros:
- IEFSSOBH
- IEFJSSIB
- IAZSSJP
- IAZJPNJN (NJE Node Information)

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, IAZSSJP, and IAZJPNJN, control blocks can reside in 24- or 31-bit virtual storage
**Recovery** The caller should provide an ESTAE-type recovery environment. See z/OS MVS Programming: Assembler Services Guide for more information about an ESTAE-type recovery environment.

Figure 23 on page 219 shows the environment at the time of the call for SSI function code 82, NJE Node Information Subfunction.
**Input Register Information:** Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

**Input Parameters:** Input parameters for the function routine are:
- SSOB
- SSIB
- IAZSSJP
- IAZJPNJN (NJE Node Information)

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 82(SSOBSSJP)</td>
</tr>
</tbody>
</table>

---

**Figure 23. Environment at Time of Call for SSI Function Code 82, NJE Node Information Subfunction**

---

![Diagram of the environment at time of call for SSI Function Code 82, NJE Node Information Subfunction](image-url)
### SSI Function Code 82

**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 about the life-of-job SSIB.

**SSOBINDV**: Address of the function-dependent area (IAZSSJP control block)

Set all other fields in the SSIB 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier ‘SSIB’</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name: name of the subsystem to which this NJE Node Information Services request is directed</td>
</tr>
</tbody>
</table>

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

**IAZSSJP Contents**: The caller must set the following fields in the IAZSSJP control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPID</td>
<td>Eyecatcher for the control block (set to ‘SSJP’)</td>
</tr>
<tr>
<td>SSJPLEN</td>
<td>Length of the IAZSSJP (SSJPSIZE) control block</td>
</tr>
<tr>
<td>SSJPVER</td>
<td>Input version of the IAZSSJP control block. Set to SSJPVER1 for version 1 of the control block or to SSJPVERC for the current version of the control block.</td>
</tr>
<tr>
<td>SSJPfreq</td>
<td>Function to be performed on this request. Valid functions and the related SSJUSER area are:</td>
</tr>
<tr>
<td></td>
<td><strong>Field Value</strong></td>
</tr>
<tr>
<td>SSJPNJOD</td>
<td>IAZJPNJN NJE Node Information service, obtain data</td>
</tr>
<tr>
<td>SSJPNJRS</td>
<td>IAZJPNJN NJE Node Information service, release storage</td>
</tr>
<tr>
<td>SSJPUSER</td>
<td>Pointer to service specific data area ’(IAZJPNJN)’</td>
</tr>
</tbody>
</table>

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

**NJE Node Information service, IAZJPNJN contents**: For the NJE Node Information service (function code SSJPNJOD), the caller must set the following fields in the IAZJPNJN control block:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNLEYE</td>
<td>Eyecatcher of the control block (set to ‘SSJPNJNL’).</td>
</tr>
<tr>
<td>NJNLLNG</td>
<td>Length of the IAZJPNJN (NJNLSIZE) control block.</td>
</tr>
<tr>
<td>NJNLVRM</td>
<td>Input version of the IAZJPNJN control block. Set to</td>
</tr>
</tbody>
</table>
NJNLOPT1  Processing options:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNLLODMC</td>
<td>Perform security label dominance check. This check is always performed for non-authorized callers (JES2 Only).</td>
</tr>
</tbody>
</table>

NJNLSTRP  Storage management anchor for use by the subsystem that responds to this request. It is expected that the caller sets this field to zero the first time IAZJPNJN is used and from that point on the field is managed by the subsystem.

The caller can also set the following fields in the IAZJPNJN control block on input to limit (or select) which data is returned. If no filters are specified, all data is returned. If any filters are specified, at least one of the filter conditions in each of the separate filters must be matched before data is returned.

An implicit OR is performed between filters that apply to the same node attribute. For example, if both NJNL1SSG and NJNL1CSG are selected, SSI returns NJE nodes that are defined with compatible signon in addition to NJE nodes that are defined with a secure signon.

An implicit AND is performed between filters that apply to the different node attributes. For example, if both NJNL1NAM and NJNL1SNA filters are selected, SSI returns NJE nodes with the names that match NJNLNOD1 field and which are at the same time connected through SNA protocol.

If a filter is not recognized or does not apply, it does not have an impact on the result of the SSI call. For example, JES3-only filters do not have impact on SSI output from JES2.

Field Name  Description

NJNLFLT1  Filter by node attributes:

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNL1NAM</td>
<td>Select by the node name specified in NJNLNOD1</td>
</tr>
<tr>
<td>NJNL1RNG</td>
<td>Select by a range of node numbers specified in NJNLRNGL and NJNLRNGH (JES2 Only)</td>
</tr>
</tbody>
</table>
SSI Function Code 82

- **NJNL1SSG**: Select nodes with a secure signon.
- **NJNL1CSG**: Select nodes with a compatible signon.
- **NJNL1NET**: Select by the subnet name specified in NJNLSUBN (JES2 only).
- **NJNL1SNA**: Select nodes using the SNA protocol (JES3 Only).
- **NJNL1BSC**: Select nodes using the BSC protocol (JES3 Only).
- **NJNL1TCP**: Select nodes using the TCP protocol (JES3 Only).

**NJNLFLT2** Filter by node attributes:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNL2PMY</td>
<td>Select nodes managed by path manager (JES2 Only)</td>
<td></td>
</tr>
<tr>
<td>NJNL2PMN</td>
<td>Select nodes not managed by path manager (JES2 Only)</td>
<td></td>
</tr>
<tr>
<td>NJNL2TLS</td>
<td>Select nodes using secure sockets (JES3 Only)</td>
<td></td>
</tr>
</tbody>
</table>

**NJNLFLT2** Filter by connection status:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNL2OWN</td>
<td>Select only the own node (JES2) or the home node (JES3). This filter should not be used with any other connection filter.</td>
<td></td>
</tr>
<tr>
<td>NJNL2ADJ</td>
<td>Select adjacent nodes. An adjacent node is one hop away from the own (local) node.</td>
<td></td>
</tr>
<tr>
<td>NJNL2DIR</td>
<td>Select directly attached nodes.</td>
<td></td>
</tr>
</tbody>
</table>
Directly attached nodes are adjacent nodes that use dedicated lines.

**NJNLCCNC** Select connected nodes. A connected node is one that JES can communicate with in order to send data.

**NJNLNCN** Select not-connected nodes. A not-connected node is configured, but JES is unable to communicate with it.

**NJNLCPDN** Select nodes pending connection.

**NJNLVIA** Select nodes connected with the adjacent node that is specified in NJNLNOD2.

**NJNLFLTA** Filter by node authority (JES2 Only):

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNLADCY</td>
<td>Select nodes with authority to device commands</td>
</tr>
<tr>
<td>NJNLADCN</td>
<td>Select nodes without authority to device commands</td>
</tr>
<tr>
<td>NJLAJCY</td>
<td>Select nodes with authority to job commands</td>
</tr>
<tr>
<td>NJLJCN</td>
<td>Select nodes without authority to job commands</td>
</tr>
<tr>
<td>NJLANY</td>
<td>Select nodes with authority to net commands</td>
</tr>
<tr>
<td>NJLANCN</td>
<td>Select nodes without authority to net commands</td>
</tr>
<tr>
<td>NJLASCY</td>
<td>Select nodes with</td>
</tr>
</tbody>
</table>
SSI Function Code 82

authority to system commands

**NJNLASCN** Select nodes without authority to system commands

If none of the following filters is specified, the SSI only returns data from the system where the SSI was called. To request information from other systems in a JESplex, specify the MVS system name or JES member selection filters.

**NJNLFLTS**
Filter by MVS System name or JES Member name (JES2 Only)

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NJNLSSYS</strong> Filter by the MVS System name specified by NJNLSSYS</td>
<td></td>
</tr>
<tr>
<td><strong>NJNLSMBR</strong> Filter by the JES Member name specified by NJNLSMBR</td>
<td></td>
</tr>
</tbody>
</table>

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

For the NJE Node Information service function code SSJPNJRS (release storage), the caller should not alter any fields in the IAZJPNJN control block returned on the last SSJPNJOD function call.

**Output Register Information:** When control returns to the caller, the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 -- 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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)**

<table>
<thead>
<tr>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSRTOK (0)</strong> The NJE Node Information services request completed. Check the SSOBRETN field for specific function information.</td>
</tr>
<tr>
<td><strong>SSRTNSUP (4)</strong> The subsystem specified in the SSIBSSNM field does not support the NJE Node Information services function call.</td>
</tr>
</tbody>
</table>
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
• SSJPRETN
• IAZJPNJN (NJE Node Information service)

SSOBRETN Contents: When control returns to the caller and register 15 contains a zero, the NJE Node Information services function places one of the following decimal values in the SSOBRETN field:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPOK (0)</td>
<td>Request successful.</td>
</tr>
</tbody>
</table>

SSJPERRW (4)
Request completed with possible errors, see SSJPRETN for reason code.

SSJPERRU (8)
Request cannot be completed because of user error, see SSJPRETN for reason code.

SSJPERRJ (12)
Request cannot be completed, see SSJPRETN for reason code.

SSJPPARM (16)
Error in the parameter list. For example, the SSJP extension has an invalid format:
• It is not an SSJP
• The service version number is not supported
• The SSJP is not large enough

SSJPSTOR (20)
Request cannot be processed because required storage cannot be obtained. No data can be returned to the caller.

SSJPRETN Contents: In addition to the return code in SSOBRETN, the field SSJPRETN contains the service related error or more specific information about the error. SSJPRETN can be set to one of the following values if SSOBRETN is not zero:
SSI Function Code 82

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPUNSF (4)</td>
<td>Unsupported subfunction requested.</td>
</tr>
<tr>
<td>SSJPNTDS (8)</td>
<td>SSJPU does not point to the correct control block.</td>
</tr>
<tr>
<td>SSJPUNSD (12)</td>
<td>Version number in the control block pointed to by SSJPU is not correct.</td>
</tr>
<tr>
<td>SSJPSMLE (16)</td>
<td>Length field in the control block pointed to by SSJPU is too small.</td>
</tr>
<tr>
<td>SSJPEYEE (20)</td>
<td>Eyecatcher in the control block pointed to by SSJPU is not correct.</td>
</tr>
<tr>
<td>SSJPGETM (128)</td>
<td>$GETMAIN failed.</td>
</tr>
<tr>
<td>SSJPSTGO (132)</td>
<td>STORAGE OBTAIN failed.</td>
</tr>
<tr>
<td>SSJPINVA (136)</td>
<td>Invalid filter arguments were specified.</td>
</tr>
<tr>
<td>SSJPGlbl (140)</td>
<td>Function not supported on the global (JES3 only).</td>
</tr>
<tr>
<td>NJNDSUBF (256)</td>
<td>Function code specified in SSJPFREQ not supported.</td>
</tr>
<tr>
<td>NJNDSpte (260)</td>
<td>Invalid NJNLSTRP pointer.</td>
</tr>
<tr>
<td>NJNDRNGE (264)</td>
<td>The high bound for the range of node numbers that is specified by NJNLRNGH is less than the low bound specified by NJNLRNGL</td>
</tr>
<tr>
<td>NJNDRNGZ (268)</td>
<td>The low bound for the range of node numbers, NJNLRNGL, is set to zero.</td>
</tr>
<tr>
<td>NJNDOWNE (272)</td>
<td>The own or home node filter, NJNLCPWN, should not be used with other connection status filters.</td>
</tr>
<tr>
<td>NJNDSTRE (276)</td>
<td>The caller did not provide enough storage to hold all the data returned by the subfunction call.</td>
</tr>
<tr>
<td>NJNDINTE (280)</td>
<td>Internal error building the system information data area.</td>
</tr>
</tbody>
</table>

**NJE Node Information service, IAZJPNJN contents:** For the NJE Node Information service (function code SSJPNJOD) the following is returned in IAZJPNJN:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNLSVRM</td>
<td>Subsystem version number (currently 1).</td>
</tr>
</tbody>
</table>
NJNLDPTR  Pointer to data for first NJE node data area.
NJNLMPTR  Pointer to first system information data area.
NJNLDDNUM Number of NJE node data areas returned.
NJNLMMNUM Number of member data areas returned.

The following DSECTs define data structures returned by NJE node SSI.

After a successful call to the SSI, field NJNLDPTR points to a chain of data areas representing data for each NJE node. In addition, for JES2 only, the field NJNLMPTR points to a chain of data areas representing member information.

For each NJE node that passes the filter requirements, an element is added to the chain pointed to by NJNLDPTR. Each element is composed of the following sections:

<table>
<thead>
<tr>
<th>DSECT Name</th>
<th>DSECT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNHDR</td>
<td>NJE Node Data Header Section</td>
</tr>
<tr>
<td>NJNFPREF</td>
<td>Prefix Section</td>
</tr>
<tr>
<td>NJNCCMN</td>
<td>NJE Node Common Section</td>
</tr>
</tbody>
</table>

In addition to the preceding common sections, JES2 returns the following sections:

<table>
<thead>
<tr>
<th>DSECT Name</th>
<th>DSECT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2NGEN</td>
<td>JES2 General Data Section</td>
</tr>
<tr>
<td>N2NPATH</td>
<td>JES2 Path Information Section</td>
</tr>
</tbody>
</table>

Note: This is an optional section that contains one or more of the following entries:

<table>
<thead>
<tr>
<th>DSECT Name</th>
<th>DSECT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2NPTEN</td>
<td>JES2 Path Information Entry</td>
</tr>
</tbody>
</table>

In addition to the common sections listed above, JES3 returns the following sections:

<table>
<thead>
<tr>
<th>DSECT Name</th>
<th>DSECT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3NGEN</td>
<td>JES3 General Data Section</td>
</tr>
<tr>
<td>N3NPATH</td>
<td>JES3 Path Information Section</td>
</tr>
</tbody>
</table>

Note: This is an optional section that contains one or more of the following entries:

<table>
<thead>
<tr>
<th>DSECT Name</th>
<th>DSECT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3NPTEN</td>
<td>JES3 Path Information Entry</td>
</tr>
</tbody>
</table>

The following is a layout of the various sections of the NJE Node Information output data area.

```
+----------------+          +----------------+
| NODE INFO SECTION|          | NEXT NODE Info |
|-----------------|          |----------------|
| NJNHDR | NHNEXIT | POINTER TO THE NEXT |
|        |         | NJNHDR IN THE CHAIN. |
|        |         | ZERO IF END OF CHAIN. |
| NJNFPREF | Prefix Section |
```
In addition to the Node information, JES2 sets the pointer NLNLMPTR to the first element in a separate chain of system information elements. For JES2, one such element is returned for each SSI call to obtain data. A single element contains an entry for each system that meets the selection filters. These entries contain basic information about the systems in the JESplex that were processed to obtain data for this SSI call. The element consists of the following contiguous data structures:

- Header Section mapped by NJSHDR
- Prefix section mapped by JPSYSPRF in macro IAZJPLXI
- System information section mapped by JPSYSINF in macro IAZJPLXI

Note: Repeated calls to the obtain data subfunction of this SSI (SSJPNJOD) without intervening call to release storage subfunction (SSJPNJRS), will cause data from a new SSI call to be added ahead of the data from an earlier SSI call.

NJE Node Data Header Section:

The fields in the NJNHDR section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNHEYE</td>
<td>Eyecatcher. This should be set to ‘JPNJNODE’.</td>
</tr>
<tr>
<td>NJNHOHDR</td>
<td>Offset to the NJNFREF prefix section.</td>
</tr>
<tr>
<td>NJNHNEXT</td>
<td>Address of next NJE node element.</td>
</tr>
</tbody>
</table>
**Prefix Section:** This section contains the total length of the data returned for an NJE node.

The fields in the NJNFPREF section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNFLNG</td>
<td>Total length of all the sections for this element. This does not include the length of the header section.</td>
</tr>
<tr>
<td>NJNFTYPE</td>
<td>Type of this section.</td>
</tr>
<tr>
<td>NJNFMOD</td>
<td>Modifier for this section.</td>
</tr>
</tbody>
</table>

**NJE Node Common Section:** This section contains attributes common for JES2 and JES3.

The fields in the NJNCMN section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNCLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>NJNCTYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>NJNCMOD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>NJNCNAME</td>
<td>Node name</td>
</tr>
<tr>
<td>NJNCSYSN</td>
<td>Name of the reporting system</td>
</tr>
<tr>
<td>NJNCMBRN</td>
<td>MAS member name of the reporting system (JES2 Only)</td>
</tr>
<tr>
<td>NJNCSFLG</td>
<td>Node status flags:</td>
</tr>
<tr>
<td></td>
<td>Bit Value</td>
</tr>
<tr>
<td>NJSCLCL</td>
<td>Set if this node is the own or home node</td>
</tr>
<tr>
<td>NJSCLCN</td>
<td>Set if this node is a connected node where at least one path is connected</td>
</tr>
<tr>
<td>NJSNDPND</td>
<td>Set if this node is a pending node where at least one path is pending</td>
</tr>
<tr>
<td>NJSADJ</td>
<td>Set if this node is an adjacent node</td>
</tr>
<tr>
<td>NJSDIR</td>
<td>Set if this node is a directly attached node</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNCSFLG1</td>
<td>Processing flags:</td>
</tr>
<tr>
<td></td>
<td>Bit Value</td>
</tr>
<tr>
<td>NJSCLPW</td>
<td>Send the signon password</td>
</tr>
<tr>
<td>NJSVPW</td>
<td>Verify the signon password</td>
</tr>
<tr>
<td>NJSPLPW</td>
<td>Encrypt the job password</td>
</tr>
<tr>
<td>NJSWSL</td>
<td>Local password check (JES3 only)</td>
</tr>
<tr>
<td>NJSWSG</td>
<td>Secure sign-on</td>
</tr>
<tr>
<td>NJSWSG</td>
<td>Compatible sign-on</td>
</tr>
</tbody>
</table>
SSI Function Code 82

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>NJNCFLG2</th>
<th>More processing flags:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNC2TRC</td>
<td></td>
<td>Trace requested</td>
</tr>
<tr>
<td>NJNC2RST</td>
<td></td>
<td>Autoconnect or restart</td>
</tr>
<tr>
<td>NJNC2HDJ</td>
<td></td>
<td>Hold received jobs</td>
</tr>
<tr>
<td>NJNC2HDS</td>
<td></td>
<td>Hold received SYSOUT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NJNCLINE</th>
<th>The associated line name. This line is:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• a dedicated line (JES2 only)</td>
</tr>
<tr>
<td></td>
<td>• a default line (JES3 only)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NJNCRINT</th>
<th>Automatic restart (reconnect) interval in minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJNCRETR</td>
<td>Max number of reconnection retries. Zero in this</td>
</tr>
<tr>
<td></td>
<td>field means an indefinite number of retries.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NJNCSECL</th>
<th>Security label (JES2 only)</th>
</tr>
</thead>
</table>

**JES2 General Data Section:** This section contains node attributes that are unique for JES2.

The fields in the N2NGEN section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2NGLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>N2NGTYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>N2NGMOD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>N2NGNUM</td>
<td>Node number</td>
</tr>
<tr>
<td>N2NGSFLG</td>
<td>Node status flags:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2NGSPMD</td>
<td></td>
<td>Path manager is down</td>
</tr>
<tr>
<td>N2NGSNOP</td>
<td></td>
<td>Non path manager mode</td>
</tr>
<tr>
<td>N2NGSEND</td>
<td></td>
<td>End node (no forwarding)</td>
</tr>
<tr>
<td>N2NGSPRV</td>
<td></td>
<td>Private node</td>
</tr>
<tr>
<td>N2NGSDIR</td>
<td></td>
<td>Only allow direct connection</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N2NGFLG1</th>
<th>Processing flags:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2NG1ADV</td>
<td></td>
<td>Authority to device commands</td>
</tr>
<tr>
<td>N2NG1AJB</td>
<td></td>
<td>Authority to job commands</td>
</tr>
<tr>
<td>N2NG1ANT</td>
<td></td>
<td>Authority to net commands</td>
</tr>
<tr>
<td>N2NG1ASY</td>
<td></td>
<td>Authority to system commands</td>
</tr>
<tr>
<td>N2NG1XMJ</td>
<td></td>
<td>Transmit jobs</td>
</tr>
<tr>
<td>N2NG1XMS</td>
<td></td>
<td>Transmit SYSOUT</td>
</tr>
<tr>
<td>N2NG1RCJ</td>
<td></td>
<td>Receive SYSOUT</td>
</tr>
<tr>
<td>N2NG1RCS</td>
<td></td>
<td>Receive sysout</td>
</tr>
</tbody>
</table>
More processing flags:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2NG2ARS</td>
<td></td>
<td>Accept resistance</td>
</tr>
</tbody>
</table>

N2NGCMPT  Compaction table id
N2NGREST  Node resistance
N2NGSUBN  NJE subnet name
N2NGLOGM  VTAM® logmode
N2NGLOGN  Logon device name
N2NGNSVN  NETSRV name

**JES2 Path Information Section:** This section contains an array of JES2 Path Information Entries.

The fields in the N2NPATH section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2NPLNG</td>
<td>Length of this section including all the Path Information entries</td>
</tr>
<tr>
<td>N2NPTYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>N2NPMOD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>N2NPOENT</td>
<td>Offset to the first Path Information entry</td>
</tr>
<tr>
<td>N2NPNENT</td>
<td>Number of Path Information entries</td>
</tr>
<tr>
<td>N2NPSENT</td>
<td>Size of each Path Information entry</td>
</tr>
</tbody>
</table>

**JES2 Path Information Entry:** This section contains NJE path attributes unique to JES2.

The fields in the N2NPTEN section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N2NPSFLG</td>
<td>Path status flags:</td>
</tr>
<tr>
<td>N2NPSVLN</td>
<td>Connected by line</td>
</tr>
<tr>
<td>N2NPSVMB</td>
<td>Connected by member</td>
</tr>
<tr>
<td>N2NPSAWR</td>
<td>Awaiting reset</td>
</tr>
<tr>
<td>N2NPSSGN</td>
<td>Signon in progress</td>
</tr>
<tr>
<td>N2NPSPND</td>
<td>Connection pending</td>
</tr>
</tbody>
</table>

N2NPNAM1  Set to the intermediate node name when path status is one of the following situations:
- Connected by line: N2NPSVLN
- Connection pending: N2NPSPND
- Awaiting reset: N2NPSAWR

N2NPNAM2  Associated line name or member name. Set to the associated line name if the path status is set to:
- Connected by line: N2NPSVLN
SSI Function Code 82

Set to the associated member name if the path status is set to:

- Connected by member: N2NPSVMB
- Connection pending: N2NPSPND

**JES3 General Data Section:** This section contains attributes unique to JES3.

The fields in the N3NGEN section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3NGLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>N3NGTYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>N3NGMOD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>N3NGSFLG</td>
<td>Node connection status:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3NGSSNA</td>
<td>Connected via SNA</td>
<td></td>
</tr>
<tr>
<td>N3NGSBSC</td>
<td>Connected via BSC</td>
<td></td>
</tr>
<tr>
<td>N3NGSTCP</td>
<td>Connected via TCP</td>
<td></td>
</tr>
<tr>
<td>N3NGSIND</td>
<td>Indirect node</td>
<td></td>
</tr>
<tr>
<td>N3NGSALS</td>
<td>Alias of home node</td>
<td></td>
</tr>
<tr>
<td>N3NGSCTC</td>
<td>CTC node</td>
<td></td>
</tr>
<tr>
<td>N3NGSSGS</td>
<td>Send signature</td>
<td></td>
</tr>
<tr>
<td>N3NGSSGV</td>
<td>Verify signature</td>
<td></td>
</tr>
</tbody>
</table>

| N3NGFLG1   | Processing flags:              |

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3NG1DFC</td>
<td>Default class</td>
<td></td>
</tr>
<tr>
<td>N3NG1XNR</td>
<td>Writer name is required to hold SYSOUT for external writer</td>
<td></td>
</tr>
<tr>
<td>N3NG1NTH</td>
<td>Net hold</td>
<td></td>
</tr>
<tr>
<td>N3NG1TLS</td>
<td>Secure socket (TLS)</td>
<td></td>
</tr>
</tbody>
</table>

| N3NGEPR | NETPR |
| N3NGEPU | NETPU |
| N3NGBUFS | Buffer size                      |
| N3NGPRCL | PRTDEF class                     |
| N3NGTSC1 | PRTTTSO class                    |
| N3NGXWCL | PRTXWTR class                    |
| N3NGPUCL | PUNDEF class                     |
| N3NGPART | Spool partition                  |
| N3NGBDTI | Bulk data transfer (BDT) id      |
N3NGSTRM: Stream
N3NGMAXL: Max number of lines
N3NGNRJT: Number of job transmitters
N3NGNRJR: Number of job receivers
N3NGNROT: Number of output transmitters
N3NGNOR: Number of output receivers

**JES3 Path Information Section:** This section contains an array of JES3 path information entries.

The fields in the N3NPATH section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3NPLNG</td>
<td>Length of this section</td>
</tr>
<tr>
<td>N3NPTYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>N3NPSEQ</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>N3NPENT</td>
<td>Offset to first JES3 Path Information entry</td>
</tr>
<tr>
<td>N3NPSEN</td>
<td>Number of entries</td>
</tr>
<tr>
<td>N3NPSEN</td>
<td>Size of each entry</td>
</tr>
</tbody>
</table>

**JES3 Path Information Entry:** This section contains the NJE path attributes that are unique for JES3.

The fields in the N3NPTEN section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N3NPNNAM</td>
<td>Node name</td>
</tr>
</tbody>
</table>

System information header: The prefix information addressed by this header is mapped by the JPSYSPRF section of the IAZJPLXI macro. In addition, the system information addressed from this prefix section is mapped by the JPSYSINF section of the IAZJPLXI macro.

The fields in the NJSHDR section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJSHEYE</td>
<td>Eyecatcher. Should be set to ‘JPNJSYSI’</td>
</tr>
<tr>
<td>NJSSEOHDR</td>
<td>Offset to first (prefix) section</td>
</tr>
<tr>
<td>NJSHNEXT</td>
<td>Address of the next NJSHDR header element</td>
</tr>
</tbody>
</table>

**SPOOL Volume Information**

The SPOOL Volume Information service provides information about the JES managed SPOOL volumes. Information can be obtained on all SPOOL volumes or filters can be supplied to limit which volumes are returned. The returned information is grouped into partitions to be compatible with how JES3 organizes SPOOL volumes. JES2 will only return one partition structure that contains all the SPOOL volumes being used by JES2.

See the following sections for more information about SPOOL Volume Information:

- [“Type of Request” on page 234]
SSI Function Code 82

- "Use Information"
- "Issued to"
- "Related SSI Codes"
- "Related Concepts"
- "Environment"
- "Input Register Information" on page 235
- "Input Parameters" on page 235
- "Output Register Information" on page 238
- "Return Code Information" on page 238
- "Output Parameters" on page 239

**Type of Request:** Directed SSI Call.

**Use Information:** To use the JES property information services SSI, callers must first decide the function they want to perform. The appropriate parameter list must be obtained and pointed to by SSJUSER.

**Issued to:** A JES subsystem (either primary or secondary). The subsystem does not have to be associated with the requesting address space.

**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:
- IEFSSOBH
- IEFJSSIB
- IAZSSJP
- IAZJPSPL (SPOOL Volume Information)

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, IAZSSJP, and IAZJPSPL, control blocks can be in 24- or 31-bit virtual storage
- **Recovery**  The caller should provide an ESTAE-type recovery environment. See [z/OS MVS Programming: Assembler Services Guide](https://www.ibm.com/support/docview.wss?rs=612&context=COMDOC100000&ssg1=SG24-8217-00) for more information about an ESTAE-type recovery environment.

Figure 24 on page 235 shows the environment at the time of the call for SSI function code 82, SPOOL Volume Information Subfunction.
**Input Register Information:** Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

**Input Parameters:** Input parameters for the function routine are:
- SSOB
- SSIB
- IAZSSJP
- IAZJPSPL (SPOOL Volume Information)

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 82 (SSOBSSJP)</td>
</tr>
</tbody>
</table>

**Figure 24. Environment at Time of Call for SSI Function Code 82, SPOOL Volume Information Subfunction**
### 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 about the life-of-job SSIB.

### SSOBINDV
Address of the function-dependent area (IAZSSJP 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name: name of the subsystem to which this SPOOL Volume Information Services request is directed.</td>
</tr>
</tbody>
</table>

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

### IAZSSJP Contents: The caller must set the following fields in the IAZSSJP control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPID</td>
<td>Eyecatcher for the control block (set to 'SSJP')</td>
</tr>
<tr>
<td>SSJPLEN</td>
<td>Length of the IAZSSJP (SSJPSIZE) control block</td>
</tr>
<tr>
<td>SSJPVER</td>
<td>Input version of the IAZSSJP control block. Set to SSJPVER1 for version 1 of the control block or to SSJPVERC for the current version of the control block.</td>
</tr>
<tr>
<td>SSJPFREQ</td>
<td>Function to be performed on this request. Valid functions and their related SSJPUSER area are:</td>
</tr>
<tr>
<td>Field Value</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SSJPSPOD</td>
<td>IAZJPSPLOL SPOOL Volume Information service, obtain data</td>
</tr>
<tr>
<td>SSJPSPRS</td>
<td>IAZJPSPLOL SPOOL Volume Information service, release storage</td>
</tr>
<tr>
<td>SSJPUSER</td>
<td>Pointer to service specific data area 'IAZJPSPLOL'</td>
</tr>
</tbody>
</table>

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

### SPOOL Volume Information service, IAZJPSPLOL contents: For the SPOOL Volume Information service (function code SSJPSPOD), the caller must set the following fields in the IAZJPSPLOL control block:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSPSSID</td>
<td>Eyecatcher of the control block (set to 'JPSPPOOL')</td>
</tr>
<tr>
<td>JPSPLEN</td>
<td>Length of the IAZJPSPLOL (JPSPSZE) control block</td>
</tr>
</tbody>
</table>
| JPSPVER    | Input version of the IAZJPSPLOL control block. Set to
The caller can also set the following fields in the IAZJPSPL control block on input to limit (or select) which data is returned. If no filters are specified, all data is returned. If any filters are specified, at least one of the filter conditions in each of the separate filters must be matched before data is returned.

Field Name | Description
---|---
JPSPPARF | Partition filters. Each bit corresponds to a filter condition. This filter is matched if at least one of the specified filter conditions is met.

Bit Value | Description
---|---
JPSPFULL | Filter on spool partitions that are FULL (JES3 only)
JPSPNNM | Filter on the spool partition name specified in field JPSPPNAM (JES3 only)
JPSPALD | Filter on spool partitions for which spool allocations are allowed (JES3 only)
JPSPNLAD | Filter on spool partitions for which spool allocations are not allowed (JES3 only)
JPSPLFTP | Filter on the default spool partition (JES3 only)
JPSPIDTA | Filter on spool partitions that contain initialization data (JES3 only)
JPSPNOVF | Filter on spool partitions that cannot overflow (JES3 only)
JPSPPOVF | Filter on spool partitions for which at least one other partition can overflow into them (JES3 only)

JPSPELF1 | Extent Status Filters. Each bit corresponds to a filter condition. This filter will be matched if at least one of the specified filter conditions is met.

Bit Value | Description
---|---
JPSPACT | Extent Active
JPSPSTRT | Extent Starting (JES2 Only)
JPSPDRN | Extent Draining
JPSPHALT | Extent Halting (JES2 Only)
Set all other fields in the IAZJPSPL control block to binary zeros before issuing the initial IEFSSREQ macro invocation.

For the SPOOL Volume Information service function code SSJPSPRS (release storage), the caller should not alter any fields in the IAZJPSPL control block returned on the last SSJPSPOD function call.

**Output Register Information:** When control returns to the caller, the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SSRTOK (0)  The SPOOL Volume Information services request completed. Check the SSOBRETN field for specific function information.

SSRTNSUP (4)  The subsystem specified in the SSIBSSNM field does not support the SPOOL Volume Information services 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
- SSJPRETN
- IAZJPSPL (SPOOL Volume Information service)

**SSOBRETN Contents:**  When control returns to the caller and register 15 contains a zero, the SPOOL Volume Information services function places one of the following decimal values in the SSOBRETN field:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPOK (0)</td>
<td>Request successful.</td>
</tr>
<tr>
<td>SSJPERRW (4)</td>
<td>Request completed with possible errors, see SSJPRETN for reason code.</td>
</tr>
<tr>
<td>SSJPERRU (8)</td>
<td>Request cannot be completed because of user error, see SSJPRETN for reason code.</td>
</tr>
<tr>
<td>SSJPERRJ (12)</td>
<td>Request cannot be completed, SSJPRETN contains internal reason code.</td>
</tr>
<tr>
<td>SSJPPARM (16)</td>
<td>Error in the parameter list. For example, the SSJP extension is an invalid format:</td>
</tr>
<tr>
<td></td>
<td>• It is not an SSJP</td>
</tr>
<tr>
<td></td>
<td>• The service version number is not supported</td>
</tr>
<tr>
<td></td>
<td>• The SSJP is not large enough</td>
</tr>
<tr>
<td>SSJPPSTOR (20)</td>
<td>Request cannot be processed because required storage cannot be obtained. No data can be returned to the caller.</td>
</tr>
</tbody>
</table>

**SSJPRETN Contents:**  In addition to the return code in SSOBRETN, the field SSJPRETN contains the service related error or more specific information about the error. SSJPRETN can be set to one of the following values if SSOBRETN is not zero:
SSI Function Code 82

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPUNSF (4)</td>
<td>Unsupported subfunction requested.</td>
</tr>
<tr>
<td>SSJPNNTDS (8)</td>
<td>SSJUSER does not point to the correct control block.</td>
</tr>
<tr>
<td>SSJPUNSD (12)</td>
<td>Version number in the control block pointed to by SSJUSER is not correct.</td>
</tr>
<tr>
<td>SSJPSMLE (16)</td>
<td>Length field in the control block pointed to by SSJUSER is too small.</td>
</tr>
<tr>
<td>SSJPEYEE (20)</td>
<td>Eyecatcher in the control block pointed to by SSJUSER is not correct.</td>
</tr>
<tr>
<td>SSJPGETM (128)</td>
<td>$GETMAIN failed.</td>
</tr>
<tr>
<td>SSJPSTGO (132)</td>
<td>STORAGE OBTAIN failed.</td>
</tr>
<tr>
<td>SSJPINVA (136)</td>
<td>Invalid filter arguments were specified.</td>
</tr>
<tr>
<td>SSJPGBL (140)</td>
<td>Function not supported on the global (JES3 only).</td>
</tr>
</tbody>
</table>

**SPOOL Volume Information service, IAZJPSPL contents:** For the SPOOL Volume Information service (function code SSJPSPOD), the following parameters are returned in IAZJPSPL:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSPVERO</td>
<td>Subsystem version number (currently 1).</td>
</tr>
<tr>
<td>JPSPLPTR</td>
<td>Pointer to data for first partition data area.</td>
</tr>
<tr>
<td>JPSPNPAR</td>
<td>Number of partition data areas returned.</td>
</tr>
<tr>
<td>JPSPTGT</td>
<td>Number of track groups defined across all partitions.</td>
</tr>
<tr>
<td>JPSPTGIU</td>
<td>Number of track groups in use across all partitions.</td>
</tr>
<tr>
<td>JPSPTKT</td>
<td>Number of tracks across all partitions.</td>
</tr>
<tr>
<td>JPSPTKU</td>
<td>Number of tracks in use across all partitions.</td>
</tr>
</tbody>
</table>

For each Spool Partition that passes the filter requirements, a Spool Partition element is added to the chain pointed to by JPSPLPTR. Each element is composed of the following sections:

<table>
<thead>
<tr>
<th>DSECT Name</th>
<th>DSECT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPPHDR</td>
<td>Partition Header Section</td>
</tr>
<tr>
<td>SPPREF</td>
<td>Partition Prefix Section</td>
</tr>
<tr>
<td>SPPGENI</td>
<td>Partition General Information Section</td>
</tr>
<tr>
<td>SPPJES3I</td>
<td>Partition JES3 Specific Information Section</td>
</tr>
</tbody>
</table>

Each Spool Partition element has a chain of one or more Spool Extent sections. The first of these sections is pointed to by the field SPPFRSTE in the Partition Header section. The Spool Extent information is composed of the following sections:

<table>
<thead>
<tr>
<th>DSECT Name</th>
<th>DSECT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEHDR</td>
<td>Spool Extent Header Section</td>
</tr>
</tbody>
</table>
The following is a layout of the various sections of the Spool Information output data area. The basic layout is a chain of Spool Partition Information sections. Each Partition section has a chain of Extent Information sections.

SSI 82 SPOOL DATA PARAMETER LIST:

```
+----------------+
JPSPL | ... |
  | JPSPLPTR = ... |
+----------------+

NOTE: The output partition rollup values reside here.

SPOOL PARTITION INFO SECTION:

```
+----------------+
SPPHDR | SPPNXTP  --------------> POINTER TO THE NEXT SPPHDR IN THE CHAIN. |
  | SPPFRSTE = ... | ZERO IF END OF CHAIN. |
+----------------+
SPPFRSTE = ... |
+----------------+
SPPJES3I | Optional JES3 Partition Info Section |
+----------------+
```

NOTE: JES2 will always have 1 partition. JES3 can have multiple.

SPOOL EXTENT INFO SECTION:

```
+----------------+
SPEHDR | SPENXTE  --------------> POINTER TO THE NEXT SPEHDR IN THE CHAIN. |
+----------------+
SPEJ2I | Optional JES2 Extent Info |
```

NOTE: Included if JES2
Spool Partition Header Section: Each Spool Partition information element begins with a header section. There can be multiple Spool Partitions returned, and the SPPNXTP pointer is used to navigate to the next Spool Partition Header section in the chain.

Note: JES2 will only have a single information element in this chain.

The fields in the SPPHDR section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPPEYE</td>
<td>Eyecatcher. Should be set to ‘SPOOLPRT’</td>
</tr>
<tr>
<td>SPOPREF</td>
<td>Offset to the prefix section</td>
</tr>
<tr>
<td>SPPNXTP</td>
<td>Address of the next Spool Partition information element</td>
</tr>
<tr>
<td>SPPFRSTE</td>
<td>Address of the first Extent section for this Partition</td>
</tr>
</tbody>
</table>

Spool Partition Prefix Section: This section holds the length of all the information reported for this Spool Partition. This length does not include the length of the Spool Partition Header section: This length does include all the storage needed to report both the Spool Partition sections as well as all the related Extent sections. To get addressability to this section, add the SPOPREF header field to the header (SPPHDR) address.

The fields in the SPPREF section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPPPRLN</td>
<td>Length of the entire Spool Partition element, not including the length of the Partition Header.</td>
</tr>
<tr>
<td>SPPPRTP</td>
<td>Type of this section</td>
</tr>
<tr>
<td>SPPPRMD</td>
<td>Modifier for this section</td>
</tr>
</tbody>
</table>

Spool Partition General Information Section: This section holds details that are common to both JES2 and JES3. To obtain addressability to this section, add the SPOPREF header field and the prefix size (SPPPRSZ) to the header (SPPHDR) address.

The fields in the SPPGENI section are:
### Spool Partition JES3 Specific Information

This section contains a series of flags that describe JES3 Spool Partition information. To get addressability to this section, add the SPPGLN field to the Partition General Information section (SPPENI) address.

The fields in the SPPJES3I section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPP3LN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>SPP3TY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>SPP3MD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>SPP3OPAR</td>
<td>Overflow partition name</td>
</tr>
<tr>
<td>SPP3STSF</td>
<td>Partition Status flags</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPP3ALD</td>
<td>Partition allocation is allowed</td>
</tr>
<tr>
<td>SPP3DFTP</td>
<td>This is the default partition</td>
</tr>
<tr>
<td>SPP3IDTA</td>
<td>Initialization data exists on this partition</td>
</tr>
<tr>
<td>SPP3OVER</td>
<td>This partition has overflowed into another partition</td>
</tr>
<tr>
<td>SPP3POVI</td>
<td>At least one other partition might overflow into this partition</td>
</tr>
<tr>
<td>SPP3POVO</td>
<td>This partition might overflow into another partition</td>
</tr>
</tbody>
</table>
### SSI Function Code 82

#### Spool Extent Header Section:
Each individual Spool Extent Information section begins with a header section. This section holds an offset to the start of the detailed extent information, as well as a pointer to the next extent defined for this spool partition.

To get addressability to this section, use the SPPFRSTE pointer in the partition header (SPPHDR).

There can be multiple extents returned. The EXTNXTE pointer is used to navigate to the next extent in the chain.

To get from this header to the extent prefix section for this extent, add the SPEOPRF offset to the extent header (SPEHDR) address.

The fields in the SPEHDR section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEEYE</td>
<td>Eyecatcher. It should be set to ‘SPOOLEXT’</td>
</tr>
<tr>
<td>SPEOPRF</td>
<td>Offset to the Extent Prefix section</td>
</tr>
<tr>
<td>SPENXTE</td>
<td>Address of the next Extent Header section</td>
</tr>
</tbody>
</table>

#### Spool Extent Prefix Section:
This section holds the combined length of all the sections needed to report information about an individual initiator. To get addressability to this section, add the SPEOPRF header field to the header (SPEHDR) address.

The fields in the SPEPREF section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEPRLN</td>
<td>Combined length of the individual extent sections. This does not include the length of the header section.</td>
</tr>
<tr>
<td>SPEPRTP</td>
<td>Type of this section</td>
</tr>
<tr>
<td>SPEPRMD</td>
<td>Modifier for this section</td>
</tr>
</tbody>
</table>

#### Spool Extent General Information Section:
This section holds general information about the spool extent. To get addressability to this section, add the SPEOPRF header field and the prefix size (SPEPRSZ) to the header (SPEHDR) address.

The fields in the SPEGENI section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEGLN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>SPEGTY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
</tr>
<tr>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>SPEGACT</td>
<td>ACTIVE status</td>
</tr>
<tr>
<td>SPEGSTRT</td>
<td>STARTING status</td>
</tr>
<tr>
<td>SPEGHALT</td>
<td>HALTING status</td>
</tr>
<tr>
<td>SPEGDRN</td>
<td>DRAINING status</td>
</tr>
<tr>
<td>SPEGINAC</td>
<td>INACTIVE status</td>
</tr>
<tr>
<td>SPEGHELD</td>
<td>HELD status (JES3 Only)</td>
</tr>
<tr>
<td>SPEGFLG1</td>
<td>Extent General Status Flags</td>
</tr>
<tr>
<td>SPEGEXTI</td>
<td>Extent identifier. This is the Volume Name in JES2 and the DDNAME in JES3.</td>
</tr>
<tr>
<td>SPEGDSNM</td>
<td>Data set name</td>
</tr>
<tr>
<td>SPEGGTGT</td>
<td>Total track groups</td>
</tr>
<tr>
<td>SPEGGU</td>
<td>Total track groups in use</td>
</tr>
<tr>
<td>SPEGTRKR</td>
<td>Total tracks</td>
</tr>
<tr>
<td>SPEGTKU</td>
<td>Total tracks in use</td>
</tr>
<tr>
<td>SPEGLCYL</td>
<td>Low cylinder. Note that this is a normalized value (cccCC)</td>
</tr>
<tr>
<td>SPEGLHED</td>
<td>Low head</td>
</tr>
<tr>
<td>SPEGLMTR</td>
<td>Low MQTR value for JES2; Low MMRRRR value for JES3</td>
</tr>
<tr>
<td>SPEGLMM</td>
<td>Defines JES3 extent number</td>
</tr>
<tr>
<td>SPEGLRRN</td>
<td>Defines JES3 record number</td>
</tr>
<tr>
<td>SPEGHCYL</td>
<td>High Cylinder. Note this is the normalized value (cccCC)</td>
</tr>
<tr>
<td>SPEGHHED</td>
<td>High head</td>
</tr>
<tr>
<td>SPEGHMTR</td>
<td>High MQTR value for JES2; High MMRRRR value for JES3</td>
</tr>
<tr>
<td>SPEGHMM</td>
<td>Defines JES3 extent number</td>
</tr>
<tr>
<td>SPEGHRRN</td>
<td>Defines JES3 record number</td>
</tr>
<tr>
<td>SPEGTPCY</td>
<td>Tracks per cylinder</td>
</tr>
<tr>
<td>SPEGRPTK</td>
<td>Records per track</td>
</tr>
<tr>
<td>SPEGTPTG</td>
<td>Tracks per track group</td>
</tr>
<tr>
<td>SPEGEXTN</td>
<td>Extent number</td>
</tr>
</tbody>
</table>
SSI Function Code 82

- **SPEGLTRK**: Low Track Number
- **SPEGHTRK**: High Track Number

Spool Extent JES2 Specific Information: This section holds Extent information specific to JES2. To get addressability to this section, add the SPEGLN field to the General Information section (SPEGENI) address.

The fields in the SPEJ2I section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPE2LN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>SPE2TY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>SPE2MD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>SPE2CMD</td>
<td>Current Command string. This is set to blanks if no command is active.</td>
</tr>
<tr>
<td>SPE2CMDB</td>
<td>Current Command byte</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPE2NCMD</td>
<td>No command is Active</td>
</tr>
<tr>
<td>SPE2STRT</td>
<td>START command</td>
</tr>
<tr>
<td>SPE2FRMT</td>
<td>FORMAT command</td>
</tr>
<tr>
<td>SPE2HALT</td>
<td>HALT command</td>
</tr>
<tr>
<td>SPE2DRN</td>
<td>DRAIN command</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPE2STNT</td>
<td>If set ON, this extent is stunted</td>
</tr>
<tr>
<td>SPE2ALLM</td>
<td>If set ON, ALL members have affinity to this volume. The Affinity Array sections DO NOT exist. If set OFF, SOME members have affinity to this volume. The Affinity Array sections DO exist.</td>
</tr>
</tbody>
</table>

Spool Extent JES2 Affinity Specific Information Section: to get addressability to this section, add the SPE2LN field to the JES2 Info section (SPEJ2I) address.

This section only exists if the SPE2ALLM indicator in the JES2 info section (SPEJ2I) is OFF.

The fields in the SPEJ2AI section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPE2ALN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>SPE2ATY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>SPE2AMD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>SPE2ANUM</td>
<td>Number of entries in the Affinity Array.</td>
</tr>
</tbody>
</table>
Note: This number can be ZERO in cases where no members match the selection filters.

**SPE2ALEN**
Length of an entry in the Affinity Array

Spool Extent JES2 Affinity Array Entry: to get addressability to the first array entry, add the SPE2ALN field in the SPEJ2AI JES2 Affinity Array header to the SPEJ2AI Affinity Array header address.

Use the SPE2ANUM and SPE2ALEN fields in the SPEJ2AI JES2 Affinity Array header to loop through the array entries.

This array will not exist if ALL members have affinity to the extent. See the SPE2ALLM indicator for further discussion.

The fields in the SPEJ2AE section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPE2EMBR</td>
<td>JES2 member Name</td>
</tr>
<tr>
<td>SPE2ESYS</td>
<td>MVS System Name</td>
</tr>
</tbody>
</table>

**Spool Extent JES3 Specific Information Section:** This section holds Extent information specific to JES2. To get addressability to this section, add the SPEGLN field to the General Information section (SPEGENI) address.

The fields in the SPEJ3I section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPE3LN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>SPE3TY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>SPE3MD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>SPE3RCSZ</td>
<td>Extent record size</td>
</tr>
<tr>
<td>SPE3FLG1</td>
<td>Extent Status Indicators:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPE3STRK</td>
<td>Set ON for a Single Track Table</td>
</tr>
<tr>
<td>SPE3BTRK</td>
<td>Set ON when the extent contains a bad track</td>
</tr>
</tbody>
</table>

**Initiator Information**

The Initiator Information service reports information about the resources associated with job execution. These resources include groups, the systems on which these groups are enabled, the classes contained by the groups, and the initiators that are allocated for managing jobs.

See the following sections for more information about Initiator Information:

- “Type of Request” on page 248
- “Use Information” on page 248
- “Issued to” on page 248
- “Related SSI Codes” on page 248
- “Related Concepts” on page 248
- “Environment” on page 248
Type of Request: Directed SSI Call.

Use Information: To use the JES property information services SSI, callers must first decide the function they want to perform. The appropriate parameter list must be obtained and pointed to by SSJUSER.

Issued to: A JES subsystem (either primary or secondary). The subsystem does not have to be associated with the requesting address space.

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:
- IEFSSOBH
- IEFJSSIB
- IAZSSJP
- IAZJPITD (Initiator Information)

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, IAZSSJP, and IAZJPITD, control blocks can be in 24- or 31-bit virtual storage
Recovery The caller should provide an ESTAE-type recovery environment. See z/OS MVS Programming: Assembler Services Guide for more information about an ESTAE-type recovery environment.

Figure 25 on page 249 shows the environment at the time of the call for SSI function code 82, Initiator Information Subfunction.
Input Register Information: Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

Input Parameters: Input parameters for the function routine are:
- SSOB
- SSIB
- IAZSSJP
- IAZJPITD (Initiator Information)

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 82 (SSOBSSJP)</td>
</tr>
</tbody>
</table>
### SSIB

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 about the life-of-job SSIB.

### SSOBINDV

Address of the function-dependent area (IAZSSJP control block)

Set all other fields in the SSIB 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name: name of the subsystem to which this Initiator Information Services request is directed.</td>
</tr>
</tbody>
</table>

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

**IAZSSJP Contents:** The caller must set the following fields in the IAZSSJP control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPID</td>
<td>Eyecatcher for the control block (set to 'SSJP')</td>
</tr>
<tr>
<td>SSJPLEN</td>
<td>Length of the IAZSSJP (SSJPSIZE) control block</td>
</tr>
<tr>
<td>SSJPVER</td>
<td>Input version of the IAZSSJP control block. Set to SSJPVER1 for version 1 of the control block or to SSJPVERC for the current version of the control block.</td>
</tr>
<tr>
<td>SSJPFREQ</td>
<td>Function to be performed on this request. Valid functions and their related SSJPUSER area are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPITOD</td>
<td>IAZJPITD Initiator Information service, obtain data</td>
</tr>
<tr>
<td>SSJPITRS</td>
<td>IAZJPITD Initiator Information service, release storage</td>
</tr>
<tr>
<td>SSJPUSER</td>
<td>Pointer to service specific data area '(IAZJPITD)'</td>
</tr>
</tbody>
</table>

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

**Initiator Information service, IAZJPITD contents:** For the Initiator Information service (function code SSJPITOD), the caller must set the following fields in the IAZJPITD control block:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPITSSID</td>
<td>Eyecatcher of the control block (set to ' JPINITDT')</td>
</tr>
<tr>
<td>JPLLEN</td>
<td>Length of the IAZJPITD (JPITSZE) control block</td>
</tr>
<tr>
<td>JPITSVVER</td>
<td>Input version of the IAZJPITD control block. Set to JPITSVR1 for version 1 of the control block. Set to JPITSVR# for the current (latest) version</td>
</tr>
</tbody>
</table>
JPITSTRP

Storage management anchor for use by the subsystem that responds to this request. It is expected that the caller will set this field to zero the first time IAZJPITD is used. From that point on, the field will be managed by the subsystem.

The caller can also set the following fields in the IAZJPITD control block on input to limit (or select) which data will be returned.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPITFLG1</td>
<td>Flag byte which describes which filters to use to limit or select the data to be returned. Each bit corresponds to a filter that must be matched before data is returned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPIT1GRP</td>
<td>Return initiator information if one or more groups match the group name indicated by JPITGNAM.</td>
</tr>
<tr>
<td>JPIT1NAS</td>
<td>Return initiator information if one or more systems match the MVS system name indicated by JPITSNAM.</td>
</tr>
<tr>
<td>JPIT1NAM</td>
<td>Return initiator information if one or more members match the JES member name indicated by JPITMNAM.</td>
</tr>
</tbody>
</table>

**Note:** For JES3, the JES member name is synonymous with the MVS system name.

The following two bits are used together to determine class filtering. If the first bit (JPIT1CLS) is set to ON, it indicates that class filtering is requested. If class filtering is requested, then the second bit JPIT1CLW has meaning:

- If JPIT1CLW is OFF, the caller is requesting jobclass filtering.
- If JPIT1CLW is ON, the caller is requesting service class filtering.

**JES2 Usage:**

Jobclass filtering for JES2 Initiators returns any JES2 Initiators that have the one character jobclass specified in field JPITSCCLS in their list of supported job classes. Jobclass filtering is not valid if WLM group filtering is requested.

Service class filtering for WLM initiators returns any WLM initiators that are selecting on the service class specified in field JPITSCCLS. Wildcard names are supported for service class filtering. Service class filtering is not valid if JES2 group filtering is requested.

**JES3 Usage:**
For JES3, class filtering will only take place if JPIT1CLS is set ON and JPIT1CLW is set OFF. JES3 will only do class filtering for job classes.

**Note:** JES3 accepts 8-character job classes.

**JPIT1CLS**
Return initiator information if one or more classes match the class name indicated by JPITSCLS.

**JPIT1CLW**
Interpret the class filtering as looking for either job classes or service classes (Service class filtering only allowed for JES2).

**JPIT1DOM**
If this bit is set ON, an authorized caller is requesting a security label dominance check for batch job data (JES2 only)

**JPITGNAM**
Filter field JPITGNAM might contain an Initiator group name. The bit JPIT1GRP indicates whether filter JPITGNAM is used. JES2 accepts the constant group names "JES2" and "WLM". JES3 group names are not constants, Also for JES3, wildcard names are supported.

**JPITSNAM**
Filter field JPITSNAM might contain an MVS System Name. Bit JPIT1NAS indicates whether the filter JPITSNAM is used. Wildcard names are supported.

**Note:** For JES3, the MVS System name is the same as the JES Member name.

**JPITMNAM**
Filter field JPITMNAM might contain a JES member name. Bit JPIT1NAM indicates whether the filter JPITMNAM is used. Wildcard names are supported.

**JPITSCLS**
Filter field JPITSCLS might contain a service or job class. See the descriptions for filter bits JPIT1CLS and JPIT1CLW for usage information.

**JPITSTAT**
Report Initiator information for only those Initiators that have the following status. If no status is specified, all initiators are reported (JES2 only).

**JPITSDRI**
Return information for initiators in the Draining state.

**JPITSDRD**
Return information for initiators in the Drained state.

**JPITSHLI**
Return information for initiators in the Halting state.

**JPITSHLD**
Return information for initiators in the Halted state.
JPITSINA
Return information for initiators in the
Inactive state.

JPITSACT
Return information for initiators in the Active
state.

JPITSSTR
Return information for initiators in the
Starting state.

Note: If the caller asks to look for WLM groups
(JPIT1GRP is ON and JPITGNAM is set to
'WLM'), only JPITSINA or JPITSACT can be
requested.

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

For the Initiator Information service function code SSJPITRS (release storage), the
caller should not alter any fields in the IAZJPITD control block returned on the last
SSJPITOD function call.

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>The Initiator Information services request completed. Check the SSOBRETN field for specific function information.</td>
</tr>
<tr>
<td>SSRTNSUP (4)</td>
<td>The subsystem specified in the SSIBSSNM field does not support the Initiator Information services function call.</td>
</tr>
<tr>
<td>SSRTNTUP (8)</td>
<td>The subsystem specified in the SSIBSSNM field exists but is not active.</td>
</tr>
<tr>
<td>SSRTNOSS (12)</td>
<td>The subsystem specified in the SSIBSSNM field is not defined to MVS.</td>
</tr>
<tr>
<td>SSRTDIST (16)</td>
<td>The pointer to the SSOB control block or the SSIB control block is not valid, or the function code</td>
</tr>
</tbody>
</table>
SSI Function Code 82

If the value 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
- SSJPRETN
- IAZJPITD (Initiator Information service)

**SSOBRETN Contents:** When control returns to the caller and register 15 contains a zero, the Initiator Information services function places one of the following decimal values in the SSOBRETN field:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPOK (0)</td>
<td>Request successful.</td>
</tr>
<tr>
<td>SSJPERRW (4)</td>
<td>Request completed with possible errors, see SSJPRETN for reason code.</td>
</tr>
<tr>
<td>SSJPERRU (8)</td>
<td>Request cannot be completed because of user error, see SSJPRETN for reason code.</td>
</tr>
<tr>
<td>SSJPERRJ (12)</td>
<td>Request cannot be completed, SSJPRETN contains internal reason code.</td>
</tr>
<tr>
<td>SSJPPARM (16)</td>
<td>SSJP extension is in an invalid format.</td>
</tr>
<tr>
<td></td>
<td>• It is not an SSJP.</td>
</tr>
<tr>
<td></td>
<td>• The service version number is not supported.</td>
</tr>
<tr>
<td></td>
<td>• The SSJP is not large enough</td>
</tr>
<tr>
<td>SSJPSTOR (20)</td>
<td>Request cannot be processed because required storage cannot be obtained. No data can be returned to the caller.</td>
</tr>
</tbody>
</table>

**SSJPRETN Contents:** In addition to the return code in SSOBRETN, the field SSJPRETN contains the service related error or more specific information about the error. SSJPRETN can be set to one of the following values if SSOBRETN is not zero:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPUNSF (4)</td>
<td>Unsupported subfunction requested.</td>
</tr>
<tr>
<td>SSJPNTDS (8)</td>
<td>SSJPUSER does not point to the correct control block.</td>
</tr>
<tr>
<td>SSJPUNSD (12)</td>
<td>Version number in the control block pointed to by SSJPUSER is not correct.</td>
</tr>
<tr>
<td>SSJPSMLE (16)</td>
<td>Length field in the control block pointed to by SSJPUSER is too small.</td>
</tr>
<tr>
<td>SSJPEYEE (20)</td>
<td>Eyecatcher in the control block pointed to by SSJPUSER is not correct.</td>
</tr>
<tr>
<td>SSJPGETM (128)</td>
<td>$GETMAIN failed.</td>
</tr>
<tr>
<td>SSJPSTGO (132)</td>
<td>STORAGE OBTAIN failed.</td>
</tr>
<tr>
<td>Function Code</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SSJPINVA (136)</td>
<td>Invalid filter arguments were specified.</td>
</tr>
<tr>
<td>SSJPGLBL (140)</td>
<td>Function not supported on the global (JES3 only).</td>
</tr>
<tr>
<td>JPITJ2SC (256)</td>
<td>User supplied group of JES2 is not valid with the service class</td>
</tr>
<tr>
<td>JPITWLJC (260)</td>
<td>User supplied group of WLM is not valid with the job class</td>
</tr>
<tr>
<td>JPITWLST (264)</td>
<td>User supplied group of WLM is not valid with a status filter other than 'Active'</td>
</tr>
<tr>
<td>JPITEYLN (268)</td>
<td>User provided bad storage for the Initiator Information service specific data area. Either the eyecatcher was incorrect, or the length of the data area was incorrect.</td>
</tr>
<tr>
<td>JPITJCLN (272)</td>
<td>User supplied jobclass for the JES2 group filter is longer than one character.</td>
</tr>
</tbody>
</table>

Initiator Information service, IAZJPITD contents: For the Initiator Information service (function code SSJPITOD), the following parameters are returned in IAZJPITD:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPITVERO</td>
<td>Subsystem version number (currently 1)</td>
</tr>
<tr>
<td>JPITDPTR</td>
<td>Pointer to data for first Initiator data area.</td>
</tr>
<tr>
<td>JPITNIG</td>
<td>Number of Initiator data areas returned.</td>
</tr>
</tbody>
</table>

For each initiator group that passes the filter requirements, an element is added to the chain pointed to by JPITDPTR. Each element is composed of the following sections:

<table>
<thead>
<tr>
<th>DSECT Name</th>
<th>DSECT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITGHDHDR</td>
<td>Initiator Group Header Section</td>
</tr>
<tr>
<td>ITGPDGRP</td>
<td>Initiator Group Prefix Section</td>
</tr>
<tr>
<td>ITGGDGGI</td>
<td>Initiator Group General Information Section</td>
</tr>
</tbody>
</table>

In addition to the common sections listed above, JES3 returns these additional group sections:

<table>
<thead>
<tr>
<th>DSECT Name</th>
<th>DSECT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT3GDG3I</td>
<td>JES3 Group Information Section</td>
</tr>
<tr>
<td>IT3HDG3S</td>
<td>JES3 Group System Information Section will contain one or more IT3SDISY JES3 System Information Entry sections.</td>
</tr>
<tr>
<td>IT3SDISY</td>
<td>JES3 System Information Entry</td>
</tr>
<tr>
<td>IT3JDG3J</td>
<td>IT3JDG3J JES3 Group Jobclass Information Section will contain one or more IT3CD3JC JES3 Jobclass Entry sections.</td>
</tr>
<tr>
<td>IT3CD3JC</td>
<td>JES3 Jobclass Entry</td>
</tr>
</tbody>
</table>
Each Initiator Group has a chain of zero or more initiator entries. The first initiator is pointed to by field ITGHINIT in the Initiator Group Header section. A count of the number of initiators returned for the group is stored in field ITGHNINT in the Initiator Group Header section.

The Initiator Header and its related sections are only returned by JES2. The data returned for each Initiator contains these sections:

<table>
<thead>
<tr>
<th>DSECT Name</th>
<th>DSECT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITIHDIHD</td>
<td>Initiator Header Section</td>
</tr>
<tr>
<td>ITIPDINT</td>
<td>Initiator Prefix Section</td>
</tr>
<tr>
<td>ITIGDIGI</td>
<td>Initiator General Information Section</td>
</tr>
<tr>
<td>IT2IDI2I</td>
<td>JES2 Initiator Information Section</td>
</tr>
<tr>
<td>IT2JDI2J</td>
<td>JES2 Initiator Jobclass Information Section will contain one or more of the IT2CDIJC sections.</td>
</tr>
<tr>
<td>IT2CDIJC</td>
<td>JES2 Jobclass Entry Section</td>
</tr>
</tbody>
</table>

The following example is a layout of the various sections of the Initiator Information output data area.

```
INITIATOR GROUP HEADER SECTION
+----------------+
| ITGHDHDR | ITGHNEXT =----------> POINTER TO THE NEXT ITGHDHDR IN THE CHAIN. |
|          | ITGHINIT =----------> ZERO IF END OF CHAIN. |
|          | ... |
| ITGPDGRP | Prefix Section |
|          | ... |
| ITGGDGGI | General Information Section |
|          | ... |
| IT3GDG3I | Optional JES3 Group Info Section |
|          | ... |
| IT3HDG3S | Optional JES3 Group System Info Section |
|          | ... |
| IT3DISY  | Optional JES3 System Info Array Elements |
| (1..N)   | ... |
| IT3JDG3J | Optional JES3 Group Jobclass Info Section |
|          | ... |
| IT3CDJJC | Optional JES3 Jobclass Array Elements |
| (1..N)   | ... |
```

NOTE: JES2 will always have 2 groups, one for JES2 initiators. JES3 can have multiple groups.

NOTE: # elements is IT3H3SNS

NOTE: # elements is IT3JJCCT

SSI Function Code 82
**Initiator Group Header Section:** Each Initiator information element begins with a Group Header. This section holds an offset to the group, system and class details, and holds a pointer to the initiator section, which is a chain of the initiators that pertain to this group.

The fields in the ITGHDHDR section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITGHEYE</td>
<td>Eyecatcher. Should be set to ‘DINITGRP’.</td>
</tr>
<tr>
<td>ITGHOHDR</td>
<td>Offset to first group section.</td>
</tr>
<tr>
<td>ITGHNEXT</td>
<td>Address of the next Initiator Group Information element.</td>
</tr>
<tr>
<td>ITGHINIT</td>
<td>Address of the first Initiator section for this group (JES2 Only).</td>
</tr>
<tr>
<td>ITGHNINT</td>
<td>The number of initiator sections for this group (JES2 Only).</td>
</tr>
</tbody>
</table>

**Initiator Group Prefix Section:** This holds the length of all the information reported for this group. This length does not include the length of the Group Header section. This length does include all storage needed to report both the Group, Class and System information as well as the storage needed to report all the related Initiator sections.

The fields in the ITGPDGRP section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
SSI Function Code 82

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITGPGLEN</td>
<td>Length of the entire element, not including the length of the Group Header.</td>
</tr>
<tr>
<td>ITGPGTYP</td>
<td>Type of this section.</td>
</tr>
<tr>
<td>ITGPGMOD</td>
<td>Modifier for this section.</td>
</tr>
</tbody>
</table>

**Initiator Group General Information Section:** The Group Name field ITGGGAM is set to ‘JES2’ when JES2 is reporting its initiators, and to ‘WLM’ when JES2 is reporting the WLM initiators. When JES3 is reporting its initiators, this field will be set to the configured JES3 Group Name.

The fields in the ITGGDGGI section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITGGLEN</td>
<td>Length of this section.</td>
</tr>
<tr>
<td>ITGGTYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>ITGGMOD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>ITGGGNAME</td>
<td>Group Name</td>
</tr>
<tr>
<td>ITGGFLAG</td>
<td>Group Flags</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITGGWLM</td>
<td>ON for WLM group processing</td>
</tr>
<tr>
<td></td>
<td>OFF for JES group processing</td>
</tr>
</tbody>
</table>

**JES3 Group Information Section:** This section contains general information about a JES3 group. It contains the group barrier definition, and reports if this is the JES3 default group.

The fields in the IT3GDG3I section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT3GLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>IT3TYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>IT3GMOD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>IT3G3IBR</td>
<td>JES3 group barrier</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Barrier value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-15</td>
<td>job priority</td>
</tr>
<tr>
<td>16</td>
<td>no barrier</td>
</tr>
<tr>
<td>PRTY</td>
<td>each job priority is a barrier</td>
</tr>
</tbody>
</table>

**IT3GFLAG**

JES3 group flag

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT3GDEFG</td>
<td>ON if this is the JES3 default group</td>
</tr>
<tr>
<td></td>
<td>OFF if this is not the JES3 default group</td>
</tr>
</tbody>
</table>
**JES3 Group System Information Section:** This section contains an offset to the first JES3 System Information section, as well as the number of the JES3 System Information Entry sections.

The fields in the IT3HDG3S section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT3HLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>IT3HTYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>IT3HMOD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>IT3H3SOS</td>
<td>Offset to the first of the JES3 System Information Entry sections</td>
</tr>
<tr>
<td>IT3H3SNS</td>
<td>Number of JES3 System Information Entry sections</td>
</tr>
<tr>
<td>IT3H3SLS</td>
<td>Length of a JES3 System Information entry</td>
</tr>
</tbody>
</table>

**JES3 System Information Entry:** This section holds information about one of the systems on which the group is enabled to run.

The fields in the IT3DISY section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT3SSYSN</td>
<td>System Name</td>
</tr>
<tr>
<td>IT3SDICT</td>
<td>Count of initiators defined for this system</td>
</tr>
<tr>
<td>IT3SAICT</td>
<td>Count of initiators allocated for this system</td>
</tr>
<tr>
<td>IT3SUICT</td>
<td>Count of initiators in use for this system</td>
</tr>
<tr>
<td>IT3SFLAG</td>
<td>Flag Byte</td>
</tr>
<tr>
<td>Bit Name</td>
<td>Description</td>
</tr>
<tr>
<td>IT3SMANA</td>
<td>ON for Manual allocation</td>
</tr>
<tr>
<td>OFF for Dynamic Allocation</td>
<td></td>
</tr>
<tr>
<td>IT3SMANU</td>
<td>ON for Manual unallocation</td>
</tr>
<tr>
<td>OFF for Dynamic unallocation</td>
<td></td>
</tr>
</tbody>
</table>

**JES3 Group Job-Class Information Section:** This section contains an offset to the first JES3 Job-Class Information, as well as the number of the JES3 Job-Class Entry sections contained by this group.

The fields in the IT3JDG3J section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT3JLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>IT3JTYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>IT3JMOD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>IT3JJC0F</td>
<td>Offset to the first Job-Class Entry section</td>
</tr>
<tr>
<td>IT3JJCCT</td>
<td>Number of Job-Class Entry sections for this Group</td>
</tr>
<tr>
<td>IT3JJCCLN</td>
<td>Length of a single Job-Class entry</td>
</tr>
</tbody>
</table>
JES3 Job-Class Entry: This section holds information about one of the
Job-Classes contained by this Group.

The fields in the IT3CD3JC section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT3CJCNM</td>
<td>Job-Class name</td>
</tr>
</tbody>
</table>

The following individual initiator sections are only returned by JES2.

Initiator Header Section: Each individual Initiator Information section begins with a
header section. This section holds an offset to the start of the detailed initiator
information, as well as a pointer to the next initiator defined for the group.

The fields in the ITIHDIHD section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITIHIEYE</td>
<td>Eyecatcher. Should be set to ‘DINITDTA’.</td>
</tr>
<tr>
<td>ITIHOHDR</td>
<td>Offset to the prefix section</td>
</tr>
<tr>
<td>ITIHNEXT</td>
<td>Address of the next individual initiator information section.</td>
</tr>
</tbody>
</table>

Initiator Prefix Section: This section holds the combined length of all the sections
needed to report information about an individual initiator.

The fields in the ITIPDINT section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITIPILEN</td>
<td>Combined length of the individual initiator sections. This does not include the length of the header section.</td>
</tr>
<tr>
<td>ITIPITYP</td>
<td>Type of this section</td>
</tr>
<tr>
<td>ITIPIMOD</td>
<td>Modifier for this section</td>
</tr>
</tbody>
</table>

Initiator General Information Section: This section holds general information
about an individual initiator.

The fields in the ITIGDIGI section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITIGIILN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>ITIGIITY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>ITIGIIMD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>ITIGASID</td>
<td>Address Space Identifier for the initiator job</td>
</tr>
<tr>
<td>ITIGSTAT</td>
<td>Initiator Status flag</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITIGIDRI</td>
<td>Draining</td>
<td></td>
</tr>
<tr>
<td>ITIGDDRD</td>
<td>Drained</td>
<td></td>
</tr>
<tr>
<td>ITIGIHLI</td>
<td>Halting</td>
<td></td>
</tr>
</tbody>
</table>
The following fields are associated with the currently active batch job in the initiator:

- **ITIGJNAM**: Job name from the job card
- **ITIGJBID**: Job ID of the batch job
- **ITIGOWNR**: User ID from the job card
- **ITIGSTEP**: Job step name
- **ITIGPRSN**: Procedure step name
- **ITIGSECL**: SECLABEL for the address space
- **ITIGJCLS**: Job class
- **ITIGSCLS**: If the currently active job is JES managed, this is the service class of that job. If the job is WLM managed, this is the service class the WLM Initiator is currently selecting on.

**JES2 Initiator Information Section**: This information is not available for WLM managed initiators.

The fields in the IT2IDI2I section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT2ILEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>IT2ITYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>IT2IMOD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>ITIITID</td>
<td>Initiator partition identifier</td>
</tr>
<tr>
<td>ITIITJI</td>
<td>Initiator job identifier</td>
</tr>
</tbody>
</table>

**JES2 Initiator Job-Class Information Section**: This information is not available for WLM managed initiators. Field IT2JJCOS is the offset to the first job-class entry section, and field IT2JJCCT indicates the number of job-class entries. The length of these entries is stored in IT2JJCLN, and these entries follow immediately after this section.

The fields in the IT2JDI2J section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT2JLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>IT2JTYPE</td>
<td>Type of this section</td>
</tr>
<tr>
<td>IT2JMOD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>ITJJCOS</td>
<td>Offset to the first job-class entry</td>
</tr>
</tbody>
</table>
SSI Function Code 82

**ITJJCCT**  Number of job-class entries

**ITJJCLN**  Length of a single job-class entry

**JES2 Job-Class Entry:** Each JES2 Initiator can support multiple job classes.

The fields in the IT2CDIJC section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT2CJCNM</td>
<td>Job-Class name</td>
</tr>
<tr>
<td>IT2CFLAG</td>
<td>Flag byte</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| IT2CJCWY | ON if the job-class is WLM eligible  
OFF if the job-class is not WLM eligible |

**JESplex Information**

The JESplex Information service provides detailed information about the characteristics of each of the systems or members in a JESplex. These characteristics include the name, current status, last start type, and operating system level of each of the systems or members.

See the following sections for more information about JESplex Information:

- "Type of Request"
- "Use Information"
- "Issued to"
- "Related SSI Codes"
- "Related Concepts"
- "Environment"
- "Input Register Information" on page 264
- "Input Parameters" on page 264
- "Output Register Information" on page 267
- "Return Code Information" on page 267
- "Output Parameters" on page 268

**Type of Request:** Directed SSI Call.

**Use Information:** To use the JES property information services SSI, callers must first decide the function they want to perform. The appropriate parameter list must be obtained and pointed to by SSJUSER.

**Issued to:** A JES subsystem (either primary or secondary). The subsystem does not have to be associated with the requesting address space.

**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:

- IEFSSOBH
- IEFJSSIB
- IAZSSJP
- IAZJPLEX (JESplex Information)

The caller must meet the following requirements:

<table>
<thead>
<tr>
<th>Minimum Authorization</th>
<th>Problem state, any PSW key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatchable unit mode</td>
<td>Task</td>
</tr>
<tr>
<td>AMODE</td>
<td>24-bit or 31-bit</td>
</tr>
<tr>
<td>Cross memory mode</td>
<td>PASN=HASN=SASN</td>
</tr>
<tr>
<td>ASC mode</td>
<td>Primary</td>
</tr>
<tr>
<td>Interrupt status</td>
<td>Enabled for I/O and external interrupts</td>
</tr>
<tr>
<td>Locks</td>
<td>No locks held</td>
</tr>
<tr>
<td>Control Parameters</td>
<td>The SSOB, SSIB, IAZSSJP, and IAZJPLEX, control blocks can be in 24- or 31-bit virtual storage</td>
</tr>
<tr>
<td>Recovery</td>
<td>The caller should provide an ESTAE-type recovery environment. See <a href="https://www.ibm.com/support/knowledgecenter/ST61S7_5.1.0/com.ibm.doc.asm6h.doc_5.1.0/reading/g_gde_recover.html">z/OS MVS Programming: Assembler Services Guide</a> for more information about an ESTAE-type recovery environment.</td>
</tr>
</tbody>
</table>

Figure 26 on page 264 shows the environment at the time of the call for SSI function code 82, JESplex Information Subfunction.
SSI Function Code 82

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

Input Parameters: Input parameters for the function routine are:
- SSOB
- SSIB
- IAZSSJP
- IAZJPLEX (JESplex Information)

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 82 (SSOBSSJP)</td>
</tr>
</tbody>
</table>
**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 about the life-of-job SSIB.

**SSOBINDV**  Address of the function-dependent area (IAZSSJP control block)

Set all other fields in the SSIB 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier ‘SSIB’</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem to which this JESplex Information Services request is directed.</td>
</tr>
</tbody>
</table>

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

**IAZSSJP Contents:** The caller must set the following fields in the IAZSSJP control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPID</td>
<td>Eyecatcher for the control block (set to ‘SSJP’)</td>
</tr>
<tr>
<td>SSJPLEN</td>
<td>Length of the IAZSSJP (SSJPSIZE) control block</td>
</tr>
<tr>
<td>SSJPVER</td>
<td>Input version of the IAZSSJP control block. Set to SSJPVER1 for version 1 of the control block or to SSJPVERC for the current version of the control block.</td>
</tr>
<tr>
<td>SSJPFREQ</td>
<td>Function to be performed on this request. Valid functions and their related SSJPUSER area are:</td>
</tr>
<tr>
<td></td>
<td>Field Value</td>
</tr>
<tr>
<td></td>
<td>SSJPJXOD</td>
</tr>
<tr>
<td></td>
<td>SSJPJXRS</td>
</tr>
<tr>
<td></td>
<td>SSJPUSER</td>
</tr>
</tbody>
</table>

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

**JESplex Information service, IAZPLEX contents:** For the JESplex Information service (function code SSJPJXOD), the caller must set the following fields in the IAZPLEX control block:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPLXID</td>
<td>Eyecatcher of the control block (set to 'JESPLEXI')</td>
</tr>
<tr>
<td>JPLXLEN</td>
<td>Length of the IAZPLEX (JPLXSZE) control block</td>
</tr>
<tr>
<td>JPLXVER</td>
<td>Input version of the IAZPLEX control block. Set to JPLXV010 for version 1 of the control block. Set to JPLXSVR# for the current (latest) version</td>
</tr>
</tbody>
</table>
SSI Function Code 82

**JPLXSTRP**  
Storage management anchor for use by the subsystem that responds to this request. It is expected that the caller will set this field to zero the first time IAZJPLEX is used and from that point on the field will be managed by the subsystem.

The caller can also set the following fields in the IAZJPLEX control block on input to limit (or select) which data will be returned. If no filters are specified, all data will be returned. If any filters are specified, at least one of the filter conditions in each of the separate filters must be matched before data will be returned.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPLXFLTR</td>
<td>Flag byte for filtering results. Each bit corresponds to a filter condition that must be matched before data is returned.</td>
</tr>
<tr>
<td>JPLXSTS1</td>
<td>Flag byte to filter on Status. Each bit corresponds to a filter condition. This filter will be matched if at least one of the specified filter conditions is met.</td>
</tr>
<tr>
<td>JPLXSPEC</td>
<td>Flag byte to filter on JES Specific values. Each bit corresponds to a filter condition. This filter will be matched if at least one of the specified filter conditions is met.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPLXFSNM</td>
<td>Filter on the MVS system name specified in the field JPLXSNAM</td>
</tr>
<tr>
<td>JPLXFNMN</td>
<td>Filter on the JES member name specified in the field JPLXMNAM</td>
</tr>
<tr>
<td>JPLDRAIN</td>
<td>Return JESPLEX information only if the system or member is in a Drained or Down state.</td>
</tr>
<tr>
<td>JPLINTZ</td>
<td>Return JESPLEX information only if the system or member is in an Initializing state (JES2 Only)</td>
</tr>
<tr>
<td>JPLXACTV</td>
<td>Return JESPLEX information only if the system or member is in an Active state</td>
</tr>
<tr>
<td>JPLXDRING</td>
<td>Return JESPLEX information only if the system or member is in a Draining state (JES2 Only)</td>
</tr>
<tr>
<td>JPLOUDRF</td>
<td>Omit undefined members (JES2 Only)</td>
</tr>
<tr>
<td>JPLNATCH</td>
<td>Return JESPLEX information only if the system or member is in a Not Attached state (JES3 Only)</td>
</tr>
<tr>
<td>JPLXINDP</td>
<td>Return JESPLEX information only if the system or member is Independent (JES2 Only)</td>
</tr>
</tbody>
</table>
SSI Function Code 82

| JPLXBOSS | Return JESPLEX information only if the system or member is BOSS (JES2 Only) |
| JPLXPRIM | Return JESPLEX information only if the system or member is the Primary Subsystem (JES2 Only) |
| JPLXGLOB | Return JESPLEX information only if the system is the Global system (JES3 Only) |
| JPLXLOCL | Return JESPLEX information only if the system is a Local system (JES3 Only) |

JPLXSNAM MVS System Name filter.
JPLXMNAM MVS Member Name filter.

Note: For JES3, the member name is the same as the system name.

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

For the JESplex Information service function code SSJPJXRS (release storage), the caller should not alter any fields in the IAZJPLEX control block returned on the last SSJPJXOD function call.

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 JESplex Information services request completed. Check the SSOBRETN field for specific function information.
SSRTNSUP (4) | The subsystem specified in the SSIBSSNM field does not support the JESplex Information services function call.
SSRTNTUP (8) | The subsystem specified in the SSIBSSNM field exists but is not active.
SSI Function Code 82

| 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
- SSJPRETN
- IAZJPLEX (JESplex Information service)

**SSOBRETN Contents:** When control returns to the caller and register 15 contains a zero, the JESplex Information services function places one of the following decimal values in the SSOBRETN field:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPOK (0)</td>
<td>Request successful.</td>
</tr>
<tr>
<td>SSJPERRW (4)</td>
<td>Request completed with possible errors. See SSJPRETN for reason code.</td>
</tr>
<tr>
<td>SSJPERRU (8)</td>
<td>Request cannot be completed because of user error. See SSJPRETN for reason code.</td>
</tr>
<tr>
<td>SSJPERRJ (12)</td>
<td>Request cannot be completed; SSJPRETN contains internal reason code.</td>
</tr>
<tr>
<td>SSJPPARM (16)</td>
<td>The parameter list, that is the SSJP extension is an invalid format - it is not an SSJP, the service version number is not supported, or the SSJP is not large enough.</td>
</tr>
<tr>
<td>SSJPSTOR (20)</td>
<td>Request cannot be processed because required storage cannot be obtained. No data can be returned to the caller.</td>
</tr>
</tbody>
</table>

**SSJPRETN Contents:** In addition to the return code in SSOBRETN, the field SSJPRETN contains the service related error or more specific information about the error. SSJPRETN can be set to one of the following values if SSOBRETN is not zero:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPUNSF (4)</td>
<td>Unsupported subfunction requested.</td>
</tr>
<tr>
<td>SSJPNNTDS (8)</td>
<td>SSJPUSER does not point to the correct control block.</td>
</tr>
<tr>
<td>SSJPUNSD (12)</td>
<td>Version number in the control block pointed to by SSJPUSER is not correct.</td>
</tr>
<tr>
<td>SSJPSMLE (16)</td>
<td>Length field in the control block pointed to by SSJPUSER is too small.</td>
</tr>
</tbody>
</table>
SSI Function Code 82

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPEYEE (20)</td>
<td>Eyecatcher in the control block pointed to by SSJUSER is not correct.</td>
</tr>
<tr>
<td>SSJPGTM (128)</td>
<td>$GETMAIN failed.</td>
</tr>
<tr>
<td>SSJPSTGO (132)</td>
<td>STORAGE OBTAIN failed.</td>
</tr>
<tr>
<td>SSJPINVA (136)</td>
<td>Invalid filter arguments were specified.</td>
</tr>
<tr>
<td>SSJPGLBL (140)</td>
<td>Function not supported on the global (JES3 only).</td>
</tr>
<tr>
<td>JPLXINVA (132)</td>
<td>Invalid search arguments</td>
</tr>
<tr>
<td>JPLXINV1 (136)</td>
<td>Status filter invalid</td>
</tr>
<tr>
<td>JPLXINV2 (140)</td>
<td>System name filter invalid</td>
</tr>
<tr>
<td>JPLXINV3 (144)</td>
<td>Member name filter invalid</td>
</tr>
<tr>
<td>JPLXINV4 (148)</td>
<td>JES Specific filter invalid</td>
</tr>
</tbody>
</table>

**JESplex Information service, IAZJPLEX contents:** For the JESplex Information service (function code SSJPJXOD), the following parameters are returned in IAZJPLEX:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPLXVERO</td>
<td>Subsystem version number (currently 1)</td>
</tr>
<tr>
<td>JPLXL PTR</td>
<td>Pointer to data for first Initiator data area.</td>
</tr>
<tr>
<td>JPLXNM BR</td>
<td>Number of member data areas returned.</td>
</tr>
<tr>
<td>JPLXTRKT</td>
<td>Total number of SPOOL tracks defined across all partitions</td>
</tr>
<tr>
<td>JPLXTRKU</td>
<td>Total number of SPOOL tracks used across all partitions</td>
</tr>
</tbody>
</table>

For each system or member that passes the filter requirements, an element is added to the chain pointed to by JPLXL PTR. Each element is composed of the following sections:

<table>
<thead>
<tr>
<th>DSECT Name</th>
<th>DSECT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPXHDR</td>
<td>JESplex Header Section</td>
</tr>
<tr>
<td>JPXPREF</td>
<td>JESplex Prefix Section</td>
</tr>
<tr>
<td>JPXGENI</td>
<td>JESplex General Information Section</td>
</tr>
<tr>
<td>JPXJES3I</td>
<td>JESplex JES3 Specific Information Section</td>
</tr>
<tr>
<td>JPXJES2I</td>
<td>JESplex JES2 Specific Information Section</td>
</tr>
<tr>
<td>JPXCPRF</td>
<td>JESplex Command Prefix Information</td>
</tr>
<tr>
<td>JPXCPRFE</td>
<td>JESplex Command Prefix Array Entry</td>
</tr>
</tbody>
</table>

The following example is a layout of the various sections of the JESplex Information output data area. The basic layout is a chain of JESplex Information sections. Each JESplex section has a common section and a subsystem specific section.
SSI Function Code 82

**JESplex Header Section:** Each JESplex information element begins with a header section. There can be multiple JESPLEX Areas returned. The JPXNXTP pointer is used to navigate to the next JESplex Header section in the chain.

The fields in the JPXHDR section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPXEYE</td>
<td>Eyecatcher. Should be set to ‘JPXHDR’.</td>
</tr>
<tr>
<td>JPXOPRF</td>
<td>Offset to the prefix section</td>
</tr>
<tr>
<td>JPXNXTP</td>
<td>Address of the next JESplex information element</td>
</tr>
</tbody>
</table>

**JESplex Prefix Section:** This section holds the length of all the information reported for one of the JESplex members. This length does not include the length of the JESplex Header section. For addressability to this section, add the JPXOPRF header field to the header (JPXHDR) address.

The fields in the JPXPREF section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPXPRLN</td>
<td>Length of the entire element, not including the JESplex header section</td>
</tr>
<tr>
<td>JPXPRTYP</td>
<td>Type of this section.</td>
</tr>
<tr>
<td>JPXPRMD</td>
<td>Modifier for this section.</td>
</tr>
</tbody>
</table>

**JESplex General Information Section:** This section holds details that are common to both JES2 and JES3. To get addressability to this section, add the JPXOPRF header field and the prefix size (JPXPRLN) to the header (JPXHDR) address.

The fields in the JPXGENI section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPXGLN</td>
<td>Length of this section.</td>
</tr>
<tr>
<td>JPXGTY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>JPXGMD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>JPSBSNM</td>
<td>Subsystem name</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td>JPXTIME</td>
<td>Last start date and time in STCK format</td>
</tr>
<tr>
<td>JPSSTAT1</td>
<td>Member status</td>
</tr>
</tbody>
</table>

**Field** | **Value** | **Description** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSDRAIN</td>
<td>Drained or Down member</td>
<td></td>
</tr>
<tr>
<td>JPSINITZ</td>
<td>Initializing (JES2 Only)</td>
<td></td>
</tr>
<tr>
<td>JPSACTIV</td>
<td>Active member</td>
<td></td>
</tr>
<tr>
<td>JPSDRING</td>
<td>Draining member (JES2 Only)</td>
<td></td>
</tr>
<tr>
<td>JPSNATCH</td>
<td>Not Attached (JES3 Only)</td>
<td></td>
</tr>
</tbody>
</table>

| JPSSTATC | Character string representation of the member status |

<table>
<thead>
<tr>
<th>JPMVSNM</th>
<th>MVS system name</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSMEMNM</td>
<td>JES member name</td>
</tr>
<tr>
<td>JPSVERSN</td>
<td>Product version in character format</td>
</tr>
<tr>
<td>JPSMFDI</td>
<td>SMF identifier</td>
</tr>
<tr>
<td>JPSYSLG</td>
<td>Syslog indicator flag</td>
</tr>
</tbody>
</table>

**Field** | **Value** | **Description** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSLOGY</td>
<td>Release 11 syslog support is Active for this member</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JPMEMNO</th>
<th>Member number (JES2 Only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSSTPE</td>
<td>Type of last start</td>
</tr>
</tbody>
</table>

**Field** | **Value** | **Description** |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSCOLD</td>
<td>Cold start</td>
<td></td>
</tr>
<tr>
<td>JPS2COLF</td>
<td>Cold start with format (JES2 Only)</td>
<td></td>
</tr>
<tr>
<td>JPSWARM</td>
<td>Warm start</td>
<td></td>
</tr>
<tr>
<td>JPS2SRMS</td>
<td>Single member warm start (JES2 Only)</td>
<td></td>
</tr>
<tr>
<td>JPS3WRMD</td>
<td>Warm start to replace a spool dataset (JES3 Only)</td>
<td></td>
</tr>
<tr>
<td>JPS3WRMA</td>
<td>Warm start with analysis (JES3 Only)</td>
<td></td>
</tr>
<tr>
<td>JPS3WDA</td>
<td>Warm start to replace a spool dataset with analysis (JES3 Only)</td>
<td></td>
</tr>
<tr>
<td>JPSHOT</td>
<td>Hot start</td>
<td></td>
</tr>
<tr>
<td>JPS3HOTR</td>
<td>Hot start with refresh (JES3 Only)</td>
<td></td>
</tr>
<tr>
<td>JPS3HOTA</td>
<td>Hot start with analysis (JES3 Only)</td>
<td></td>
</tr>
<tr>
<td>JPS3HTRA</td>
<td>Hot start with refresh and analysis (JES3 Only)</td>
<td></td>
</tr>
<tr>
<td>JPS2QUICK</td>
<td>Quick start (JES2 Only)</td>
<td></td>
</tr>
<tr>
<td>JPS3LCL</td>
<td>Local start (JES3 Only)</td>
<td></td>
</tr>
</tbody>
</table>
SSI Function Code 82

<table>
<thead>
<tr>
<th>JPXPRODL</th>
<th>Product level in binary format</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPXSERVL</td>
<td>Service level in binary format</td>
</tr>
</tbody>
</table>

**JESplex JES3 Specific Information Section:** This section holds details that are specific to JES3. To get addressability to this section, add the prefix offset (JPXOPRF), the prefix size (JPXPRSZ), and the general information section size (JPXGLN) to the header (JPXHDR) address.

The fields in the JPXJES3I section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX3LN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JPX3TY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>JPX3MD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>JPX3GCON</td>
<td>Last global contact time in STCK format</td>
</tr>
<tr>
<td>JPX3TRK1</td>
<td>Primary track group allocation</td>
</tr>
<tr>
<td>JPX3TRK2</td>
<td>Secondary track group allocation</td>
</tr>
<tr>
<td>JPX3WTOL</td>
<td>WTO message limit</td>
</tr>
<tr>
<td>JPX3WTOI</td>
<td>WTO message interval in seconds</td>
</tr>
<tr>
<td>JPX3CSA</td>
<td>PBUF CSA limit</td>
</tr>
<tr>
<td>JPX3AUX</td>
<td>PBUF JES3AUX limit</td>
</tr>
<tr>
<td>JPX3FIX</td>
<td>Fixed PBUFs</td>
</tr>
<tr>
<td>JPX3USR</td>
<td>User pages per open SYSOUT dataset</td>
</tr>
<tr>
<td>JPX3SELM</td>
<td>Selection mode constant</td>
</tr>
<tr>
<td>JPX3SP1</td>
<td>Spool partition name</td>
</tr>
<tr>
<td>JPX3MPFX</td>
<td>Message prefix</td>
</tr>
<tr>
<td>JPX3MDST</td>
<td>Message destination</td>
</tr>
<tr>
<td>JPX3FLG1</td>
<td>Flag byte</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX3GBL</td>
<td>Global node</td>
<td></td>
</tr>
<tr>
<td>JPX3ONL</td>
<td>Online</td>
<td></td>
</tr>
<tr>
<td>JPX3FLSH</td>
<td>Flushed</td>
<td></td>
</tr>
<tr>
<td>JPX3CNN</td>
<td>Connected</td>
<td></td>
</tr>
<tr>
<td>JPX3NCNN</td>
<td>Not connected</td>
<td></td>
</tr>
<tr>
<td>JPX3DOWN</td>
<td>Down</td>
<td></td>
</tr>
<tr>
<td>JPX3ATT</td>
<td>Attached</td>
<td></td>
</tr>
<tr>
<td>JPX3NATT</td>
<td>Not attached</td>
<td></td>
</tr>
</tbody>
</table>

**JESplex JES2 Specific Information Section:** This section holds details that are specific to JES2. To get addressability to this section, add the prefix offset (JPXOPRF), the prefix size (JPXPRSZ), and the general information section size (JPXGLN) to the header (JPXHDR) address.
The fields in the JPXJES2I section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX2LN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JPX2TY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>JPX2MD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>JPX2FLG1</td>
<td>JES2 indicators</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX21IND</td>
<td>Independent mode</td>
<td></td>
</tr>
<tr>
<td>JPX21BOS</td>
<td>BOSS indicator</td>
<td></td>
</tr>
<tr>
<td>JPX21PRI</td>
<td>Primary subsystem indicator</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX2ITIM</td>
<td>Time of last checkpoint access</td>
</tr>
<tr>
<td>JPX2FLG2</td>
<td>Current command being processed indicator</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX21P</td>
<td>$P command</td>
<td></td>
</tr>
<tr>
<td>JPX21PXQ</td>
<td>$PXEQ command</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX2HOLD</td>
<td>Current setting for MASDEF HOLD</td>
</tr>
<tr>
<td>JPX2MIND</td>
<td>Current setting for MASDEF MIN DORMANCY</td>
</tr>
<tr>
<td>JPX2MAXD</td>
<td>Current setting for MASDEF MAX DORMANCY</td>
</tr>
<tr>
<td>JPX2SYNC</td>
<td>Current setting for MASDEF SYNCTOL</td>
</tr>
<tr>
<td>JPX2AHLD</td>
<td>Actual HOLD value on the last checkpoint</td>
</tr>
<tr>
<td>JPX2ADRM</td>
<td>Actual DORMANCY value on the last checkpoint</td>
</tr>
<tr>
<td>JPX2RSID</td>
<td>Name of the member doing the reset</td>
</tr>
<tr>
<td>JPX2STAT</td>
<td>Specific Status Indictor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX2DOWN</td>
<td>DOWN</td>
<td></td>
</tr>
<tr>
<td>JPX2DEF</td>
<td>DEFINED</td>
<td></td>
</tr>
<tr>
<td>JPX2INU</td>
<td>INUSE</td>
<td></td>
</tr>
<tr>
<td>JPX2FAIL</td>
<td>FAILED</td>
<td></td>
</tr>
<tr>
<td>JPX2UNDF</td>
<td>Member UNDEFINED</td>
<td></td>
</tr>
<tr>
<td>JPX2UPND</td>
<td>Member UNDEFINED-PENDING</td>
<td></td>
</tr>
<tr>
<td>JPX2ACTV</td>
<td>Member ACTIVE</td>
<td></td>
</tr>
<tr>
<td>JPX2INAC</td>
<td>Member TERMINATED</td>
<td></td>
</tr>
<tr>
<td>JPX2INIT</td>
<td>Member INITIALIZING</td>
<td></td>
</tr>
<tr>
<td>JPX2TERM</td>
<td>Member TERMINATING</td>
<td></td>
</tr>
<tr>
<td>JPX2JESF</td>
<td>Member JES2-FAILED</td>
<td></td>
</tr>
<tr>
<td>JPX2XCFF</td>
<td>Member JESXCF-FAILED</td>
<td></td>
</tr>
<tr>
<td>JPX2MVSG</td>
<td>Member MVS-GONE</td>
<td></td>
</tr>
</tbody>
</table>
**SII Function Code 82**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPX2DORM</td>
<td>Member DORMANT</td>
</tr>
<tr>
<td>JPX2DRAN</td>
<td>Member DRAINED</td>
</tr>
<tr>
<td>JPX2ALIC</td>
<td>Member awaiting ALICE</td>
</tr>
</tbody>
</table>

**JESplex Command Prefix Information:** This section acts as a header to the array of command prefix entries.

This section follows immediately after the JES2 or JES3 specific section depending on whether JES2 or JES3 implementation.

This is a variable length section. The length depends upon the number of command prefix entries. JES2 will have only one entry. JES3 might have up to 14 entries.

The fields in the JPXCPREF section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPXCLN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>JPXCTY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>JPXCMD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>JPXPRXC</td>
<td>Count of command prefix array entries</td>
</tr>
<tr>
<td>JPXFUXL</td>
<td>Length of each command prefix array entry</td>
</tr>
<tr>
<td>JXPFXO</td>
<td>Offset to the first command prefix array entry</td>
</tr>
</tbody>
</table>

**JESplex Command Prefix Array Entry:** This section maps out the individual entries in the command prefix array. To get addressability to this array, add the prefix offset (JXPXOPRF), the prefix size (JXPXRPSZ), the general information section size (JXPXGENSZ), the sizes of any optional sections supplied, and the array offset (JXPXO) to the header (JPXHDR) address.

The fields in the JPXCPFE section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPXCPFXS</td>
<td>Scope Flags</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPXCSYSP</td>
<td>SYSPLEX Scope</td>
</tr>
<tr>
<td>JPXCSYST</td>
<td>System Scope</td>
</tr>
</tbody>
</table>

**Command Prefix value**

**Job Class Information**

The Job Class Information service provides information about the attributes of JES job classes. Information can be obtained on all JES job classes or filters can be supplied to limit which job classes are returned. Information is returned as a chained list of data areas, each representing a JES job class.

See the following sections for more information about Job Class Information:

- "Type of Request" on page 275
- "Use Information" on page 275
- "Issued to" on page 275
- "Related SSI Codes" on page 275
Use Information: To use the JES property information services SSI, callers must first decide the function they want to perform. The appropriate parameter list must be obtained and pointed to by SSJUSER.

Issued to: A JES subsystem (either primary or secondary). The subsystem does not have to be associated with the requesting address space.

Related SSI Codes: 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:
- IEFSSOBH
- IEFJSSIB
- IAZSSJP
- IAZJPCLS (Job Class Information)

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, IAZSSJP, and IAZJPCLS, control blocks can be in 24- or 31-bit virtual storage
Recovery: The caller should provide an ESTAE-type recovery environment. See z/OS MVS Programming: Assembler Services Guide for more information about an ESTAE-type recovery environment.

Figure 27 on page 276 shows the environment at the time of the call for SSI function code 82, Job Class Information Subfunction.
### Input Register Information:
Before issuing the IEFSSREQ macro, the caller must ensure that the following general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area.</td>
</tr>
</tbody>
</table>

### Input Parameters:
Input parameters for the function routine are:
- SSOB
- SSIB
- IAZSSJP
- IAZJPCLS (Job Class Information)

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 82 (SSOBSSJP)</td>
</tr>
</tbody>
</table>

---

**Figure 27. Environment at Time of Call for SSI Function Code 82, Job Class Information Subfunction.**
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 about the life-of-job SSIB.

SSOBINDV  Address of the function-dependent area (IAZSSJP 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name: name of the subsystem to which this JESplex Information Services request is directed.</td>
</tr>
</tbody>
</table>

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

**IAZSSJP Contents:** The caller must set the following fields in the IAZSSJP control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPID</td>
<td>Eyecatcher for the control block (set to 'SSJP')</td>
</tr>
<tr>
<td>SSJPLEN</td>
<td>Length of the IAZSSJP (SSJPSIZE) control block</td>
</tr>
<tr>
<td>SSJPVER</td>
<td>Input version of the IAZSSJP control block. Set to SSJPVER1 for version 1 of the control block or to SSJPVERC for the current version of the control block.</td>
</tr>
<tr>
<td>SSJPFREQ</td>
<td>Function to be performed on this request. Valid functions and their related SSJPUSER area are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPJCOD</td>
<td>IAZJPCLS Job Class Information service, obtain data</td>
</tr>
<tr>
<td>SSJPJCRS</td>
<td>IAZJPCLS Job Class Information service, release storage</td>
</tr>
<tr>
<td>SSJPUSER</td>
<td>Pointer to service specific data area '(IAZJPCLS)'</td>
</tr>
</tbody>
</table>

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

**Job Class Information service, IAZJPCLS contents:** For the Job Class Information service (function code SSJPJCOD), the caller must set the following fields in the IAZJPCLS control block:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPCLID</td>
<td>Eyecatcher of the control block (set to 'JPCLASSD')</td>
</tr>
<tr>
<td>JPCLLEN</td>
<td>Length of the IAZJPCLS (JPCLSZE) control block</td>
</tr>
</tbody>
</table>
| JPCLVER    | Input version of the IAZJPCLS control block. Set to
The caller can also set the following fields in the IAZJPCLS control block on input to limit (or select) which data will be returned.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPCLFLG1</td>
<td>Flag byte which describes which filters to use to limit or select the data to be returned. Each bit corresponds to a filter that must be matched before data is returned.</td>
</tr>
</tbody>
</table>

Bit Value | Description
----------|-------------
JPCL1CLS | Return job class information for those job classes that match the class name indicated by JPCLCNAM.

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

For the Job Class Information service function code SSJPJCRS (release storage), the caller should not alter any fields in the IAZJPCLS control block returned on the last SSJPJCOD function call.

**Output Register Information:** When control returns to the caller, the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>2 - 13</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSRTOK (0)</td>
<td>The Job Class Information services request completed. Check the SSOBRETN field for specific function information.</td>
</tr>
<tr>
<td>SSRTNSUP (4)</td>
<td>The subsystem specified in the SSIBSSNM field does not support the Job Class Information services function call.</td>
</tr>
</tbody>
</table>
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
- SSJPRETN
- IAZJPCLS (Job Class Information service)

**SSOBRETN Contents:** When control returns to the caller and register 15 contains a zero, the Job Class Information services function places one of the following decimal values in the SSOBRETN field:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPOK (0)</td>
<td>Request successful.</td>
</tr>
<tr>
<td>SSJPERRW (4)</td>
<td>Request completed with possible errors. See SSJPRETN for reason code.</td>
</tr>
<tr>
<td>SSJPERRU (8)</td>
<td>Request cannot be completed because of user error. See SSJPRETN for reason code.</td>
</tr>
<tr>
<td>SSJPERRJ (12)</td>
<td>Request cannot be completed; SSJPRETN contains internal reason code.</td>
</tr>
<tr>
<td>SSJPPARM (16)</td>
<td>The parameter list, that is the SSJP extension is an invalid format</td>
</tr>
<tr>
<td></td>
<td>- It is not an SSJP</td>
</tr>
<tr>
<td></td>
<td>- The service version number is not supported</td>
</tr>
<tr>
<td></td>
<td>- The SSJP is not large enough</td>
</tr>
<tr>
<td>SSJPSTOR (20)</td>
<td>Request cannot be processed because required storage cannot be obtained. No data can be returned to the caller.</td>
</tr>
</tbody>
</table>

**SSJPRETN Contents:** In addition to the return code in SSOBRETN, the field SSJPRETN contains the service related error or more specific information about the error. SSJPRETN can be set to one of the following values if SSOBRETN is not zero:

<table>
<thead>
<tr>
<th>Value (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSJPUNSF (4)</td>
<td>Unsupported subfunction requested.</td>
</tr>
<tr>
<td>SSJPNTDS (8)</td>
<td>SSJPUSER does not point to the correct control block.</td>
</tr>
<tr>
<td>SSJPUNSD (12)</td>
<td>Version number in the control block pointed to by SSJPUSER is not correct.</td>
</tr>
</tbody>
</table>
SSI Function Code 82

Length field in the control block pointed to by SSJPUSER is too small.

Eyecatcher in the control block pointed to by SSJPUSER is not correct.

$GETMAIN failed.

STORAGE OBTAIN failed.

**Job Class Information service, IAZJPCLS contents:** For the Job Class Information service (function code SSJPJCOD), the following parameters are returned in IAZJPCLS:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPCLVERO</td>
<td>Subsystem version number (currently 1)</td>
</tr>
<tr>
<td>JPCLDPTR</td>
<td>Pointer to data for first Initiator data area.</td>
</tr>
<tr>
<td>JPCLNCLS</td>
<td>Number of member data areas returned.</td>
</tr>
</tbody>
</table>

For each job class that passes the filter requirements, an element is added to the chain pointed to by JPCLDPTR. Each element is composed of the following sections:

**DSECT Name** | **DSECT Description**
--- | ---
CLSHDR | Job Class Header Section
CLSPREF | Job Class Prefix Section
CLSGENI | Job Class General Information Section

In addition to the common sections listed above, JES2 returns these additional job class sections:

**DSECT Name** | **DSECT Description**
--- | ---
CLSJES2I | JES2 Job Class Information Section

In addition to the common sections listed above, JES3 returns these additional job class sections:

**DSECT Name** | **DSECT Description**
--- | ---
CLSJES3I | JES3 Job Class Information will contain zero or more of the JES3 TLIMIT Entry entries.
CLS3TLIM | JES3 TLIMIT Entry

Each job class has a chain of zero or more member entries. The first member is pointed to by field CLSFRSTM in the Job Class Header section. Each member element is composed of the following sections:

**DSECT Name** | **DSECT Description**
--- | ---
CLMHDR | Job Class Member Header Section
CLMPREF | Job Class Member Prefix Section
CLMGENI | Job Class Member General Information Section

JES3 returns these additional job class member sections:

**DSECT Name** | **DSECT Description**
--- | ---

280 z/OS V1R11.0 MVS Using the Subsystem Interface
CLMJES3I  JES3 Job Class Member Information will contain zero or more of the JES3 MLIMIT entries.

CLM3MLIM  JES3 MLIMIT Entry

The following is a layout of the various sections of the Job Class Information output data area.

```
CLASS INFO SECTION
+----------------+
| CLSHDR | CLSNXTP ------------> POINTER TO THE NEXT CLSHDR IN THE CHAIN. |
| | CLSFRTSTM ------------> ZERO IF END OF CHAIN. |
+----------------+

Prefix Section

General Class Info Section

Optional JES2 Class Info Section

Optional JES3 Class Info Section

Optional JES3 TLIMIT Array Elements

NOTE: Included if JES2

NOTE: Included if JES3

NOTE: Number of elements specified in CLS3TLCT.

CLASS MEMBER INFO SECTION
+----------------+
| CLMHDR | CLMNXTP ------------> POINTER TO THE NEXT CLMHDR IN THE CHAIN. |
| | CLMPRTM ------------> ZERO IF END OF CHAIN. |
+----------------+

Prefix Section

General Member Info Section

Optional JES3 Member Info Section

Optional JES3 MLIMIT Array Elements

NOTE: Included if JES3

NOTE: Number of elements specified in CLM3MLCT.
**Job Class Header Section:** Each Job Class information element begins with a Header which holds an eyecatcher, an offset to the prefix section, a pointer to the next job class, and a pointer to the member section, which is a chain of the members that pertain to this job class.

The fields in the CLSDHR section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLSID</td>
<td>Eyecatcher. Should be set to 'CLASSHDR'.</td>
</tr>
<tr>
<td>CLSOPRF</td>
<td>Offset to prefix section.</td>
</tr>
<tr>
<td>CLSNXTP</td>
<td>Address of the next Job Class Information element.</td>
</tr>
<tr>
<td>CLSFRSTM</td>
<td>Address of the first Member section for this job class.</td>
</tr>
</tbody>
</table>

**Job Class Prefix Section:** This section holds the length of all the information reported for this job class. This length does not include the length of the Job Class Header section. This length does include all storage needed to report the General and JES2 or JES3 Job Class information.

The fields in the CLSPREF section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLSPRDLN</td>
<td>Length of the entire element, not including the Job Class Header.</td>
</tr>
<tr>
<td>CLSPRTTP</td>
<td>Type of this section.</td>
</tr>
<tr>
<td>CLSPRMD</td>
<td>Modifier for this section.</td>
</tr>
</tbody>
</table>

**Job Class General Information Section:** This section contains job class attributes that are common for JES2 and JES3.

The fields in the CLSGENI section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLSGLEN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>CLSGCTY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>CLSGMD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>CLSNAME</td>
<td>Class name</td>
</tr>
<tr>
<td>CLSGFLG1</td>
<td>Class flag 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLSG1WLM</td>
<td>Class is in WLM mode</td>
<td></td>
</tr>
<tr>
<td>CLS1JRN</td>
<td>No journal option</td>
<td></td>
</tr>
</tbody>
</table>

**CLSGREST**

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLSGRCAN</td>
<td>Print output, then cancel the job (JES3 only)</td>
</tr>
<tr>
<td>CLSGRHLD</td>
<td>Hold the job (JES3 only)</td>
</tr>
</tbody>
</table>
SSI Function Code 82

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLGRTPT</td>
<td>Print output, then hold the job (JES3 only)</td>
</tr>
<tr>
<td>CLGSTRRT</td>
<td>Allow warmstart to re-queue to Execution Phase</td>
</tr>
</tbody>
</table>

| CLGJFLG     | JESLOG default settings |
| Bit Value   | Description |
| CLGEI      | Spin eligible |
| CLGTM      | Spin on time interval |
| CLGTMID    | Spin on time of day |
| CLGLINE    | Spin upon line delta |
| CLGSUP     | Suppress |
| CLGNOSEP   | No spin |

| CLGJVAL     | Spin value. This is the number of seconds in the interval if CLGJFLG is set to CLGSTM. This is the number of seconds past midnight if CLGJFLG is set to CLGTMID. This is the number of lines if CLGJFLG is set to CLGLINE. |

| CLGMAXJ     | Maximum number of concurrently executing jobs of this class (TDEPTH for JES3 if specified). |
| CLGCURJ     | Current number of concurrently executing jobs of this class. |
| CLGQSIZ     | Number of jobs of this class that are eligible for execution (awaiting job selection) (JES2 only). |
| CLGHELD     | Number of jobs of this class that are held (JES2 only). |

JES2 Job Class Information Section: This section contains job class attributes that are unique to JES2.

The fields in the CLSJES2I section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS2LN</td>
<td>Length of this section</td>
</tr>
<tr>
<td>CLS2TY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>CLS2MD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>CLS2JBFL</td>
<td>Job class flag</td>
</tr>
</tbody>
</table>

| Bit Value   | Description |
| CLS2BCH      | Batch job |
| CLS2TSU      | Time sharing user |
| CLS2STC      | Started task |
| CLS2NOUT     | No output option |

| CLS2TYPR     | TYPRUN setting |

| Bit Value   | Description |
### SSI Function Code 82

**CLS2CACT**

Accounting information

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS2CSWA</td>
<td></td>
<td>SWA above 16M line</td>
</tr>
<tr>
<td>CLS2CNUM</td>
<td></td>
<td>Account number required</td>
</tr>
<tr>
<td>CLS2CNAM</td>
<td></td>
<td>Programmer name required</td>
</tr>
<tr>
<td>CLS2CNON</td>
<td></td>
<td>No information required</td>
</tr>
<tr>
<td>CLS2CALL</td>
<td></td>
<td>Account number and programmer name required</td>
</tr>
</tbody>
</table>

**CLS2CTIM**

Default for job time limit

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS2CMNT</td>
<td>Maximum minutes</td>
</tr>
<tr>
<td>CLS2CSEC</td>
<td>Maximum seconds</td>
</tr>
</tbody>
</table>

**CLS2CREG**

Default for job step region

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS2CRGN</td>
<td>Numeric specification</td>
</tr>
<tr>
<td>CLS2CRGA</td>
<td>Kilobyte or megabyte specification</td>
</tr>
</tbody>
</table>

**CLS2CMND**

Command disposition

<table>
<thead>
<tr>
<th>Field Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS2CEXE</td>
<td>Pass the command through</td>
</tr>
<tr>
<td>CLS2CDSP</td>
<td>Display and then pass the command</td>
</tr>
<tr>
<td>CLS2CVER</td>
<td>Ask operator disposition</td>
</tr>
<tr>
<td>CLS2CIGN</td>
<td>Ignore the command</td>
</tr>
</tbody>
</table>

**CLS2CBLP**

Bypass label processing

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS2CBLY</td>
<td>Process bypass label parameter</td>
</tr>
</tbody>
</table>

**CLS2COCG**

Operator command group

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS2CGSY</td>
<td>Group 1 commands (SYS)</td>
</tr>
<tr>
<td>CLS2CGIO</td>
<td>Group 2 commands (I/O)</td>
</tr>
<tr>
<td>CLS2CGCO</td>
<td>Group 3 commands (CONS)</td>
</tr>
<tr>
<td>CLS2CGAL</td>
<td>All command groups</td>
</tr>
</tbody>
</table>

**CLS2CJCL**

Default MSGLEVEL, JCL listing if not MSGLEVEL

<table>
<thead>
<tr>
<th>CLS2CMMSG</th>
<th>Allocation termination messages value of MSGLEVEL</th>
</tr>
</thead>
</table>

---

<table>
<thead>
<tr>
<th>function</th>
<th>code</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>82</td>
<td>284</td>
<td>z/OS V1R11.0 MVS Using the Subsystem Interface</td>
</tr>
<tr>
<td>Bit</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>CLS2JOPT</td>
<td></td>
<td>Job options flag</td>
</tr>
<tr>
<td>CLS2NLOG</td>
<td>No joblog indicator</td>
<td></td>
</tr>
<tr>
<td>CLS2XBM</td>
<td>XBM II job class</td>
<td></td>
</tr>
<tr>
<td>CLS2QHLD</td>
<td>Class queue is held</td>
<td></td>
</tr>
<tr>
<td>CLS2XBM</td>
<td>Procedure name for XBM II jobs</td>
<td></td>
</tr>
<tr>
<td>CLS2PRCN</td>
<td>Procedure library number</td>
<td></td>
</tr>
<tr>
<td>CLS2SMF</td>
<td>SMF flags</td>
<td></td>
</tr>
<tr>
<td>CLS2NUSO</td>
<td>Do not take IEFUSO exit</td>
<td></td>
</tr>
<tr>
<td>CLS2NTY6</td>
<td>Do not produce Type 6 SMF record</td>
<td></td>
</tr>
<tr>
<td>CLS2NUJP</td>
<td>Do not take IEFUJP exit</td>
<td></td>
</tr>
<tr>
<td>CLS2NT26</td>
<td>Do not produce Type 26 SMF record</td>
<td></td>
</tr>
<tr>
<td>CLS2PERF</td>
<td>Default performance group</td>
<td></td>
</tr>
<tr>
<td>CLS2DMCL</td>
<td>Default message class, TSU and STC classes only</td>
<td></td>
</tr>
<tr>
<td>CLS2FLG1</td>
<td>Normal output disposition for JES data sets</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>CLS21CDP</td>
<td>Conditionally purge output for jobs in this class</td>
<td></td>
</tr>
<tr>
<td>CLS21NOK</td>
<td>ABNORMAL OUTDISP=KEEP</td>
<td></td>
</tr>
<tr>
<td>CLS21NOL</td>
<td>ABNORMAL OUTDISP=LEAVE</td>
<td></td>
</tr>
<tr>
<td>CLS2FLG2</td>
<td>Abnormal output disposition for JES data sets</td>
<td></td>
</tr>
<tr>
<td>Bit</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>CLS23SPC</td>
<td>Special class (STC/TSU)</td>
<td></td>
</tr>
<tr>
<td>CLS23SNV</td>
<td>Default SCHENV (CLS2SCHE) no longer defined to WLM</td>
<td></td>
</tr>
<tr>
<td>CLS23DOK</td>
<td>Duplicate job names OK for this job class</td>
<td></td>
</tr>
</tbody>
</table>

"Chapter 3. SSI Function Codes Your Program Can Request 285"
**SSI Function Code 82**

**JES3 Job Class Information Section:** This section contains job class attributes that are unique to JES3.

The fields in the CLSJES3I section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS3LN</td>
<td>Length of this section, including the variable length TLIMIT information.</td>
</tr>
<tr>
<td>CLS3TY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>CLS3MD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>CLS3GRP</td>
<td>Job class group name</td>
</tr>
<tr>
<td>CLS3PART</td>
<td>Spool partition name</td>
</tr>
<tr>
<td>CLS3TRK1</td>
<td>Primary track group allocation</td>
</tr>
<tr>
<td>CLS3TRK2</td>
<td>Secondary track group allocation</td>
</tr>
<tr>
<td>CLS3SDEP</td>
<td>SDEPTH setting</td>
</tr>
<tr>
<td>CLS3PTY</td>
<td>JES3 priority</td>
</tr>
<tr>
<td>CLS3FLG1</td>
<td>Flag byte</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS31DEF</td>
<td></td>
<td>Default class</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS3JOPT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS3NLOG</td>
<td></td>
<td>Suppress JESMSG</td>
</tr>
<tr>
<td>CLS3LOG</td>
<td></td>
<td>Log JESMSG</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS3TLOF</td>
<td>Offset to first JES3 TLIMIT entry</td>
</tr>
<tr>
<td>CLS3TLCT</td>
<td>JES3 TLIMIT entry count</td>
</tr>
<tr>
<td>CLS3TLSI</td>
<td>Size of a JES3 TLIMIT entry</td>
</tr>
</tbody>
</table>

**JES3 TLIMIT Entry:** This section contains the JES3 TLIMIT information for the job class.

The fields in the CLS3TLIM section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS3TCLS</td>
<td>Controlling class name</td>
</tr>
<tr>
<td>CLS3TMAX</td>
<td>Maximum jobs in controlling class</td>
</tr>
<tr>
<td>CLS3TCUR</td>
<td>Current jobs in controlling class</td>
</tr>
</tbody>
</table>

**Job Class Member Header Section:** Each Job Class member information element begins with a Header which holds an eyecatcher, an offset to the prefix section, and a pointer to the next member section.

The fields in the CLMDHR section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
CLMID Eyecatcher. Should be set to ‘CLASSMBR’.
CLMOPRF Offset to prefix section.
CLMNXTM Address of the next Job Class Member Information element.

**Job Class Member Prefix Section:** This section holds the length of all the information reported for this member. This length does not include the length of the Job Class Member Header section. This length does include all storage needed to report the General and JES3 Job Class Member information.

The fields in the CLMPREF section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLMPRLN</td>
<td>Length of the entire element, not including the length of the Job Class Member Header.</td>
</tr>
<tr>
<td>CLMPRTYP</td>
<td>Type of this section.</td>
</tr>
<tr>
<td>CLMPRMOD</td>
<td>Modifier for this section.</td>
</tr>
</tbody>
</table>

**Job Class Member General Information Section:** This section contains the member information that is common between JES2 and JES3.

The fields in the CLMGENI section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLMGLEN</td>
<td>Length of this section.</td>
</tr>
<tr>
<td>CLMGTY</td>
<td>Type of this section.</td>
</tr>
<tr>
<td>CLMGMD</td>
<td>Modifier for this section.</td>
</tr>
<tr>
<td>CLMGMNAM</td>
<td>Member name</td>
</tr>
<tr>
<td>CLMGSNAM</td>
<td>MVS System name</td>
</tr>
<tr>
<td>CLMFLG1</td>
<td>Flag byte</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLMG1ENB</td>
<td>Class is enabled or active on the member</td>
</tr>
<tr>
<td>CLMG1ACT</td>
<td>Member is active</td>
</tr>
<tr>
<td>CLMG1PXQ</td>
<td>Class is on halted member, $PXEQ issued (JES2 only)</td>
</tr>
<tr>
<td>CLMG1DRN</td>
<td>Class is on draining member (JES2 only)</td>
</tr>
<tr>
<td>CLMG1DEF</td>
<td>Class is defined on member (JES3 only)</td>
</tr>
<tr>
<td>CLMGJMAX</td>
<td>Maximum job count for this class on member (MDEPTH for JES3 if specified)</td>
</tr>
<tr>
<td>CLMGJCUR</td>
<td>Current active job count for this class on member</td>
</tr>
</tbody>
</table>

**JES3 Job Class Member Information Section:** This section contains the member information that is unique to JES3.
The fields in the CLMJES3I section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLM3LN</td>
<td>Length of this section, including the variable length MLI</td>
</tr>
<tr>
<td>CLM3TY</td>
<td>Type of this section</td>
</tr>
<tr>
<td>CLM3MD</td>
<td>Modifier for this section</td>
</tr>
<tr>
<td>CLM3SELM</td>
<td>Selection mode name</td>
</tr>
<tr>
<td>CLM3MLOF</td>
<td>Offset to first JES3 MLIMIT entry</td>
</tr>
<tr>
<td>CLM3MLCT</td>
<td>JES3 MLIMIT entry count</td>
</tr>
<tr>
<td>CLM3MLSI</td>
<td>Size of a JES3 MLIMIT entry</td>
</tr>
</tbody>
</table>

JES3 MLIMIT Entry: This section contains the JES3 MLIMIT information for the member.

The fields in the CLM3MLIM section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLM3MCLS</td>
<td>Controlling class name</td>
</tr>
<tr>
<td>CLM3MMAX</td>
<td>Maximum jobs in controlling class</td>
</tr>
<tr>
<td>CLM3MCUR</td>
<td>Current jobs in controlling class</td>
</tr>
</tbody>
</table>

System Information
Some of the JES property information services return system information that is mapped by the IAZJPLXI macro. This information is composed of the following sections:

DSECT Name DSECT Description
JPSYSPRF System Information Prefix Section
JPSYSINF JES System Information Section. This section contains one or more of the following entries:
JPSYSIFE System Information Entry

System Information Prefix Section: This section holds the length of all the information reported for the systems.

The fields in the JPSYSPRF section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSYXLNG</td>
<td>Length of all the sections.</td>
</tr>
<tr>
<td>JPSYXTYP</td>
<td>Type of this section.</td>
</tr>
<tr>
<td>JPSYXMOD</td>
<td>Modifier for this section</td>
</tr>
</tbody>
</table>

JES System Information Section: This section contains information about JES systems (MAS members for JES2) which were processed to obtain data for an SSI 82 call.

Note: This section reports only those systems (JES2 MAS members) which passed the system or member selection filters.
The fields in the JPSYSINF section are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSYLNG</td>
<td>Length of the section, including all system information entries.</td>
</tr>
<tr>
<td>JPSYTYPE</td>
<td>Type of this section.</td>
</tr>
<tr>
<td>JPSYMOD</td>
<td>Modifier for this section.</td>
</tr>
<tr>
<td>JPSYOENT</td>
<td>Offset to the first system information entry.</td>
</tr>
<tr>
<td>JPSYNENT</td>
<td>Number of system information entries returned.</td>
</tr>
<tr>
<td>JPSYSENT</td>
<td>Size of a system information entry.</td>
</tr>
</tbody>
</table>

**System Information Entry**: This entry contains information about a JES system (MAS member for JES2) which was processed to obtain data for an SSI 82 call.

The fields in the JPSYSIFE entry are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPSYSYSN</td>
<td>MVS system name.</td>
</tr>
<tr>
<td>JPSYMBRN</td>
<td>JES2 MAS member name.</td>
</tr>
<tr>
<td>JPSYSUBS</td>
<td>JES subsystem name.</td>
</tr>
<tr>
<td>JPSYCMCL</td>
<td>JES command prefix length.</td>
</tr>
<tr>
<td>JPSYCMCH</td>
<td>JES command prefix.</td>
</tr>
<tr>
<td>JPSYVERN</td>
<td>Version of JES.</td>
</tr>
<tr>
<td>JPSYFLAG</td>
<td>Processing flags.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>JPSYFPRC</td>
<td>Data processed for this system.</td>
</tr>
<tr>
<td></td>
<td>JPSYFNDT</td>
<td>No data returned for this system because no data was available or no data matched the filters.</td>
</tr>
<tr>
<td></td>
<td>JPSYFSUP</td>
<td>No data returned for this system - not supported.</td>
</tr>
<tr>
<td></td>
<td>JPSYFINA</td>
<td>No data returned for this system because system is not active.</td>
</tr>
<tr>
<td></td>
<td>JPSYFGLB</td>
<td>Global system in a complex (JES3).</td>
</tr>
<tr>
<td></td>
<td>JPSYFPRI</td>
<td>Primary subsystem.</td>
</tr>
<tr>
<td></td>
<td>JPSYVERD</td>
<td>Version of the data returned from this system.</td>
</tr>
<tr>
<td></td>
<td>JPSYMBNR</td>
<td>JES2 MAS member number.</td>
</tr>
</tbody>
</table>
SSI Function Code 82
Chapter 4. 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 Chapter 6, “SSI Function Codes Your Subsystem Can Support,” on page 317 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reg 0</td>
<td>Address of the SSCVT (mapped by the IEFJSCVT macro)</td>
</tr>
<tr>
<td>Reg 1</td>
<td>Address of the SSOB control block passed by the requestor. This is explained in &quot;Subsystem Options Block (SSOB)&quot; on page 7.</td>
</tr>
<tr>
<td>Reg 13</td>
<td>Standard 18-word save area</td>
</tr>
<tr>
<td>Reg 14</td>
<td>Return address</td>
</tr>
<tr>
<td>Reg 15</td>
<td>Entry point address</td>
</tr>
</tbody>
</table>

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 z/OS 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 Chapter 2, “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 Chapter 8, “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 306 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 304 or “Enabling Your Subsystem for New Functions” on page 306 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 306 for information on IEFSSVT and see "Deactivating Your Subsystem" on page 308 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 z/OS 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 z/OS MVS Initialization and Tuning Reference for detailed information on the syntax and rules for coding IEFSSNxx. See z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG for information on the syntax and rules for coding the IEFSSI macro. See z/OS 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 your 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 z/OS 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 302 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 IEFJ003I (see "Handling Initialization Errors" on page 383) 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 304 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 302.

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 294 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 z/OS MVS System Commands and z/OS MVS JCL Reference for more information on the START command.

Figure 28 on page 297 shows how you can initialize your subsystem either by specifying an initialization routine or by using the START command.
**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 [z/OS 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 [z/OS MVS Initialization and Tuning Reference](https://www.ibm.com/support/knowledgecenter/SSEPEG_10.1.0/using/subsys/parmlib/parmlib_subبار.html) 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 [z/OS MVS System Management Facilities (SMF)](https://www.ibm.com/support/knowledgecenter/POWER7/using/subsys/parmlib/parmlib_subبار.html) 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 364 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 z/OS 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.
Setting Up
Chapter 5. 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 [z/OS 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:

<table>
<thead>
<tr>
<th>Word</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>Address of the SSCVT (mapped by the IEFJSCVT macro).</td>
</tr>
<tr>
<td>Two</td>
<td>Address of the subsystem initialization parameter list (JSIPL, mapped by IEFJSIPL). See <a href="#">z/OS MVS Data Areas, Vol 3</a> (IVT-RCWK) for the format of JSIPL.</td>
</tr>
</tbody>
</table>

Figure 29 on page 303 shows the input to the initialization routine when your initialization routine gets control from the system.
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 310 and "Retrieving Subsystem-specific Information" on page 310 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 310 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 z/OS 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 z/OS 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 389 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 Chapter 9, “Using IEFJSVEC with Your Subsystem,” on page 403 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 z/OS 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 296.
- 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 z/OS 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 293 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 309 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 Chapter 4, “Setting Up Your Subsystem,” on page 291 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 Chapter 9, “Using IEFJSVEC with Your Subsystem,” on page 403 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 304 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 z/OS MVS System Commands for more information on the SETSSI ACTIVATE command and "Defining Subsystem Options" on page 310 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 304.
- 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
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 z/OS MVS Programming: Authorized Assembler Services Reference EDT-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 WORKASPB 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 30 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 30. 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 45 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 XMSET request places one fullword of subsystem passed data in the subsystem affinity entry, which is identified by the current TCB and the subsystem affinity index.

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 z/OS DFSMSdfp 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:

```
 SSAFF [SET [,TCB=tcb-address]]
 {XMSET} {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.

XMSET requests have the following requirements:

- The caller must be enabled, unlocked, and in supervisor state, key 0.
OBTAINT 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.

**XMSET**
- indicates that MVS is to place the value specified by the DATA parameter into the subsystem's associated subsystem affinity entry. The XMSET request destroys the contents of registers 14, 15, 0, and 1.

**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:
Services

- 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 XMSET 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 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.
Chapter 6. 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.

<table>
<thead>
<tr>
<th>Function Code</th>
<th>Requested Function</th>
<th>Type of Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>End-of-task</td>
<td>Broadcast</td>
</tr>
<tr>
<td>8</td>
<td>End-of-address space (End-of-memory)</td>
<td>Broadcast</td>
</tr>
<tr>
<td>9</td>
<td>WTO/WTOR</td>
<td>Broadcast</td>
</tr>
<tr>
<td>10</td>
<td>Command processing</td>
<td>Broadcast</td>
</tr>
<tr>
<td>14</td>
<td>Delete operator message</td>
<td>Broadcast</td>
</tr>
<tr>
<td>48</td>
<td>Help Call</td>
<td>Broadcast</td>
</tr>
<tr>
<td>50</td>
<td>Early notification of end-of-task</td>
<td>Broadcast</td>
</tr>
<tr>
<td>54</td>
<td>Request subsystem version information</td>
<td>Directed</td>
</tr>
<tr>
<td>58</td>
<td>SMF SUBPARM option change</td>
<td>Directed</td>
</tr>
<tr>
<td>78</td>
<td>Tape device selection</td>
<td>Broadcast</td>
</tr>
</tbody>
</table>

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 Chapter 8, “Examples — Subsystem Interface Routines” for coding examples of function routines.
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 [z/OS MVS Programming](https://www.ibm.com/support/knowledgecenter/SSDMDU_1.11.0/com.ibm.oscompr/zos_mvs_programming.pdf).

**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 291](https://www.ibm.com/support/knowledgecenter/SSDMDU_1.11.0/com.ibm.oscompr/zos_mvs_programming.pdf), 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 403 and “Enabling Your Subsystem for New Functions” on page 407 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
- IEFSET

The function routine receives control in the following environment:

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum authorization</td>
<td>Supervisor state with PSW key 0</td>
</tr>
<tr>
<td>Dispatchable unit mode</td>
<td>Task</td>
</tr>
<tr>
<td>AMODE</td>
<td>24-bit or 31-bit</td>
</tr>
<tr>
<td>Cross memory mode</td>
<td>PASN=HASN=SASN</td>
</tr>
<tr>
<td>ASC mode</td>
<td>Primary</td>
</tr>
<tr>
<td>Interrupt status</td>
<td>Enabled for I/O and external interrupts</td>
</tr>
<tr>
<td>Locks</td>
<td>No locks held</td>
</tr>
<tr>
<td>Control parameters</td>
<td>The SSIB, SSOB, and SSET control blocks reside in storage below 16 megabytes.</td>
</tr>
<tr>
<td>Recovery</td>
<td>The function routine should provide an ESTAE-type recovery environment. See [z/OS MVS Programming: Authorized Assembler Services Guide] for more information on how to set up an ESTAE-type recovery environment.</td>
</tr>
</tbody>
</table>

Figure 31 on page 320 shows the 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the subsystem’s SSCVT</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 4 (SSOBEOT)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of the SSIB control block</td>
</tr>
<tr>
<td>SSOBRETN</td>
<td>Return code from previous subsystem function routine or zero.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem enabled to receive this function code</td>
</tr>
</tbody>
</table>

**SSET Contents:** MVS sets the following fields in the SSET control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSETLEN</td>
<td>Length of the SSET (SSETSIZE) control block</td>
</tr>
<tr>
<td>SSETASID</td>
<td>ASID of the address space in which the task was active</td>
</tr>
<tr>
<td>SSETFLAG</td>
<td>Flag indicators</td>
</tr>
<tr>
<td></td>
<td>• SSETYPE ON — indicates an abnormal ending task</td>
</tr>
<tr>
<td></td>
<td>• SSETYPE OFF — indicates a normal ending task</td>
</tr>
<tr>
<td>SSETCBA</td>
<td>Address of ending task’s TCB</td>
</tr>
<tr>
<td>SSETASCB</td>
<td>Address of ending task’s ASCB</td>
</tr>
</tbody>
</table>

**Output Register Information**

Upon exit from the function routine, the general purpose registers must contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 12</td>
<td>Restored to contents on entry</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Return Code (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The function routine recognized the request but did not process it.</td>
</tr>
<tr>
<td>4</td>
<td>The function routine recognized the request and processed it.</td>
</tr>
</tbody>
</table>
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 [z/OS MVS Programming: Authorized Assembler Services Guide](https://www.ibm.com/support/docview.wss?uid=swg21185003).

**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 291, 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 403 and "Enabling Your Subsystem for New Functions" on page 407 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 [z/OS MVS Programming: Authorized Assembler Services Guide](https://www.ibm.com) for more information on an ESTAE-type recovery environment.

[Figure 32 on page 324](#) shows the 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the subsystem’s SSCVT</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
</table>
SSI Function Code 8

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 8 (SSOBEOM)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of SSIB control block</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of function dependent area (SSET control block)</td>
</tr>
</tbody>
</table>

**SSIB Contents:** MVS sets the following fields in the SSIB control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier ‘SSIB’</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of subsystem that is enabled to receive this function code.</td>
</tr>
</tbody>
</table>

**SSEN Contents:** MVS sets the following fields in the SSEN control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSENLEN</td>
<td>Length of SSEN (SSENSIZE) control block</td>
</tr>
<tr>
<td>SSENASID</td>
<td>ASID of ending address space</td>
</tr>
<tr>
<td>SSENFLAG</td>
<td>Flag indicators</td>
</tr>
<tr>
<td></td>
<td>• SSENTYPE ON — indicates an abnormal ending address space</td>
</tr>
<tr>
<td></td>
<td>• SSENTYPE OFF — indicates a normal ending address space</td>
</tr>
<tr>
<td>SSENJBNM</td>
<td>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).</td>
</tr>
<tr>
<td>SSENASCB</td>
<td>Address of ending address space’s ASCB</td>
</tr>
</tbody>
</table>

**Output Register Information**

Upon exit from the function routine, the general purpose registers must contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 12</td>
<td>Restored to contents on entry</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The function routine recognized the request but did not process it.</td>
</tr>
</tbody>
</table>
SSI Function Code 8

The function routine recognized the request and processed it.
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

DOBRCST 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,SCSVSN 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 IEAVG700 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 [z/OS MVS Planning: Operations](#).
- **Installation-written exit routines** — see [z/OS MVS Installation Exits](#).
- **Automation** — see [z/OS 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 [z/OS 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 EMCS) are the recommended programming interface for receiving MVS messages. See [z/OS 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 (such as
routing codes) plays in determining the destinations of a message. See z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO for more information.

Environment
Review “Function Routines/Function Codes” on page 291, 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 403 and “Enabling Your Subsystem for New Functions” on page 407 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
- IHAOWEQ
- 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 [z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG](https://www.ibm.com) 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” for more information about specifying to the system the action that should be taken by your function routine.

Note: Although the system supports AMODE=24 SSI routines monitoring function code 9, IBM recommends converting these SSI routines to AMODE=31. In a future release, IBM may change the location of the SSOB, SSIB, SSWT, and WQE control blocks to reside in 31-bit storage.

Input Register Information

On entry to the function routine the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the SSCVT</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSIBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 9 (SSOBWTO)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of the SSIB control block</td>
</tr>
<tr>
<td>SSOBRETN</td>
<td>Return code from previous function routine (when SSI function code 9 is operating in broadcast mode).</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function dependent area (SSWT control block)</td>
</tr>
</tbody>
</table>
**SSIB Contents:** MVS sets the following fields in the SSIB control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem which is enabled to receive this function code.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSWTLEN</td>
<td>Length of the SSWT (SSWTSIZE) control block</td>
</tr>
<tr>
<td>SSWTWQE</td>
<td>Address of the WQE control block</td>
</tr>
<tr>
<td>SSWTNMOD</td>
<td>Value of SUBSMOD keyword on the WTO macro</td>
</tr>
<tr>
<td>SSWTPRSP</td>
<td>Indicates whether the SSWTPRTY field is valid</td>
</tr>
</tbody>
</table>

**Note:** Because messages are no longer assigned priorities, this field is ignored. It remains present for compatibility purposes only.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSWTPRTY</td>
<td>Value of the PRTY keyword on the WTO macro</td>
</tr>
</tbody>
</table>

**Note:** Because messages are no longer assigned priorities, this field is ignored. It remains present for compatibility purposes only.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSWTSNSP</td>
<td>Indicates whether the JOBNAME keyword was specified on the WTO</td>
</tr>
<tr>
<td>SSWTSISP</td>
<td>Indicates whether the JOBID keyword was specified on the WTO</td>
</tr>
</tbody>
</table>

Figure 33 shows the environment for a single-line WTO in the SSWT control block.

![Figure 33](image-url)

Figure 33. 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WQETXTLN</td>
<td>Length of the message text</td>
</tr>
<tr>
<td>WQETS</td>
<td>EBCDIC time stamp</td>
</tr>
<tr>
<td>WQEJOBNM</td>
<td>Jobname (inserted by the primary subsystem)</td>
</tr>
<tr>
<td>WQETXT</td>
<td>Message text</td>
</tr>
<tr>
<td>WQEXA</td>
<td>Indicators</td>
</tr>
<tr>
<td></td>
<td>• WQEWTOR — indicates the message is a WTOR</td>
</tr>
<tr>
<td></td>
<td>• WQEAUTH — indicates the message is issued by an authorized program.</td>
</tr>
<tr>
<td>WQEASID</td>
<td>ASID of the message issuer</td>
</tr>
<tr>
<td>WQETCB</td>
<td>TCB address of the message issuer</td>
</tr>
<tr>
<td>WQESEQ#</td>
<td>Message DOM id</td>
</tr>
<tr>
<td>WQEMCSF1</td>
<td>Indicators</td>
</tr>
<tr>
<td></td>
<td>• WQEMCSA — indicates the WQEROUT and WQEDESCD fields are valid</td>
</tr>
<tr>
<td></td>
<td>• WQEMCSB — indicates the WQECNID and WQECNNME fields are valid</td>
</tr>
<tr>
<td></td>
<td>• WQEMCSC — indicates the message is a command response</td>
</tr>
<tr>
<td></td>
<td>• WQEMCSD — indicates the WQEMSGTP field is valid</td>
</tr>
<tr>
<td></td>
<td>• WQEMCSE — indicates the message is reply to WTOR</td>
</tr>
<tr>
<td></td>
<td>• WQEMCSFF — indicates BRDCAST was specified on the WTO</td>
</tr>
<tr>
<td></td>
<td>• WQEMCSG — indicates HCONLY was specified on the WTO.</td>
</tr>
<tr>
<td>WQEMCSF2</td>
<td>Indicators</td>
</tr>
<tr>
<td></td>
<td>• WQEMCSM — indicates the message is a hardcopy image of the operator command</td>
</tr>
<tr>
<td></td>
<td>• WQEMCSN — indicates that NOCPY was specified on the WTO.</td>
</tr>
<tr>
<td>WQEMSGTP</td>
<td>Message type</td>
</tr>
<tr>
<td>WQEROUT</td>
<td>Routing codes</td>
</tr>
<tr>
<td>WQEFLG1</td>
<td>Indicators</td>
</tr>
<tr>
<td></td>
<td>• WQERETAN — indicates that the message is retained by AMRF</td>
</tr>
<tr>
<td></td>
<td>• WQENMOD — indicates the subsystem cannot modify the message</td>
</tr>
<tr>
<td></td>
<td>• WQEPPNA — non-action message issued by a non-authorized program</td>
</tr>
<tr>
<td></td>
<td>• WQERISS — indicates the message is an SVC reissue of a message that has</td>
</tr>
<tr>
<td></td>
<td>already been processed by SVC. WTO MPF and the SSI have already processed</td>
</tr>
<tr>
<td></td>
<td>the message. Note that MPF processing occurs only during the original</td>
</tr>
<tr>
<td></td>
<td>SVC WTO.</td>
</tr>
</tbody>
</table>

| WQEDESCD       | Descriptor codes                                                             |
| WQEJSTCB       | Address of the job step TCB                                                 |
| WQEVRSN        | Version level — contains the WQEVRID                                        |
| WQESYSNM       | System name                                                                  |
| WQEXMOD        | Copy of the MPF/IEAVMXIT user exit request flags. Indicator:                |
• **WQERDTM** — indicates that an MPF exit has requested the deletion of the message. You can override the requested deletion of the message by setting this bit off.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WQEMMLVL</td>
<td>Message level</td>
</tr>
<tr>
<td>WQEEERC</td>
<td>Extended routing codes</td>
</tr>
<tr>
<td>WQELENG</td>
<td>Length of WQE — contains the WQESIZE</td>
</tr>
<tr>
<td>WQEKEY</td>
<td>Value of the KEY keyword on the WTO</td>
</tr>
<tr>
<td>WQETOKN</td>
<td>Value of the TOKEN keyword on the WTO</td>
</tr>
<tr>
<td>WQECNID</td>
<td>Console ID</td>
</tr>
<tr>
<td>WQEOJBID</td>
<td>Originating job ID</td>
</tr>
<tr>
<td>WQEOJBNM</td>
<td>Originating job name</td>
</tr>
<tr>
<td>WQEAUTOT</td>
<td>Value of the automation token from MPF</td>
</tr>
<tr>
<td>WQEERFS</td>
<td>Extended request flags from the MPF/IEAVMXIT user exit</td>
</tr>
<tr>
<td>WQEDOMD</td>
<td>— indicates the message has been deleted by the DOM macro.</td>
</tr>
<tr>
<td>WQENBEW</td>
<td>— indicates the message created by a 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.</td>
</tr>
<tr>
<td>WQENHABD</td>
<td>— indicates the message has been displayed on the IPL or system console. This is a result of issuing a WTO with SYNCH=YES specified.</td>
</tr>
<tr>
<td>WQECASEL</td>
<td>Message color</td>
</tr>
<tr>
<td>WQEHASEL</td>
<td>Message highlighting</td>
</tr>
<tr>
<td>WQEIASEL</td>
<td>Message intensity</td>
</tr>
<tr>
<td>WQEMISC</td>
<td>Indicator</td>
</tr>
<tr>
<td>WQEBENIP</td>
<td>• <strong>WQEDOMD</strong> — indicates the message has been deleted by the DOM macro.</td>
</tr>
<tr>
<td></td>
<td>• <strong>WQENBEW</strong> — indicates the message created by a 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.</td>
</tr>
<tr>
<td></td>
<td>• <strong>WQENHABD</strong> — indicates the message has been displayed on the IPL or system console. This is a result of issuing a WTO with SYNCH=YES specified.</td>
</tr>
<tr>
<td></td>
<td>• <strong>WQEAUTO</strong> — indicates AUTO(Y) specified in the MPF for this message.</td>
</tr>
</tbody>
</table>

**SSWT Contents for the First Line of 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 331 for the fields that MVS sets as they are the same except for the SSWTMIN field which contains the following:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSWTMIN</td>
<td>Address of the minor WQE</td>
</tr>
</tbody>
</table>

Figure 34 on page 334 shows the environment for the first line of 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMJMMLW</td>
<td>Multi-line indicator</td>
</tr>
<tr>
<td>WMJMMLWB</td>
<td>— indicates the WQE is multi-line</td>
</tr>
<tr>
<td>WMJMAREA</td>
<td>Value specified on the WTO AREA keyword</td>
</tr>
<tr>
<td>WMJMTXTL</td>
<td>Length of message text</td>
</tr>
<tr>
<td>WMJMTS</td>
<td>EBCDIC time stamp</td>
</tr>
<tr>
<td>WMJMJBNM</td>
<td>Jobname (inserted by the primary subsystem)</td>
</tr>
<tr>
<td>WMJMTXT</td>
<td>Message text</td>
</tr>
<tr>
<td>WMJMDSP</td>
<td>Indicator</td>
</tr>
<tr>
<td>WMJMDSPH</td>
<td>— indicates the message issued by an authorized program</td>
</tr>
<tr>
<td>WMJMASID</td>
<td>ASID of the message issuer</td>
</tr>
<tr>
<td>WMJMTCB</td>
<td>TCB address of the message issuer</td>
</tr>
<tr>
<td>WMJMSEQ#</td>
<td>Message DOM id</td>
</tr>
<tr>
<td>WMJMMCS1</td>
<td>Indicators</td>
</tr>
<tr>
<td>WMJMMCS1A</td>
<td>— indicates that the WMJMRTC and WMJMDEC fields are valid</td>
</tr>
<tr>
<td>WMJMMCS1B</td>
<td>— indicates that the WMJMNCNID and WMJMNCNME fields are valid</td>
</tr>
<tr>
<td>WMJMMCS1C</td>
<td>— indicates the message is a command response</td>
</tr>
<tr>
<td>WMJMMCS1D</td>
<td>— indicates the WMJMMT field is valid</td>
</tr>
<tr>
<td>WMJMMCS1E</td>
<td>— indicates the message is a reply to the WTOR</td>
</tr>
<tr>
<td>WMJMMCS1F</td>
<td>— indicates the BRDCAST keyword is specified on the WTO</td>
</tr>
<tr>
<td>WMJMMCS1G</td>
<td>— indicates the HCONLY is specified on the WTO</td>
</tr>
<tr>
<td>WMJMMCS2</td>
<td>Indicators</td>
</tr>
<tr>
<td>WMJMMCS2E</td>
<td>— indicates the message is the hardcopy image of the operator command</td>
</tr>
<tr>
<td>WMJMMCS2F</td>
<td>— indicates the NOCPY is specified on the WTO</td>
</tr>
<tr>
<td>WMJMLTY1</td>
<td>Line type indicator</td>
</tr>
</tbody>
</table>

**Figure 34. Environment for the First Line of a Multi-line WTO in the SSWT Control Block**
Note: The WMJMLTYC and WMJMLTYD fields can be on at same time.

WM JMRTC Routing codes

WMJMF LG1 Indicator

- WMJMR ETN — indicates the message will be retained by AMRF
- WMJMN MOD — indicates the subsystem cannot modify the message
- WMJMPPNA — indicates the message is issued by the problem program
- WMJM RISS — 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).

WM JMDEC Descriptor codes

WMJMJTCB Address of job step TCB

WMJMVRSN Version level — contains the WQEVRID

WMJMSNM System name

WMJMXMOD Copy of the MPF/IEAVMXIT user exit request flags. Indicator:

- WMJMRDTM — indicates that an MPF exit has requested the deletion of the message. You can override the requested deletion of the message by setting this bit off.

WMJMMLVL Message level

WMJMER C Extended routing codes

WMJMLENG Length of WQE — contains the WMJMSIZE

WMJMKEY Value of the KEY keyword on the WTO

WMJM TOKN Value TOKEN keyword on the WTO

WMJMCNID Console ID

WMJMOJBI Originating job ID

WMJMOJBN Originating job name

WMJAUTOT Value of the automation token 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

WMJ BENIP 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.

- **WMJMHABD** — 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMNMLT1</td>
<td>Line type indicators</td>
</tr>
<tr>
<td></td>
<td>• <strong>WMNMLT1B</strong> — label line</td>
</tr>
<tr>
<td></td>
<td>• <strong>WMNMLT1C</strong> — data line</td>
</tr>
<tr>
<td></td>
<td>• <strong>WMNMLT1D</strong> — end line</td>
</tr>
</tbody>
</table>

- **Note:** The WMNMLT1C and WMNMLT1D fields can be on at same time.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMNMLTH1</td>
<td>Length of the minor WQE</td>
</tr>
<tr>
<td>WMNMTL1</td>
<td>Length of the minor text</td>
</tr>
<tr>
<td>WMNMTXT1</td>
<td>Minor line text</td>
</tr>
<tr>
<td>WMN1XMOD</td>
<td>Copy of the request flags</td>
</tr>
</tbody>
</table>

Figure 35 shows the environment for minor lines of a multi-line WTO in the SSWT control block.

![Figure 35. Environment for Minor Lines of a Multi-line WTO in the SSWT Control Block](image-url)
Multi-line Use Information

The following example illustrates the method in which the system presents a multi-line message to the SSI for updating. Consider the six-line message, IEE889I:

```
1 IEE889I 09.38.52 CONSOLE DISPLAY 191
2 MSG: CURR=0 LIM=1500 RPLY:CURR=0 LIM=10 SYS=SCOTT01 PFK=01
3 CONSOLE ID ---------------- SPECIFICATIONS ---------------
4 SYSLOG COND=H AUTH=CMDS NBUF=N/A
5 ROUTCDE=ALL
6 LOG BUFFERS IN USE: 0 LOG BUFFER LIMIT: 1000
```

A multi-line message consists of a major WQE (the first line) and minor WQEs (subsequent lines). Minor WQEs are paired when possible. In IEE889I, the two minor WQE pairs are 2 & 3 and 4 & 5. The remaining minor WQE, 6, is not paired. The pairing of minor WQEs affects how the system passes message lines to the SSI, as shown below in Table 9.

The system passes IEE889I to the SSI six times. On the first call, the SSI can modify only 1, the major WQE. On the second call, only 2 (the first line of a minor WQE pair) may be modified; on the third call, only 3 (the second line of a minor WQE pair) may be modified, and so on. The system passes IEE889I to the SSI six times, giving the SSI one opportunity to modify each of the six lines.

Table 9. SSI updating of multi-line messages

<table>
<thead>
<tr>
<th>On this call...</th>
<th>...the system passes these lines to the SSI...</th>
<th>...and the SSI can modify this line:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

The SSI assigns fields to point at the lines of a multi-line message. SSWTWQE always points at the major WQE. If minor WQEs are present, then SSWTMIN points at the first minor WQE of the pair. If there is a second line of a minor WQE pair, WMNMNX1 (which is within the first minor WQE) points at it.

**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 331 for the fields that MVS sets as they are the same except for the SSWTORE field which contains the following:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSWTORE</td>
<td>Address of the ORE control block</td>
</tr>
</tbody>
</table>
SSI Function Code 9

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 339 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):

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORERPYA</td>
<td>Address of the WTOR issuer's reply buffer</td>
</tr>
<tr>
<td>OREECBA</td>
<td>Address of the WTOR issuers ECB</td>
</tr>
<tr>
<td>ORECBID</td>
<td>Acronym — 'ORE'</td>
</tr>
<tr>
<td>OREVRSN</td>
<td>Version level — OREVRID</td>
</tr>
<tr>
<td>ORELNTH</td>
<td>Maximum length of the requested reply (specified by the WTOR issuer)</td>
</tr>
<tr>
<td>ORERPIDB</td>
<td>Binary reply ID</td>
</tr>
</tbody>
</table>

Figure 36 shows the environment for a WTOR in the SSWT control block.

Figure 36. Environment for a WTOR in the SSWT Control Block

Output Register Information
Upon exit from the function routine, the general purpose registers must contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 12</td>
<td>Restored to contents on entry</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>(Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
</table>
The function routine recognized the request but did not process it.

Do not display message; hardcopy message
Display message; do not hardcopy message
Do not display message; do not hardcopy message

Note: For a multi-line WTO, the SSOBRETN field is only accepted for the first call to the function routine. The SSOBRETN field is ignored on subsequent calls to present the minor lines.

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WQETXTLN</td>
<td>New length of the message text (if text was altered)</td>
</tr>
<tr>
<td>WQETXT</td>
<td>New/changed message text</td>
</tr>
<tr>
<td>WQEMSGTP</td>
<td>New/changed message type. The WQEMCSD field must be set appropriately.</td>
</tr>
<tr>
<td>WQEROUT</td>
<td>New/changed routing codes. The WQEMCSA field must be set appropriately.</td>
</tr>
<tr>
<td>WQEDESCD</td>
<td>New/changed descriptor codes. The WQEMCSA field must be set appropriately. The WQEROUT field must be non-zero.</td>
</tr>
<tr>
<td>WQEERC</td>
<td>New/changed extended routing codes</td>
</tr>
<tr>
<td>WQECASEL</td>
<td>New/changed message color</td>
</tr>
<tr>
<td>WQEHASEL</td>
<td>New/changed message highlighting</td>
</tr>
<tr>
<td>WQEIASEL</td>
<td>New/changed message intensity</td>
</tr>
<tr>
<td>WQEFGLG1</td>
<td>Indicator</td>
</tr>
<tr>
<td></td>
<td>- <strong>WQERETAN</strong> — indicates that the message is retained by AMRF</td>
</tr>
<tr>
<td>WQEXMOD</td>
<td>Copy of the MPF/IEAVMXIT user exit request flags. Indicator:</td>
</tr>
<tr>
<td></td>
<td>- <strong>WQERDTM</strong> — indicates that an MPF exit has requested the deletion of the message. You can override the requested deletion of the message by setting this bit off.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMJMTXTLN</td>
<td>New length of the message text (if text was altered)</td>
</tr>
<tr>
<td>WMJMTXT</td>
<td>New/changed message text</td>
</tr>
<tr>
<td>WMJMXXMOD</td>
<td>Copy of the MPF/IEAVMXIT user exit request flags</td>
</tr>
<tr>
<td>WMJMMMT</td>
<td>New/changed message type. The WMJMCS1D field must be set.</td>
</tr>
<tr>
<td>WMJMRTC</td>
<td>New/changed routing codes. The WMJMCS1A field must be set.</td>
</tr>
</tbody>
</table>
SSI Function Code 9

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMJMDEC</td>
<td>New/changed descriptor codes. The WMJMCS1A field must be set. The WMJMRTC field must be non-zero.</td>
</tr>
<tr>
<td>WMJMERC</td>
<td>New/changed extended routing codes</td>
</tr>
<tr>
<td>WMJCASEL</td>
<td>New/changed message color</td>
</tr>
<tr>
<td>WMJHASEL</td>
<td>New/changed message highlighting</td>
</tr>
<tr>
<td>WMJIASEL</td>
<td>New/changed message intensity</td>
</tr>
<tr>
<td>WMJMRDTM</td>
<td>Indicates whether the message is to be deleted</td>
</tr>
<tr>
<td>WMJMRETN</td>
<td>Indicates that the message is retained by AMRF</td>
</tr>
</tbody>
</table>

ORE Contents for a WTO: Any changes to the ORE are ignored.

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMNMTL1</td>
<td>New length of the minor text (if the WMNMTXT1 field is modified)</td>
</tr>
<tr>
<td>WMNMTXT1</td>
<td>New/changed minor line text</td>
</tr>
</tbody>
</table>
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 z/OS MVS Planning: Operations for command authority concepts. See z/OS 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 291, 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 403 and “Enabling Your Subsystem for New Functions” on page 407 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
SSI Function Code 10

The function routine receives control in the following environment:

<table>
<thead>
<tr>
<th>Minimum authorization</th>
<th>Supervisor state with PSW key 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatchable unit mode</td>
<td>Task</td>
</tr>
<tr>
<td>AMODE</td>
<td>24-bit or 31-bit</td>
</tr>
<tr>
<td>Cross memory mode</td>
<td>PASN=HASN=SASN</td>
</tr>
<tr>
<td>ASC mode</td>
<td>Primary</td>
</tr>
<tr>
<td>Interrupt status</td>
<td>Enabled for I/O and external interrupts</td>
</tr>
<tr>
<td>Locks</td>
<td>No locks held</td>
</tr>
<tr>
<td>Control parameters</td>
<td>The SSOB, SSIB, and SSCM control blocks reside in storage below 16 megabytes.</td>
</tr>
<tr>
<td>Recovery</td>
<td>The function routine should provide an ESTAE-type recovery environment. See z/OS MVS Programming: Authorized Assembler Services Reference EDT-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.</td>
</tr>
</tbody>
</table>

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 345 for instructions on what your function routine should specify to the system.

Figure 37 on page 343 shows the 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the subsystem’s SSCVT</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address</td>
</tr>
</tbody>
</table>

Input Parameters
Input parameters for the function routine are:
- SSOB
### SSOB Contents
MVS sets the following fields in the SSOB control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 10 (SSOBCMND)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of the SSIB control block</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function dependent area (SSCM control block)</td>
</tr>
</tbody>
</table>

### SSIB Contents
MVS sets the following fields in the SSIB control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name 'MSTR'</td>
</tr>
</tbody>
</table>

### SSCM Contents
MVS sets the following fields in the SSCM control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSCMLLEN</td>
<td>Length of the SSCM (SSCMSIZE) control block</td>
</tr>
<tr>
<td>SSCMVRSN</td>
<td>Version level of the SSCM (SSCMVRID) control block</td>
</tr>
<tr>
<td>SSCMBUFF</td>
<td>Address of the command buffer in the MGCRPL control block</td>
</tr>
<tr>
<td>SCCMACRN</td>
<td>Identifier 'SSCM'</td>
</tr>
<tr>
<td>SSCMAUTH</td>
<td>Command authority of command issuer — see the SSCM control block information for the definition of flags within this byte.</td>
</tr>
<tr>
<td>SSCMDISP</td>
<td>Disposition flags — see the SSCM control block information for the definition of flags within this byte.</td>
</tr>
<tr>
<td>SSCMBOLEN</td>
<td>Length of the command buffer pointed to by the SSCMBUFF field.</td>
</tr>
<tr>
<td>SSCMOLIB</td>
<td>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).</td>
</tr>
<tr>
<td>SSCMOLIP</td>
<td>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.</td>
</tr>
<tr>
<td>SSCMSYMS</td>
<td>The command text was changed by symbolic substitution.</td>
</tr>
<tr>
<td>SSCMUTOK</td>
<td>Address of the UTOKEN</td>
</tr>
<tr>
<td>SSCMUTOK</td>
<td>The UTOKEN identifies the issuer of the command. The RACROUTE macro accepts the UTOKEN to perform command authorization checking using a security product (RACF).</td>
</tr>
<tr>
<td>SSCMULTH</td>
<td>Length of the UTOKEN</td>
</tr>
</tbody>
</table>
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

SSCMCTX  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 SSCSCNM 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.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSCMCVRB</td>
<td>Command identifier — REPLY</td>
</tr>
<tr>
<td>SSCMRTCB</td>
<td>TCB address of the WTOR issuer</td>
</tr>
<tr>
<td>SSCMRASI</td>
<td>ASID of the WTOR issuer</td>
</tr>
<tr>
<td>SSCMRTXT</td>
<td>Offset to the reply text in the area pointed to by either the SSCMCTX field or SSCMBUFF+4.</td>
</tr>
<tr>
<td>SSCMRFLG</td>
<td>Reply flag</td>
</tr>
<tr>
<td>SSCMRSEC</td>
<td>Indicates whether the REPLY is to a security WTOR (route code of 9).</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MGCRNLGTH</td>
<td>Length of the command text + 4</td>
</tr>
<tr>
<td>MGCRFLG2</td>
<td>Command processing flags — see the MGCRPL control block information for a definition of these flags.</td>
</tr>
<tr>
<td>MGCRTEXT</td>
<td>Command text</td>
</tr>
</tbody>
</table>

**Output Register Information**

Upon exit from the function routine, the general purpose registers must contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 12</td>
<td>Restored to contents on entry</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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:
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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MGCRLGTH</td>
<td>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.</td>
</tr>
<tr>
<td>MGCRTEXT</td>
<td>Command text — the command text can be altered and replaced. If the length is changed, MGCRLGTH (above) must also be updated.</td>
</tr>
</tbody>
</table>

**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 [z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN](https://www.ibm.com) 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 [z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU](https://www.ibm.com)) 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. Instances of 1-byte console ID usage will be recorded by the Console ID Tracking facility, which is invoked with the CNZTRKR macro. For information regarding the removal of 1-byte console IDs in favor of 4-byte console IDs, see [z/OS MVS Planning: Operations](http://www.ibm.com/.

- 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=(CN1GRPSYSCLONE.)
```

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 [z/OS MVS Data Areas, Vol 1 (ABEP-DALT)](http://www.ibm.com/) 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.
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

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 z/OS MVS Programming, Authorized Assembler Services Reference ALE-DYN and Authorized Assembler Services Guide.

Environment

Review “Function Routines/Function Codes” on page 291, 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 403 and “Enabling Your Subsystem for New Functions” on page 407 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:

<table>
<thead>
<tr>
<th>Minimum authorization</th>
<th>Supervisor state with PSW key 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatchable unit mode</td>
<td>Task</td>
</tr>
<tr>
<td>AMODE</td>
<td>24-bit or 31-bit</td>
</tr>
<tr>
<td>Cross memory mode</td>
<td>PASN=HASN=SASN</td>
</tr>
<tr>
<td>ASC mode</td>
<td>Primary</td>
</tr>
<tr>
<td>Interrupt status</td>
<td>Enabled for I/O and external interrupts</td>
</tr>
<tr>
<td>Locks</td>
<td>No locks held</td>
</tr>
<tr>
<td>Control parameters</td>
<td>The SSOB, SSIB, and SSDM control blocks reside in storage below 16 megabytes.</td>
</tr>
<tr>
<td>Recovery</td>
<td>The function routine should provide an ESTAE-type recovery environment. See <a href="https://www.ibm.com">z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG</a> for more information on these macros. Failure to establish a recovery environment ends the processing of the current DOM if an abend occurs.</td>
</tr>
</tbody>
</table>

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.

**Note:** Although the system supports AMODE=24 SSI routines monitoring function code 14, IBM recommends converting these SSI routines to AMODE=31. In a future release, IBM may change the location of the SSOB, SSIB, and SSDM control blocks to reside in 31-bit storage.

### Input Register Information

On entry to the function routine the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the subsystem’s SSCVT</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLLEN</td>
<td>Length of the SSOB (SSIBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 14 (SSOBDOM)</td>
</tr>
<tr>
<td>SSOBSIB</td>
<td>Address of the SSIB control block</td>
</tr>
</tbody>
</table>
### SSI Function Code 14

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSDMLEN</td>
<td>Length of the SSDM (SSDMSIZE) control block</td>
</tr>
<tr>
<td>SSDMVRSN</td>
<td>Version level of the SSDM (SSDMVRID) control block</td>
</tr>
<tr>
<td>SSDMACRN</td>
<td>Identifier ‘SSDM’</td>
</tr>
<tr>
<td>SSDMSEND</td>
<td>Indicator that the DOM request should be communicated to other systems</td>
</tr>
<tr>
<td>SSDMDMCB</td>
<td>The address of that part of the DOMC that is passed to the subsystem</td>
</tr>
<tr>
<td>SSDMDMC2</td>
<td>The address of the entire DOMC that is passed to the subsystem</td>
</tr>
</tbody>
</table>

**DOMC Contents:** The address of the DOMC control block is in SSDMDMC2. See [z/OS 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 12</td>
<td>Restored to contents on entry</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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:** Any changes made to the DOMC are ignored.

**Restrictions:** None.

**General Considerations:** IBM recommends that the function routine does not alter the DOMC (all changes to the DOMC are ignored).
Help Call — SSI Function Code 48

The Help call (SSI function code 48) provides the ability for a subsystem to get control during processing of a SYSABEND and a SYSUDUMP.

Type of Request
Broadcast SSI call.

Use Information
Your subsystem can use the SSI function code 48 to assist in an ABEND Dump.

Issued to
All active subsystems that indicate they support the Help function when the system (MVS) issues the Help call.

Related SSI Codes
None.

Related Concepts
None.

Environment
Review [“Function Routines/Function Codes” on page 291.], which describes both the general environment on entry to your function routine and other programming considerations that your function routine should take into account.

Data areas commonly referenced are mapped by the following mapping macros. IBM recommends you include them in your function routine:
- IEFSSOBH
- IEFJSSIB
- BLSABDPL

The function routine receives control in the following environment:

Minimum authorization: Supervisor state with PSW key 0
Dispachable 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 ABDPL control blocks reside in storage below 16 megabytes.

Recovery
The function routine should provide an ESTAE-type recovery environment. See the topic on [Writing Recovery Routines in z/OS MVS Programming: Authorized Assembler Services Guide] for more information on how to set up an ESTAE-type recovery environment.

Figure 38 on page 352 shows the environment on entry to the function routine for SSI function code 48.
### Input Register Information
On entry to the function routine the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the subsystem’s SSCVT</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address</td>
</tr>
</tbody>
</table>

### Input Parameters
Input parameters for the function routine are:
- SSOB
- SSIB
- ABDPL

**SSOB Contents:** MVS sets the following fields in the SSOB control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 48</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of the SSIB control block</td>
</tr>
<tr>
<td>SSOBRETN</td>
<td>Return code from previous subsystem function routine or zero.</td>
</tr>
</tbody>
</table>

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" on page 321 for a list of possible SSOBRETN return codes.
SSOBINDV Address of the function dependent area (ABDPL block)

**SSIB Contents:** MVS sets the following fields in the SSIB control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem enabled to receive this function code.</td>
</tr>
</tbody>
</table>

**Output Register Information**

Upon exit from the function routine, the general purpose registers must contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 12</td>
<td>Restored to contents on entry</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Return Code (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The function routine recognized the request but did not process it.</td>
</tr>
<tr>
<td>4</td>
<td>The function routine recognized the request and processed it.</td>
</tr>
</tbody>
</table>
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 z/OS 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 291, 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 403 and [Enabling Your Subsystem for New Functions] on page 407 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
- IEFSSSET

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 [z/OS MVS Programming: Authorized Assembler Services Guide](https://www.ibm.com/docs/en/zos) for more information on an ESTAE-type recovery environment.

[Figure 39 on page 356](#) shows the 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the subsystem’s SSCVT</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 50 (SSOBFEOT)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of the SSIB control block</td>
</tr>
<tr>
<td>SSOBRETN</td>
<td>Return code from previous subsystem function routine or zero.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE) control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of subsystem which is enabled to receive this function code.</td>
</tr>
</tbody>
</table>

**SSET Contents:** The following fields in the SSET control block are set on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSETLEN</td>
<td>Length of the SSET (SSETSIZE) control block</td>
</tr>
<tr>
<td>SSETASID</td>
<td>ASID of address space in which task was active</td>
</tr>
</tbody>
</table>
| 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 12</td>
<td>Restored to contents on entry</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The function routine recognized the request but did not process it.</td>
</tr>
<tr>
<td>4</td>
<td>The function routine recognized the request and processed it.</td>
</tr>
</tbody>
</table>
SSI Function Code 54

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 your 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 *z/OS MVS Programming: Authorized Assembler Services Guide* for more information on how to set up an ESTAE-type recovery environment.

*Figure 40* shows the environment on entry to the function routine for SSI function code 54.

![Figure 40. Environment on Entry to the Function Routine for SSI Function Code 54](image)

**Input Register Information**

On entry to the function routine the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the SSCVT</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
</tbody>
</table>
SSI Function Code 54

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier ‘SSOB’</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of SSOB control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 54 (SSOBSSVI)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of SSIB control block</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of function dependent area (SSVI control block)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier ‘SSIB’</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of SSIB control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the user provided subsystem invoked</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 12</td>
<td>Restored to contents on entry</td>
</tr>
<tr>
<td>14</td>
<td>Return Address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBRETN</td>
<td>The function routine sets this field to SSVIOK (decimal 0).</td>
</tr>
<tr>
<td>SSVIRLEN</td>
<td>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.</td>
</tr>
<tr>
<td>SSVIRVER</td>
<td>The function routine sets this field to the version of the SSVI macro used (SSVICVER).</td>
</tr>
<tr>
<td>SSVIFLEN</td>
<td>The function routine sets this field to the length of the fixed header output section (SSVIFSIZ).</td>
</tr>
<tr>
<td>SSVIASID</td>
<td>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.</td>
</tr>
<tr>
<td>SSVIFMID</td>
<td>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.</td>
</tr>
<tr>
<td>SSVIVERS</td>
<td>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.</td>
</tr>
<tr>
<td>SSVICNAM</td>
<td>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.</td>
</tr>
<tr>
<td>SSVIPLVL</td>
<td>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.</td>
</tr>
<tr>
<td>SSVISLVL</td>
<td>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.</td>
</tr>
<tr>
<td>SSVIUDOF</td>
<td>The function routine sets this field to zero (there is no installation variable output section).</td>
</tr>
<tr>
<td>SSVISDOF</td>
<td>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...</td>
</tr>
</tbody>
</table>
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 358 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBRETN</td>
<td>The function routine sets SSOBRETN to the value of SSVINSTR (decimal 8).</td>
</tr>
<tr>
<td>SSVIRLEN</td>
<td>The function routine sets SSVIRLEN to the amount of storage needed to satisfy the request.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBRETN</td>
<td>The function routine sets SSOBRETN to the value of SSVIPARM (decimal 16).</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBRETN</td>
<td>The function routine sets SSOBRETN to the value of SSVIABL (decimal 24).</td>
</tr>
</tbody>
</table>

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 298 for a description of this relationship and an example of the associated processing.

Environment
Review “Function Routines/Function Codes” on page 291, 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 403 and “Enabling Your Subsystem for New Functions” on page 407 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 41 shows the environment at the time of the call for SSI function code 58.

### Input Register Information

On entry to the function routine the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the SSCVT</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB control block</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Entry point address</td>
</tr>
</tbody>
</table>

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:
SSI Function Code 58

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 58 (SSOBSMAC)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of SSIB control block</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function dependent area (SSSM control block)</td>
</tr>
</tbody>
</table>

**SSIB Contents:** SMF sets the following fields in the SSIB control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB control block</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Subsystem name — name of the subsystem that this SMF SUBPARM Option Change call is directed to.</td>
</tr>
</tbody>
</table>

**SSSM Contents:** SMF sets the following fields in the SSSM control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSMLEN</td>
<td>Length of the SSSM control block</td>
</tr>
<tr>
<td>SSSMFLGS</td>
<td>Flags</td>
</tr>
</tbody>
</table>
  * 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSMCNID</td>
<td>Command console ID</td>
</tr>
<tr>
<td>SSSTOKN</td>
<td>Command CART</td>
</tr>
</tbody>
</table>

**Output Register Information**

Upon exit from the function routine, the general purpose registers must contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 12</td>
<td>Restored to contents on entry</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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 SSORETN field in the SSOB control block to one of the following:

<table>
<thead>
<tr>
<th>Return Code (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The function routine recognized the request but did not process it.</td>
</tr>
<tr>
<td>4</td>
<td>The function routine recognized the request and processed it.</td>
</tr>
</tbody>
</table>

**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 298 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 298 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 SSTAIBM 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 SSTAUSR field. If SSTAUSR bits are
set, the system generates eligibility values that combine SSTAUSR settings
and SSTAIBM 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.

Table 10 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 SSTAIBM; the second column shows the 1-bit fields the function
routine can set in SSTAUSR. 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
• SSTDADMND, 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.

Table 10. Relationship between System and User Criteria

<table>
<thead>
<tr>
<th>Importance</th>
<th>System criteria (SSTAIBM)</th>
<th>User criteria (SSTAUSR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SSTAIBM</td>
<td>SSTAINEL</td>
</tr>
<tr>
<td>2</td>
<td>SSTDADMND</td>
<td>SSTAUSR01</td>
</tr>
<tr>
<td>3</td>
<td>SSTAUSR01</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SSTAUSR02</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SSTAONUN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SSTAUSR03</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SSTAUSR04</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>SSTAUSR05</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>SSTAUSR06</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SSTAUSR07</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>SSTAUSR08</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Generic device type not specified by a bit</td>
<td></td>
</tr>
</tbody>
</table>
Table 10. Relationship between System and User Criteria (continued)

<table>
<thead>
<tr>
<th>Importance</th>
<th>System criteria (SSTAIBMM)</th>
<th>User criteria (SSTAUSRM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td>SSTAUS09</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>SSTAUS10</td>
</tr>
<tr>
<td>17</td>
<td>SSTAAACL1</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>SSTAUS11</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>SSTAUS12</td>
</tr>
<tr>
<td>20</td>
<td>SSTAAACL2</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>SSTAUS13</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>SSTAUS14</td>
</tr>
<tr>
<td>23</td>
<td>SSTAAACL3</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>SSTAUS15</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>SSTAUS16</td>
</tr>
<tr>
<td>26</td>
<td>SSTAVOLM</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>SSTAUS17</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>SSTAUS18</td>
</tr>
<tr>
<td>29</td>
<td>SSTANVOL</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>SSTAUS19</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>SSTAUS20</td>
</tr>
<tr>
<td>32</td>
<td>SSTAWVOL</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>SSTAUS21</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>SSTAUS22</td>
</tr>
<tr>
<td>35</td>
<td>SSTAAVOL</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td>SSTAUS23</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td>SSTAUS24</td>
</tr>
<tr>
<td>38</td>
<td>SSTAANAS</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
<td>SSTAUS25</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>SSTAUS26</td>
</tr>
</tbody>
</table>

Descriptions of SSTAIBMM fields are found in "Input Parameters" on page 374; descriptions of SSTAUSRM fields are found in "Output Parameters" on page 377.

Environment

Review "Function Routines/Function Codes" on page 291, 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 403 and "Enabling Your Subsystem for New Functions" on page 407 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
- IEFSSSTA

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 [z/OS MVS Programming: Authorized Assembler Services Guide](https://publib.boulder.ibm.com/infocenter/pseries/v3r1/)

The following figures show the environment at the time of the call for SSI function code 78.
SSI Function Code 78

Figure 42. Environment at Time of Call for SSI Function Code 78
**Input Register Information**

On entry to the function routine the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address of the subsystem's SSCVT</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
</tbody>
</table>

---

*Figure 43. Continuation of Environment at Time of Call for SSI Function Code 78*
SSI Function Code 78

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSOBID</td>
<td>Identifier 'SSOB'</td>
</tr>
<tr>
<td>SSOBLEN</td>
<td>Length of the SSOB (SSOBHSIZ) control block</td>
</tr>
<tr>
<td>SSOBFUNC</td>
<td>SSI function code 78 (SSOBTALC)</td>
</tr>
<tr>
<td>SSOBSSIB</td>
<td>Address of the SSIB control block</td>
</tr>
<tr>
<td>SSOBRETN</td>
<td>Return code value set by previously invoked function routine, or zero</td>
</tr>
<tr>
<td>SSOBINDV</td>
<td>Address of the function-dependent area (SSTA control block)</td>
</tr>
</tbody>
</table>

SSIB Contents: MVS sets the following fields in the SSIB on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSIBID</td>
<td>Identifier 'SSIB'</td>
</tr>
<tr>
<td>SSIBLEN</td>
<td>Length of the SSIB (SSIBSIZE)</td>
</tr>
<tr>
<td>SSIBSSNM</td>
<td>Name of the subsystem enabled to receive this function code</td>
</tr>
</tbody>
</table>

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-SUBSYSstem tape device. IBM sets the following fields on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTAID</td>
<td>Identifier 'SSTA'</td>
</tr>
<tr>
<td>SSTAVERS</td>
<td>Current SSTA version number</td>
</tr>
</tbody>
</table>
| 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                              |

z/OS V1R11.0 MVS Using the Subsystem Interface
**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTADDAP</td>
<td>Pointer to the first DD array for this job step. Set to zero if no DD array entries exist.</td>
</tr>
<tr>
<td>SSTAHDRL</td>
<td>Length of the SSTA header</td>
</tr>
</tbody>
</table>

**Field Name** | **Description**                                                   |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTADDN</td>
<td>DD name. Blank if other than the first DD in a concatenation</td>
</tr>
<tr>
<td>SSTAJFCP</td>
<td>Pointer to the JFCB for this DD section</td>
</tr>
<tr>
<td>SSTACPOS</td>
<td>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</td>
</tr>
<tr>
<td>SSTADDF1</td>
<td>DD level information, including DISP and GDG specifications</td>
</tr>
<tr>
<td>SSTADDF2</td>
<td>DD level information byte 2, including unit affinity indicator</td>
</tr>
<tr>
<td>SSTANDRA</td>
<td>Number of devices requested</td>
</tr>
<tr>
<td>SSTDRAAP</td>
<td>Pointer to the first device request section for this DD</td>
</tr>
<tr>
<td>SSTADDAN</td>
<td>Pointer to the next device request section</td>
</tr>
<tr>
<td>SSTDRAAL</td>
<td>Length of one DD array entry</td>
</tr>
</tbody>
</table>

**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 request. For example, UNIT=(TAPE,2) would generate two device request array entries. IBM sets the following fields on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTAVOLI</td>
<td>Volume serial number. Relevant only for a specific request (indicated by bit SSTASPEC in field STAREQT)</td>
</tr>
<tr>
<td>SSTANDEV</td>
<td>Number of eligible devices</td>
</tr>
<tr>
<td>STAREQT</td>
<td>Device request information flags:</td>
</tr>
<tr>
<td></td>
<td>• SSTAPRV — indicates a private request</td>
</tr>
<tr>
<td></td>
<td>• SSTASPEC — indicates a specific request</td>
</tr>
<tr>
<td></td>
<td>• SSTADEFR — indicates volume mounting is deferred</td>
</tr>
<tr>
<td>SSTAVUID</td>
<td>Volume unit ID for affinity</td>
</tr>
<tr>
<td>STADEVP</td>
<td>Pointer to the eligible device array for this DD</td>
</tr>
<tr>
<td>SSTDRAAN</td>
<td>Pointer to the next device request array entry</td>
</tr>
<tr>
<td>SSTADRAL</td>
<td>Length of one device request array entry</td>
</tr>
</tbody>
</table>

The function routine can set the SSTAUDFR and SSTAUPRF fields in the device request section. See ["Output Parameters" on page 377](#).

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTDNUM</td>
<td>Device number, in EBCDIC. Example: The representation of device number 5B0 would be F0F5C2F0. You can use this number as</td>
</tr>
</tbody>
</table>
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 z/OS MVS Programming: Assembler Services Reference ABE-HSF, 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.)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTADSK</td>
<td>This device is skipped for this request</td>
</tr>
<tr>
<td>SSTADMND</td>
<td>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.)</td>
</tr>
<tr>
<td>SSTAONUN</td>
<td>The device is online and unallocated</td>
</tr>
<tr>
<td>SSTANAFH</td>
<td>This automatically switchable device is not assigned to another system</td>
</tr>
<tr>
<td>SSTASPCM</td>
<td>The volume mounted on this device is the one requested</td>
</tr>
<tr>
<td>SSTAACL1</td>
<td>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)</td>
</tr>
<tr>
<td>SSTAACL2</td>
<td>The installed ACL is inactive</td>
</tr>
<tr>
<td>SSTAACL3</td>
<td>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</td>
</tr>
<tr>
<td>SSTAVOLM</td>
<td>One of the following conditions can occur:</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>SSTANVOL</td>
<td>A volume is not mounted on this device for one of these possible conditions:</td>
</tr>
<tr>
<td></td>
<td>This is a specific volume request. The device is automatically switchable and the last volume dismounted from this device is not the needed volume.</td>
</tr>
<tr>
<td></td>
<td>This is a non-specific request.</td>
</tr>
<tr>
<td></td>
<td>This device is not automatically switchable.</td>
</tr>
<tr>
<td>SSTAWVOL</td>
<td>A volume is mounted on this automatically switchable device, and it matches the volume</td>
</tr>
</tbody>
</table>
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 the "Output Parameters." section.

**Output Register Information**  
Upon exit from the function routine, the general purpose registers must contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Used as a work register by the system</td>
</tr>
<tr>
<td>1</td>
<td>Address of the SSOB</td>
</tr>
<tr>
<td>2 — 13</td>
<td>Restored to contents on entry</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Return Code (Decimal)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The function routine recognized the request but did not process it.</td>
</tr>
<tr>
<td>4</td>
<td>The function routine recognized the request and processed it.</td>
</tr>
</tbody>
</table>

**Output Parameters**  
Output parameters for the function routine are:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTAUDFR</td>
<td>The field in device request section that forces a request to have mounting deferred until the dataset is actually opened</td>
</tr>
</tbody>
</table>
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.

Allocation examines all eligible devices on an individual basis. Therefore, it is unlikely that the system will need a tie-breaker.

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 SSTAPREF.

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTAINEL</td>
<td>Mark the device ineligible</td>
</tr>
<tr>
<td>SSTAUSnn</td>
<td>The remaining 1-bit fields, SSTAUS01 through SSTAUS26, can be defined and set by your function routine. Table 10 on page 369 shows how each of these bits relates to the system mask SSTAIBMM.</td>
</tr>
</tbody>
</table>

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= Tape Device Selection
* *
*01* TYPE= Sample Subsystem
* *
*01* FIRST ELIGIBLE PRODUCT= HBB5520
* *
*01* FIRST INELIGIBLE PRODUCT= HBB5510
* *
*01* OPERATION=
* This is a sample tape 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
* 03* 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 12,15Establish module base register
* LR 10,0Establish addressability
  USING SSCT,10to the SSCVT
  LR 2,1Establish addressability
  USING SSOB,2to the SSOB
* GETMAIN R,LR=84,SP=230Get working storage
  ST 13,4(1)Chain saveareas forward
  ST 1,8(13)Chain saveareas backward
  LR 13,1Point to this module's savearea
  LR 11,1Point to dynamic storage
  USING DYNAM,11Base dynamic storage
*
* Validate the request
* L 9,SSOBSSIBEstablish addressability
  USING SSIB,9to the SSIB
  CLC SSIBSSNM,SSCTSNNAMVerify the subsystem name
  BNE ERRORThis should never happen
* L 8,SSOBINDVPointer to function dependent area
*
* Check for job name beginning with HSM
*
**SSI Function Code 78**

```assembly
USING SSTA,8    Set basing
CLC HSNAME,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 SSTADD,7  Base DD entries
LTR 4,4  Check for zero
BE 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 SSTADAR,6  Base request entries
LTR 4,4  Check for zero
BE 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 SSTAEV,5  Base device entries
LH 4,SSTANDEV  Get number of devices eligible
LTR 4,4  Check for zero
BE RELOOPE  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  Mark device ineligible
OI SSTAUSE1,B'10000000'  Mark device ineligible

DEVLOOPE EQU  End of eligible device loop
LA 3,12  Get size of SSTAEVA 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 RELOOOPS  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
```

**Notes:**

- `SSTADDAP` is likely a macro or a function used to address DD entries.
- `SSTADRAP` is used to address the first request.
- `SSTADEVP` addresses the first device.
- The code checks for reserved or ineligible devices using `MAKEINEL`.
- It loops through devices and requests, marking ineligible devices.
- The use of `BNZ` (Branch if Not Zero) and `BNZ` (Branch if More to Process) suggests conditional looping to only process eligible devices or requests.
```
ERROR EQU *  
MVC SSOBRETN,'F20' 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's registers
   LA 15,0 RC=0
   BSM 0,14 Return to the SSI
*
   HSMNAME DC CL3'HSM' HSM Jobname
   HSMDEV1 DC CL4'H270' Device reserved for HSM
   HSMDEV2 DC CL4'H271' 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
   IEFSSSTA
END
```

SSI Function Code 78
Chapter 7. 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 "z/OS MVS System Messages, Vol 8 (IEF-IGD)" 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:

```plaintext
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.

### Handling Function Request Errors

**NOT Programming Interface Information**

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:

```plaintext
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:

```plaintext
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
Troubleshooting

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 z/OS MVS IPCS Commands or z/OS 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 SSIB 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 Chapter 3, “SSI Function Codes Your Program Can Request,” on page 13 or Chapter 6, “SSI Function Codes Your Subsystem Can Support,” on page 317, 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.

---

Chapter 7. Troubleshooting Errors in Your Subsystem 387
Chapter 8. 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 394
  This example documents General-Use Programming Interfaces and Associated Guidance Information.

• "Example 3 — Subsystem Function Routine (DELETEIT)" on page 396
  This example documents General-Use Programming Interfaces and Associated Guidance Information.

• "Example 4 — Subsystem Function Routine (LISTEN)" on page 397
  This example documents Product-Sensitive Programming Interfaces and Associated Guidance Information.

• "Example 5 — Subsystem Requesting Routine (TSYSCALL)" on page 399
  This example documents General-Use Programming Interfaces and Associated Guidance Information.


Example 1 — Subsystem Initialization Routine (TSYSINIT)

```assembly
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 *
```

© Copyright IBM Corp. 1988, 2009

389


Appendix A — Examples

* REGISTER USE *
* 1 - TSYSCLB *
* 10 - SSCVT *
* 11 - JSIPL *
* 12 - Code register *
* 13 - Data register *
*
* MACROS *
* CVT *
* ESTAE *
* FREEMAIN *
* GETMAIN *
* IHASDA *
* IEFJESCT *
* IEFSCVT *
* 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 1,15 report it and return
*
************************************************************************
* Invoke the IEFSSVT REQUEST(REQUEST) 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=REQUEST,SUBNAME=SSCTSNAM,SSVTDATA=ROUTINE1, *
OUTTOKEN=(2),LOADTOGLOBAL=YES,MAXENTRIES=ENTRIES, *
RETCODE=RC,RSCODE=REASON, *
MF=(E,VTURMS) *
************************************************************************
Appendix A — Examples

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 Clear control block
XC TSYSCB,TSYSCB Move in eye-catcher
MVC TSYSID(4),CBACRO Put version number in control +
block
LA 7,1 Version 1
STH 7,TSYSVER Get control block length
LA 7,CBLEN 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
Appendix A — Examples

*******************************************************************************
* 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 *
  MVC FAILSRV(L'SSVTSRV),SSVTSRV Get name of failing service
  B ERMSG Issue error message
*
SSIERR EQU *
  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,DUMMY 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,DUMMY 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),CONSNAMES=JSICNAME Issue message
  B DONE
*
INITERR EQU *
  MVC INITERRD(INITMSGL),INITERRS Copy static message
  WTO MF=(E,INITERRD),CONSNAMES=JSICNAME Issue message
  B DONE
*
ESTAERR EQU *
  MVC ESTAERRD(ESTAMSGL),ESTAERRS Copy static message
  WTO MF=(E,ESTAERRD),CONSNAMES=JSICNAME Issue message
  B RETURN
*
*******************************************************************************
* Cancel the ESTAE and return to caller. *
*******************************************************************************
DONE EQU *
  ESTAEO 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,(4)
DROP 1
SETRP WKAREA=(1),RC=4,RETDADDR=(4),FRESWDWA=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',+
CONNAME=,MF=L
SERVMSGL EQU *=SERVERRS
*
INITERRS WTO 'TSYS - SUBSYSTEM INITIALIZATION FAILED',+
CONNAME=,MF=L
INITMSGL EQU *=INITERRS
*
ESTAERRS WTO 'TSYS - SUBSYSTEM ESTAE FAILED',+
CONNAME=,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 OF           
TOKEN1 DS F       Vector table token
*      
    IEFSSVT MF=(L,VTPARMS)
    IEFSSI MF=(L,SSIDPARMS)
*    SERVERRO WTO 'TSYS ERROR IN xxxxxxx SERVICE, RETCODE xx, RSNCODE xxxx',+
     CONSNAME=,MF=L
    INITERO WTO 'TSYS - SUBSYSTEM INITIALIZATION FAILED', +
     CONSNAME=,MF=L
    ESTAERO 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
TSYSLRN DS H      Length
*            
    CBLEN EQU ≡-TSYSCB
*          
    CVT DSECT=YES CVT
*      IEFJESCT JESCT
*      IEFJSCTV 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.
* ************************************************************************

394 z/OS V1R11.0 MVS Using the Subsystem Interface
* 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's registers
LA 15,0 RC=0
Appendix A — Examples

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
*
Example 4 — Subsystem Function Routine (LISTEN)

LISTEN CSECT
LISTEN AMODE ANY
LISTEN RMODE ANY

*******************************************************************************
* 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  *
* DOM MSG=(1) returned by WTO  *
MVC SSOBRETN,=F'0' Indicate function success  *
B RETURN  *
* ERROR EQU *  *
MVC SSOBRETN,=F'4' Indicate function failure  *
* RETURN EQU *  *
L 8,4(13) Pointer to caller's savearea  *
FREEMAIN R,LV=72,A=(13) Restore caller's registers  *
LR 13,8  *
LM 14,12,12(13)  *
LA 15,0  *
BSM 0,14 Return to the SSI  *
*  *
IEFJSCVT  *
*  *
IEFSSOBH  *
*  *
IEFJSSIB  *
*  *
END

Example 4 — Subsystem Function Routine (LISTEN)
Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.

************************************************************************

************************************************************************

Attributes:

This routine must be reentrant and reside in a library accessible at the time subsystem initialization occurs.
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 SSOBDID,'C'SSOB' Set control block identifier
LA 8,SSOBSOBHSIZ
STH 8,SSOBSOBLEN Set control block size
ST 11,SSOBSOSSIB Set pointer to SSIB
Appendix A — Examples

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 TSY001 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 TSY001 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
Appendix A — Examples

IEFJSSIB

*  CVT DSECT=YES
*  IEFJESCT
*  END
Appendix A — Examples
Chapter 9. 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 Chapter 5, “Services for Building and Using Your Subsystem,” on page 301 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 296.
- 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 z/OS 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 293 for more information.

Environment

The following mapping macros are supplied by IBM and may be included in your program when invoking IEFJSVEC:

- IEFVTSPXL
- IEFJSBVT
Appendix B — Using IEFJSVEC

The requirements for the caller of IEFJSVEC are:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Authorization</td>
<td>Supervisor state with any PSW key</td>
</tr>
<tr>
<td>Dispatchable unit mode</td>
<td>Task</td>
</tr>
<tr>
<td>AMODE</td>
<td>24-bit</td>
</tr>
<tr>
<td>Cross memory mode</td>
<td>PASN=HASN=SASN</td>
</tr>
<tr>
<td>ASC mode</td>
<td>Primary</td>
</tr>
<tr>
<td>Interrupt status</td>
<td>Enabled for I/O and external interrupts</td>
</tr>
<tr>
<td>Locks</td>
<td>No locks held</td>
</tr>
<tr>
<td>Control Parameters</td>
<td>The VTSPL and JSBVT control blocks must reside in storage below 16 megabytes.</td>
</tr>
<tr>
<td>Recovery</td>
<td>The caller of IEFJSVEC should provide an ESTAE-type of recovery environment. See <a href="https://www.ibm.com">z/OS MVS Programming</a> for more information on an ESTAE-type of recovery environment.</td>
</tr>
</tbody>
</table>

**Input Register Information**

Before invoking the IEFJSVEC service, you must ensure that the following general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Address of fullword that contains the address of the subsystem VTSPL</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTSID</td>
<td>Identifier 'VTSP'</td>
</tr>
<tr>
<td>VTSLEN</td>
<td>Length of the VTSPL (VTSSIZE) control block</td>
</tr>
<tr>
<td>VTSVER</td>
<td>Version number of the VTSPL (VTSCVER) control block</td>
</tr>
<tr>
<td>VTSCONID</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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 407 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 z/OS 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 VTSP control block to binary zeros.

JSBVT Contents — Fixed Header Section: Your program sets the following fields in the JSBVT control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSBID</td>
<td>Identifier ‘JSBV’</td>
</tr>
<tr>
<td>JSBLLEN</td>
<td>Length of the JSBVT (fixed header section) control block</td>
</tr>
<tr>
<td>JSBVERS</td>
<td>Version number of the JSBVT (JSBCVERS) control block</td>
</tr>
<tr>
<td>JSBFUN</td>
<td>Number of function routines specified in this table of data</td>
</tr>
</tbody>
</table>
Appendix B — Using IEFJSVEC

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.

JSBMXFR 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSBLGTH</td>
<td>Length of this function routine’s data area and function code area (also see JSBFCOD)</td>
</tr>
<tr>
<td>JSBNME</td>
<td>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.</td>
</tr>
<tr>
<td>JSBNUM</td>
<td>Number of function codes the function routine supports</td>
</tr>
</tbody>
</table>

*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 sets the following fields on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSBFCOD</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Output Register Information**

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 14</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**Return Code Information**

IEFJSVEC returns one of the following return codes in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTSSUCES (0)</td>
<td>Successful completion. The request to build an SSVT was successfully processed.</td>
</tr>
<tr>
<td>VTSINVID (4)</td>
<td>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.</td>
</tr>
<tr>
<td>VTSINVIN (8)</td>
<td>An incorrect subsystem name was specified. Check to make sure that you specified a valid subsystem</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTSSVTAD</td>
<td>Address of the SSVT, if the SSVT build request was successful (Register 15=0)</td>
</tr>
<tr>
<td>VTSSSCVT</td>
<td>Address of the SSCVT, if the SSVT build request was successful (Register 15=0)</td>
</tr>
<tr>
<td>VTSFUNCT</td>
<td>Name of the function routine processed, if an error occurred when trying to load a function routine (Register 15=20)</td>
</tr>
</tbody>
</table>

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 409.

– 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 409.

– 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 403.

– 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 Chapter 4, “Setting Up Your Subsystem,” on page 291 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 z/OS 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Address of fullword that contains the address of the subsystem VTSPL</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTSID</td>
<td>Identifier 'VTSP'</td>
</tr>
<tr>
<td>VTSLLEN</td>
<td>Length of the VTSPL (VTSSIZE) control block</td>
</tr>
<tr>
<td>VTSCVER</td>
<td>Version number of the VTSPL (VTSCVER) control block</td>
</tr>
<tr>
<td>VTSCONID</td>
<td>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 VTSNSID 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.</td>
</tr>
<tr>
<td>VTSFLAGS</td>
<td>Flags</td>
</tr>
<tr>
<td></td>
<td>• 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).</td>
</tr>
</tbody>
</table>
|            | • 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 293 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 [z/OS MVS Programming: Authorized Assembler Services Guide](https://www.ibm.com/support/docview/wc/42218015) 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 [z/OS MVS Programming: Authorized Assembler Services Guide](https://www.ibm.com/support/docview/wc/42218015) 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSBID</td>
<td>Identifier 'JSBV'</td>
</tr>
<tr>
<td>JSBLEN</td>
<td>Length of the JSBVT (fixed header section) control block</td>
</tr>
<tr>
<td>JSBVERS</td>
<td>Version number of the JSBVT (JSBCVERS) control block</td>
</tr>
<tr>
<td>JSBFUN</td>
<td>Number of function routines specified in this table of data</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSBLGTH</td>
<td>Length of this function routine’s data area and its function code area (see the JSBFCOD field)</td>
</tr>
<tr>
<td>JSBNME</td>
<td>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.</td>
</tr>
<tr>
<td>JSBNUM</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSBFCOD</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Output Register Information**

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

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 14</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**Return Code Information**

IEFJSVEC returns one of the following return codes in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTSSUCES (0)</td>
<td>Successful completion. The request to enable was successfully processed and the SSVT has been updated.</td>
</tr>
<tr>
<td>VTSINVID (4)</td>
<td>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.</td>
</tr>
</tbody>
</table>
Appendix B — Using IEFJSVEC

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTSSSCVT</td>
<td>Address of the SSCVT, if the enable request was successful (Register 15=0)</td>
</tr>
<tr>
<td>VTSFUNCT</td>
<td>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)</td>
</tr>
</tbody>
</table>
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."
- 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:

- IEFVTSPPL
- 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 z/OS MVS Programming 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:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Address of fullword that contains the address of the subsystem VTSPL</td>
</tr>
<tr>
<td>13</td>
<td>Address of a standard 18-word save area</td>
</tr>
<tr>
<td>14</td>
<td>Return address</td>
</tr>
</tbody>
</table>

Input Parameters

Input parameter areas for the IEFJSVEC service are:

- VTSPL
- JSBVT — both fixed and variable sections
**VTSP Contents**: Your program must set the following fields in the VTSP control block on input:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTSID</td>
<td>Identifier 'VTSP'</td>
</tr>
<tr>
<td>VTSLEN</td>
<td>Length of the VTSP (VTSSIZE) control block</td>
</tr>
<tr>
<td>VTSVER</td>
<td>Version number of the VTSP (VTSCVER) control block</td>
</tr>
<tr>
<td>VTSCONID</td>
<td>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.</td>
</tr>
<tr>
<td>VTSREQ</td>
<td>Request flags - defines the operation that this call performs</td>
</tr>
<tr>
<td></td>
<td>• <strong>VTSFCDIS</strong> — Disable indicator</td>
</tr>
<tr>
<td>VTSNAME</td>
<td>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.</td>
</tr>
<tr>
<td>VTSSVTD</td>
<td>Address of SSVT table data (see JSBVT contents)</td>
</tr>
<tr>
<td>VTSCNSID</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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 <a href="https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.2.0/com.ibm.zos.v2r2.bks0100k.pdf">z/OS MVS Programming: Authorized Assembler Services Guide</a> for information on how to obtain the CART and console ID from the command input buffer (CIB) control block.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>VTSCART</td>
<td>Command and response token (CART). If a CART is provided, the SSI uses it for any messages it issues for this invocation of IEFJSVEC.</td>
</tr>
<tr>
<td></td>
<td>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 <a href="https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.2.0/com.ibm.zos.v2r2.bks0100k.pdf">z/OS MVS Programming: Authorized Assembler Services Guide</a> for information on how to obtain the CART and console ID from the command input buffer (CIB) control block.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

All other fields in the VTSP 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSBID</td>
<td>Identifier 'JSBV'</td>
</tr>
<tr>
<td>JSBLLEN</td>
<td>Length of fixed header section</td>
</tr>
<tr>
<td>JSBVERS</td>
<td>Version number (JSBCVERS)</td>
</tr>
<tr>
<td>JSBFUN</td>
<td>Number of function routines specified in this table of data</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSBLGTH</td>
<td>Length of this function routine’s data area and it’s function code area (see the JSBFCOD field)</td>
</tr>
<tr>
<td>JSBNME</td>
<td>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.</td>
</tr>
<tr>
<td>JSBNUM</td>
<td>Number of function codes</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSBFCOD</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Output Register Information**
When control returns to caller of IEFJSVEC, the general purpose registers contain:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 — 14</td>
<td>Same as on entry to call</td>
</tr>
<tr>
<td>15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

**Return Code Information**
IEFJSVEC returns one of the following return codes in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTSSUCES (0)</td>
<td>Successful completion. The request to disable was successfully processed and the SSVT has been updated.</td>
</tr>
</tbody>
</table>
Appendix B — Using IEFJSVEC

**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:
- VTSP

**VTSP Contents:** The VTSP control block contains the following information upon return from your disable request:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTSSSCVT</td>
<td>Address of the SSCVT, if the disable request was successful (Register 15=0)</td>
</tr>
</tbody>
</table>
Appendix. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to z/OS TSO/E Primer, z/OS TSO/E User’s Guide, and z/OS ISPF User’s Guide Vol I for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at: http://www.ibm.com/systems/z/os/zos/bkserv/
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.

© Copyright IBM Corp. 1988, 2009
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.

Programming Interface Information

This book primarily documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/OS.

This publication also documents information that is NOT intended to be used as Program Interfaces of z/OS. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

NOT Programming Interface Information

End of NOT Programming Interface Information
Policy for unsupported hardware

Various z/OS elements, such as DFSMS™, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Other company, product, or service names may be trademarks or service marks of others.
Index

A
accessibility 417
additional recommendations for specifying keywords related to function code 54 70
address space
  subsystem 293
application thread
  caller of the SSI function code 79 147
ASCRE macro
  using to create a separate address space 294
associate a new function routine with a supported function code
  use of 307
  with the IEFSSVT macro 307
automatic restart manager
  considerations 55

B
batch jobs
  initiating a job 8
broadcast request
  description 2
build the SSVT
  considerations 304, 403
  with the IEFJSVEC service 403
  with the REQUEST=CREATE parameter 304
BULK MODIFY SAPI Call
  SSI Function Code 79 135

C
command processing call - SSI function code 10
  considerations 346
  considerations for system symbols 347
  description 341
  restrictions 346
  sysplex considerations 347
command processing call - SSI function code 14
  considerations 350
  description 348
  restrictions 350
command sensitive area
  contents 345
  for a REPLY command 345
considerations
  automatic restart manager 55
COUNT SAPI Call
  SSI Function Code 79 135
create an SSIB
  steps 9

D
DALBRTKN text unit
  use of 136
DALDSNAM text unit
  use of 29
  use of for SSI Function Code 79 141
DALRTDDN text unit
  use of 29
DALSSREQ text unit
  use of 29
  use of for SSI Function Code 79 141
define your subsystem
  description 294
directed request
  description 2
disability 417
disable previously supported functions
  use of 306, 413
  with the IEFJSVEC service 413
  with the IEFSSVT macro 306
DUNDDDNAM text unit
  use of 29, 137
DUNOVCLS text unit
  use of 30
DUNOVDSP text unit
  use of 29
DUNOVSNH text unit
  use of 30
DUNOVSUS text unit
  use of 30
dynamic SSI 4
  description 4
early notification of end-of-task call - SSI function code 50
  description 354
enable your subsystem for new functions
  use of 306, 407
  with the IEFJSVEC service 407
  with the IEFSSVT macro 306
end-of-address space (end-of-memory) call - SSI function code 8
  description 322
end-of-task call - SSI function code 4
  description 318
environment on entry to a function routine
  description 291
  register contents 292
example
  for extended status function call - SSI function code 80 213
  for request subsystem version information call - SSI function code 54 70
  for the process SYSOUT data sets call - SSI function code 1 33
  passing accounting parameters 299
extended status function call - SSI function code 80
  example 213
Extended Status function call - SSI function code 80
description 172
external writer 14
caller of the SSI function code 1 14
considerations 30

F
first line of a multi-line WTO
WQE (major WQE) contents 334
fixed header input section
contents 63
description 60
fixed header output section
contents 65
description 65
format of the variable output sections 67
function code descriptions
for SSI function codes your program can request 13
function codes (SSI)
description 7
list of 13, 317
requirements 291
function routines
placement 293
requirements 291

H
help call - SSI function code 48
description 351

I
IBM-defined keywords
related to function code 54 68
IEAVG700 module
calling the module 327
IEFJSIPL mapping macro 302
IEFJSSIB mapping macro 8
IEFJSVEC service
disabling functions 413
enabling functions 407
using to build the SSVT 403
IEFSSNxx parmlib member
planning 294
IEFSSOBH mapping macro 7
IEFSSREQ macro
description 9
introduction 7
syntax 10
use of 9
with subsystem affinity service 314
IEFSSVT macro
disabling functions 306
enabling functions 306
replacing the function routine 307
use to build SSVT 293
use to build the SSVT 304
use to disable subsystem function codes 293
use to enable functions 293

IEFSSVTI macro
use of 305
initial program load
See IPL
initialization routine 303
specifying 296
subsystem 302
initialize your subsystem
by specifying an initialization routine 296
by using the START command 296
input to the SSVT 305
installation variable output section
contents 67
description 67
restrictions 69
integrity
subsystem considerations 292
IPL (initial program load)
considerations 295

J
JES job information services - SSI function code 71
description 87
JES Properties - SSI function code 82
description 217
JES2 subsystem 1
JES3 subsystem 1
JESSPOOL SAF resource class
use of for Function Code 79 137, 144

K
keyboard 417

L
life-of-job SSIB
description 8
use 17, 148
load-to-global option 293

M
maintain information about subsystem callers
subsystem affinity entry 313
using the subsystem affinity service 313
make a request of a subsystem
summary of steps 11
using the IEFSSREQ macro 9
MCSOPER/MCSOPMSG macro services
use of 328
multi-line use information
for SSI function code 9 337
multi-line WTO
for SSI function code 9 333
SSWT contents 333
subsequent lines 336
WQE (minor WQE) contents 336
N
name your subsystem
  restrictions 294
notices 419
notify user message service call - SSI function code 75
description 128

P
placement of function routines 293
  setting the load-to-global option 293
primary subsystem
description 1
procedure of searching data strings 68
process SYSOUT data sets call - SSI function code 1
description 14
  example 33
  retrieval attributes 14
  update attributes 14
processing flow for single data set requests
  for process SYSOUT data sets call - SSI function code 1
    steps 28
  processing all data sets together 30
  processing one data set at time
    steps 28
PUT/GET SAPI Call
  SSI Function Code 79 135

R
recovery
  subsystem considerations 292
register contents
  on entry to a function routine 292
request a function of a subsystem
  steps 7
request command processing information
description 341, 348
request job ID call - SSI function code 20
description 49
  restrictions 55
request subsystem version information
  installation-defined keywords 69
request subsystem version information call - SSI function code 54
description 60, 358
  example 70
request types
  for SYSOUT Application Program Interface 134
restrictions for SSI function code 10 346
restrictions for SSI function code 14 350
return code information
  for command processing call - SSI function code 10 345
  for delete operator message - SSI function code 14 350
  for early notification of end-of-task call - SSI function code 50 357
  for end-of-address space (end-of-memory) call - SSI function code 8 325
  return code information (continued)
    for end-of-task call — SSI function code 4 321, 353
    for extend status function call - SSI function code 80 192
    for Initiator Information - SSI Function Code 82 253
    for JES job information services - SSI Function Code 71 92
    for JES Job Information Services - SSI Function Code 71 99, 109, 119, 124
    for JESPLEX Information - SSI Function Code 82 267
    for job Class Information - SSI Function Code 82 278
    for NJE NODE Information - SSI Function Code 82 224
    for notify user message service call - SSI function code 75 131
    for process SYSOUT data sets call - SSI function code 1 24
    for request job ID call - SSI function code 20 53
    for request subsystem version information call - SSI function code 54 64, 360
    for return job ID call - SSI function code 21 59
    for scheduler facilities call - SSI function code 70 83
    for SMF SUBPARM option change call - SSI function code 58 366
    for SPOOL Volume Information - SSI Function Code 82 238
    for SYSOUT Application Program Interface - SSI function code 79 161
    for user destination validation/conversion - SSI function code 11 43
    for verify subsystem function call - SSI function code 15 48
    for WTO/WTOR call - SSI function code 9 338
return codes from a directed request
  list of 10
  return job ID call - SSI function code 21
description 56

S
scheduler facilities call - SSI function code 70
description 77
Scheduler Work Blocks (SWBs)
  use of in Function Code 79 140
secondary subsystem
description 1
services for building and using your subsystem
  activating your subsystem 307
  adding your subsystem 301
  changing what your subsystem can do 305
  deactivating your subsystem 308
  defining subsystem options 310
  defining what your subsystem can do 304
description 301
initializing your subsystem 302
maintaining information about subsystem
callers 313
querying subsystem information 311

Index 425
services for building and using your subsystem
(continued)
storing and retrieving subsystem-specific information 310
swapping subsystem functions 309
services for writing your subsystem
changing what your subsystem can do 407
defining what your subsystem can do 403
SET SMF=xx command
use of 298
set up the environment
to make a request of a subsystem 7
SETSMF command
use of 298
setting up your subsystem
planning considerations 291
shortcut keys 417
single-line WTO
WQE contents 332
SJFREQ macro
use of in Function Code 79 140
SMF console command
use of 298
SMF parmlib member (SMFPROMxx)
use of 298
SMF SUBPARM option
initializing the SMF parameters 298
initializing the subsystem 298
modifying the SUBPARM value 298
processing 298
SMF SUBPARM option change call - SSI function code 58
description 364
SMFCHSUB macro
use of 299
specifying keywords
related to function code 54 69
SSAFF macro
description 314
obtain a value from an entry 314
parameters 315
DATA parameter 315
ENTRY parameter 315
OBTAIN parameter 315
SET parameter 315
symbol 315
TCB parameter 315
set a value in an entry 314
syntax 314
use 314
use of 314
SSCVT (subsystem communication vector table) 292
address of the SSCVT 292
SSI (subsystem interface) 1
attributes 1
description 1
error handling 383
examples 389
introduction 1
troubleshooting errors 383
SSI Function Code 54
use of in SSI Function Code 79 136
SSI function code descriptions
for SSI function codes your program can request 13
for SSI function codes your subsystem can support 317
SSI function codes
list of 13, 317
SSI function codes your program can request
list of 13
SSI function codes your subsystem can support
list of 317
SSI processing
controlling 3
SSIB (subsystem identification block)
description 8
SSIB data area 8
SSIBID field
setting the field to create an SSIB 9
SSIBJBID field
setting the field to create an SSIB 9
SSIBLEN field
setting the field to create an SSIB 9
SSIBSSNM field
setting the field to create an SSIB 9
SSIBSUSE field
setting the field to create an SSIB 9
SSOB (subsystem options block)
description 7
SSOB data area 7
SSOB function dependent area
description 8
SSRTDIST return code value
from a directed request 11
SSRTLERR return code value
from a directed request 11
SSRTNOSS return code value
from a directed request 11
SSRTNSUP return code value
from a directed request 11
SSRTNTUP return code value
from a directed request 11
SSRTOK return code value
from a directed request 10
SSS2BTOK field
use of 141
SSS2BULK request
use of 134
SSS2CDS field
use of 140, 142
SSS2COUN field
use of 134
SSS2CTRL field
use of 135, 141, 143
SSS2DES2 field
use of 135
426  z/OS V1R11.0 MVS Using the Subsystem Interface
SSS2DESR field
use of 135
SSS2DEST field
use of 135
SSS2DSN field
use of 136
SSS2ECBP field
use of 139, 143, 145, 150
SSS2EODS field
use of 140, 141
SSS2FSWB field
use of 140
SSS2FSWT field
use of 140
SSS2JEST field
use of 135
SSS2PUGE request
use of 134
SSS2RBA field
use of 150
SSS2RET2 field
use of 136
SSS2RLSE field
use of 143
SSS2ROUT field
use of 143
SSS2SETC field
use of 143
SSS2SWBT field
use of 140
SSS2SWTU field
use of 140
SSS2UFLG field
use of 150
SSS2WRSN field
use of 140
SSS2WRTN field
use of 140
SSSOFORM field
use of 19
SSSOFOR8 field
use of 19
SSSOWTRC field
contents of on return from the IEFSSREQ macro 32
SSWT contents for a WTO (always single-line) 337
SSWT contents for the first line of a multi-line WTO 333
START command
using to initialize your subsystem 296
started task
initiating a started task 8
subsystem 1
broadcast request 291
considerations 3
defining to MVS 294
description 1
diagnosing errors 383
directed request 291
error handling 383
examples 389
functions 291
subsystem (continued)
how to name it 294
IEFSSREQ macro 9
initialization routine 302
initializing 296
integrity 292
MVS use 291
passing accounting parameters 298
recovery 292
request types 2, 291
broadcast request 2
directed request 2
requesting a function 7
setting up your subsystem 291
subsystem affinity service 313
types 1
primary 1
secondary 1
writing your own subsystem 291
subsystem affinity service
description 313
SSAFF OBTAIN request 314
SSAFF SET request 314
subsystem communication vector table
See SSCVT
subsystem identification block
See SSIB
subsystem initialization routine
description 296
examples 296
subsystem options block
See SSOB
subsystem requests
broadcast 2
directed 2
SWBTUREQ macro
use of in Function Code 79 140
SYSOUT Application Program Interface (SAPI) - SSI function code 79
description 134
sysplex
command processing SSI call - SSI function code
10 347
system message
controlling 327
with SSI function code 9 327
system symbols
command processing SSI call - SSI function code
10 347
system variable output section
contents 67
description 67
T
tape device selection call - SSI function code 78
description 368
troubleshoot errors in your subsystem
common types of errors 383
description 383
handling initialization errors 383

Index 427
TSO/E user
    initiating a LOGON  9

types of subsystem requests
    broadcast  2
    directed  2

U
unique attributes of the SSI
    description  1

user destination validation/conversion - SSI function
code 11
    description  40

V
verify subsystem function call - SSI function code 15
description  45
VTSCREAT SSVT build indicator  405
VTSGLOAD load-to-global indicator  404

W
Wildcards
    SSI Function Code 79  135
WQE (major WQE) contents for the first line of a
    multi-line WTO  334
WQE (minor WQE) contents for subsequent lines of a
    multi-line WTO  336
WQE contents for a single-line WTO  332
write your own subsystem
    considerations  3
writer communication area
    contents  32
    description  32
writing your own subsystem
    decisions you must make  296
    recovery and integrity considerations  292
    steps  291
WTO/WTOR call - SSI function code 9
description  327
WTOR (always single-line)
    SSWT contents  337
Readers’ Comments — We’d Like to Hear from You

z/OS
MVS Using the Subsystem Interface

Publication No. SA22-7642-09

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your 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. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.
Submit your comments using one of these channels:
• Send your comments to the address on the reverse side of this form.
• Send your comments via e-mail to: mhvrdfs@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name

Address

Company or Organization

Phone No.

E-mail address
Readers’ Comments — We’d Like to Hear from You

SA22-7642-09

Business Reply Mail

First-Class Mail  Permit No. 40  Armonk, New York

Postage Will Be Paid By Addressee

IBM Corporation
MHVRCS, Mail Station P181
2455 South Road
Poughkeepsie, NY
12601-5400

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES